MENDELOVA ZEMĚDĚLSKÁ A LESNICKÁ UNIVERZITA V BRNĚ
FAKULTA PROVOZNĚ EKONOMICKÁ
ÚSTAV INFORMATIKY

DIPLOMOVÁ PRÁCE

Znakové sady v typografických systémech

Jméno a příjmení diplomanta: Martin Černý

Vedoucí diplomové práce: Ing. Jiří Rybička Dr.

BRNO 1999

Mendelova zemědělská a lesnická univerzita v Brně
Fakulta provozně ekonomická
Ústav informatiky

Diplomová práce

ZNAKOVÉ SADY V TYPOGRAFICKÝCH SYSTÉMECH

Martin Černý

Brno 1999

Poděkování

Děkuji vedoucímu diplomové práce Ing. Jiřímu Rybičkovi Dr. za metodické vedení, odborné připomínky a za poskytnuté rady při konzultacích v průběhu zpracování diplomové práce.

Upozornění

Na tomto místě bych chtěl varovat před bezduchým užíváním konverzních programů a volně citovat vedoucího diplomové práce: *„Automatické konverze fontů mohou být silným nástrojem; ale také silnou zbraní v rukou diletanta.“*

Obsah

1. Písmo a fonty ... **12**
1.1. Písmo a jeho klasifikace ... 12
1.2. Počítačové systémy a písmo ... 12
1.3. Konverze a úpravy fontů ... 14
1.4. Formáty fontů ... 14
1.4.1. Postscriptové fonty ... 15
1.4.2. TrueType fonty ... 16
1.4.3. Hlavní rozdíly mezi Type 1 a TrueType ... 16
2. Postscriptové fonty ... **18**
2.1. Adobe Type 1 ... 18
2.1.1. Stavba znaků ... 18
2.1.2. Slovník fontu ... 18
2.1.3. Typický soubor Type 1 ... 19
2.1.4. Kódovaná část ... 21
2.1.5. Unique Identification Numbers a jména fontů ... 22
2.1.6. Popis znaku ... 22
2.1.7. Alignments and Overshoots (Vyrovnání a přesahy) ... 23
2.1.8. Character Paths (Cesty znaku) ... 24
2.1.9. Technika kresby ... 25
2.2. Adobe Font Metrics File Format ... 25
2.2.1. Struktura AFM souboru ... 26
2.2.2. Global Font Information ... 26
2.2.3. Writing Direction Information ... 27
2.2.4. Individual Character Metrics ... 27
2.2.5. Kerning Data ... 28
2.2.6. Composite Chracter Data ... 29
3. TrueType fonty ... **31**
3.1. TrueType ... 31
3.1.1. Obrysy ... 31
3.1.2. FUnits, em square a souřadnicový systém ... 31
3.1.3. Grid-fitting (úprava rastru) obrysu ... 32
3.1.4. TrueType interpret ... 33
3.1.5. Používání instrukcí ... 33
3.1.6. Storage Area ... 34
3.1.7. Graphics State ... 34
3.1.8. Scan Converter ... 35
3.1.9. Dropouts ... 35
3.1.10. TrueType soubor ... 36
3.1.11. cmap (Character To Glyph Index Mapping Table) ... 37
3.2. TrueTypeOpen ... 39
4. Metafontové fonty ... **40**
4.1. Program METAFONT ... 40
4.2. Jazyk METAFONTu ... 40

4.2.1. Souřadnicový systém, body a cesty ... 40
4.2.2. Algebraické výrazy a proměnné ... 41
4.2.3. Pera, výplně a gumy ... 42
4.2.4. Podmínky, cykly a makra ... 42
4.2.5. Běžná práce s METAFONTem ... 43
4.2.6. Metafontový soubor ... 43
4.3. Produkty METAFONTu ... 44
4.3.1. Soubor TFM a PL ... 44
4.3.2. Soubor VF a VPL ... 46
4.3.3. Soubor GF, PK, PXL a PKedit ... 48
5. Fonty v TEXu ... 49
5.1. TEX ... 49
5.1.1. Práce TEXu ... 49
5.1.2. Kódování v TEXu ... 49
5.2. Jak TEX pracuje s fonty ... 50
5.2.1. Základní příkazy pro práci s fonty ... 50
5.2.2. Virtuální fonty ... 51
5.2.3. Makro NFSS ... 52
5.2.4. Standardní postscriptové fonty ... 53
5.2.5. TEX a Postscript: DVIPS, PSTricks, GhostScript ... 54
5.2.6. Adresáře instalace EmTEX ... 55
6. Převody fontů do prostředí TeXu ... 56
6.1. Teoretický základ ... 56
6.1.1. Principy konverzí z hlediska prostředí TEXu ... 56
6.1.2. Principy konverzí z hlediska počeštění fontů ... 58
6.2. Získání metrik ... 59
6.2.1. Program Afm2tfm ... 59
6.2.2. Ttf2tfm ... 61
6.2.3. Program Af2pl ... 63
6.2.4. Program Accents a L2Accents ... 63
6.2.5. Fontinst ... 65
6.2.6. Cspsfont ... 70
6.2.7. Program A2Ac ... 71
6.3. Získání bitmap ... 74
6.3.1. PS2PK ... 75
6.3.2. Ttf2PK ... 76
6.3.3. GsftoPK ... 77
6.3.4. TTFtoGF ... 77
6.4. Získávání metafontových souborů ... 78
6.4.1. PS4MF ... 79
6.4.2. TTF2MF ... 80
6.4.3. Fog2MF ... 82
6.4.4. Ega2MF a Vga2MF ... 82
6.5. Utility pro práci s fonty ... 83
6.5.1. Qdtexvpl ... 83

6.5.2. Bm2Font ... 83
6.5.3. Pkbbox ... 85
6.5.4. RUMgraph ... 85
6.5.5. PXtoPK, PKtest, Pbm2PK a Pbm2TEX ... 86

7. Převody a úpravy postscriptových a TrueType fontů ... 87

7.1. Type 1 utility ... 87
7.1.1. T1utils ... 87
7.1.2. Pfm2Afm a Pf2Afm ... 88
7.1.3. Mmpfb a Mmafm ... 88
7.1.4. T1tools ... 89
7.1.5. T1Accent ... 89
7.1.6. Další utility ... 89

7.2. TrueType utility ... 90
7.2.1. Ttf2Pfa ... 90
7.2.2. TTFDump ... 91
7.2.3. TTOAsm a TTODasm ... 92
7.2.4. Addtable ... 93
7.2.5. TtfMap ... 93

7.3. Profesionální nástroje pro postscriptové a TrueType fonty ... 93
7.3.1. FontMonger ... 94
7.3.2. Fontographer ... 94
7.3.3. Aplikace firmy FontLab ... 94
7.3.4. Další nástroje ... 94

8. Hodnocení a návrh konverzní cesty do prostředí TEXu ... 95

8.1. Hodnocení konverzních programů ... 95
8.1.1. Konverze metrických informací ... 95
8.1.2. Konverze obrysů na bitmapy ... 95
8.1.3. Konverze do metafontových zdrojů ... 96

8.2. Grafický model konverzí ... 96

8.3. Návrh optimální konverzní cesty ... 98
8.3.1. Pro postscriptové fonty ... 98
8.3.2. Pro TrueType fonty ... 98

9. Návrh konverzního systému pro prostředí TEXu ... 100

9.1. Postup při konverzích fontů ... 100
9.1.1. Postup u postscriptových fontů ... 100
9.1.2. Postup u TrueType fontů ... 101

9.2. Dávky a jejich použití ... 101

10. Přílohy ... 104

10.1. Příklady popisované v diplomové práci (disketa) ... 104

10.2. Dávky a programy pro automatizaci konverzí (disketa) ... 104

11. Závěr ... 106

12. Použitá literatura ... 107

Seznam obrázků

Obrázek 1: *Odstínění interpretu fontů od aplikace* ... **15**
Obrázek 2: *Uspořádání Type 1* ... **19**
Obrázek 3: *Horizontální a vertikální tahy (stems), patky (serifs)* ... **22**
Obrázek 4: *Počátek (origin), délka znaku (width), sidebearing* ... **23**
Obrázek 5: *Baseline, x-height, cap-height a jejich přesahové osy* ... **24**
Obrázek 6: *Tvorba cest ve správném směru* ... **24**
Obrázek 7: *Stupně plošného (track) kerningu* ... **28**
Obrázek 8: *Párový (pair-wise) kerning* ... **29**
Obrázek 9: *Znaky s jednou, dvěma a třemi konturami* ... **31**
Obrázek 10: *Dva možné způsoby umístění počátku v latinkových fontech* ... **32**
Obrázek 11: *72 a 127 bodové M v jejich em squares; upem se rovná 8 v obou případech* ... **32**
Obrázek 12: *Obrys bez grid-fitting (vlevo), s grid-fitting (vpravo) a výsledek rastrování* ... **33**
Obrázek 13: *Výpočet Winding number pro body p1 a p2* ... **35**
Obrázek 14: *Body a cesty v Metafontu* ... **41**
Obrázek 15: *Typiká práce TEXu, ovladačů a převodníků* ... **50**
Obrázek 16: *Prostředí TEXu* ... **56**
Obrázek 17: *Afm2tfm — tvorba různých variant písma* ... **60**
Obrázek 18: *TrueType TimesNewRoman* ... **63**
Obrázek 19: *Porovnání výstupů csb10 a vcmb10* ... **64**
Obrázek 20: *Počeštění Helvetiky Bold prostřednictvím L2accents* ... **64**
Obrázek 21: *Český ITC-GaramondBoldCondensed* ... **72**
Obrázek 22: *Rastrování Garamond-Bold PS2PK (navazuje na Afm2tfm a A2Ac)* ... **75**
Obrázek 23: *Bodoni.ttf ve formátu PK* ... **76**
Obrázek 24: *Program TTFtoGF* ... **78**
Obrázek 25: *Písmo Times, převedené PS4MF* ... **80**
Obrázek 26: *Definice ligatur v programu TTF2MF* ... **81**
Obrázek 27: *Bookman.ttf převedený pomocí TTF2MF* ... **81**
Obrázek 28: *Použití Bm2font* ... **85**
Obrázek 29: *Medusa Type 3 v TEXu a GhostScriptu prostřednictvím Ttf2Pfa* ... **91**
Obrázek 30: *Automatický konverzní systém pro postscriptové fonty* ... **98**
Obrázek 31: *Automatický konverzní systém pro TrueType fonty* ... **99**
Obrázek 32: *Dávka pro konverzi postscriptových fontů (zkráceno)* ... **101**
Obrázek 33: *Dávka pro konverzi NonCZ TrueType fontů (zkráceno)* ... **102**
Obrázek 34: *Dávka pro konverzi CZ TrueType fontů (zkráceno)* ... **103**

Seznam příkladů

Příklad 1: *Soubor AFM (ITC Garamond Bold Condensed)* **29**
Příklad 2: *Příklad položek CVT* **34**
Příklad 3: *Příklad Directory table* **36**
Příklad 4: *Záhlaví podtabulky 2* **39**
Příklad 5: *Zvětšené písmo 8pt a 10pt* **40**
Příklad 6: *Body a cesty v Metafontu* **41**
Příklad 7: *Příklad podmínky, cyklu a makra* **42**
Příklad 8: *Příkad MF souboru Bodoni* **44**
Příklad 9: *Soubor TFM — hlavička* **45**
Příklad 10: *Soubor TFM — ligatury a kerning* **45**
Příklad 11: *Soubor TFM — informace o znacích* **46**
Příklad 12: *Soubor VPL — hlavička* **46**
Příklad 13: *Soubor VPL — definice znaků* **47**
Příklad 14: *Zavedení fontu v TEXu* **50**
Příklad 15: *Soubor cpalatin.tex (Palatino)* **53**
Příklad 16: *Použití neběžných velikostí postscriptových fontů.* **53**
Příklad 17: *Soubor náhrad RPL* **62**
Příklad 18: *Příklad adjustačního souboru hvb.adj* **64**
Příklad 19: *Fontinst příklady:* **69**
Příklad 20: *FD soubor fontu Bookman* **71**
Příklad 21: *Kódový soubor TTFtoGF CTF* **78**
Příklad 22: *Kódový soubor pro TTF2MF* **80**
Příklad 23: *Vstup a výstup programu Fog2MF* **82**
Příklad 24: *Použití Qdtexvpl (test.tex)* **83**
Příklad 25: *Získaný VPL soubor (novy.vpl)* **83**
Příklad 26: *Ukázka souboru makefile* **86**
Příklad 27: *Ukázka práce T1utils* **87**
Příklad 28: *Změněná dávka pf2afm.bat pro Windows 95* **88**
Příklad 29: *Jednoduché testování Medúzy v GhostScriptu (test.ps)* **90**
Příklad 30: *TTFDump — příklady použití* **91**
Příklad 31: *Kerningová tabulka fontu TimesNewRoman* **91**

Seznam tabulek

Tabulka 1: *Offsetová tabulka TrueType fontu* **36**
Tabulka 2: *Povinné tabulky TrueType fontu* **37**
Tabulka 3: *Nepovinné tabulky TrueType fontu* **37**
Tabulka 4: *Platform a Encoding ID* **38**
Tabulka 5: *Tabulky TrueType Open* **39**
Tabulka 6: *Tabulka zkratek měrných systémů* **51**
Tabulka 7: *Postscriptové fonty v instalaci TEXu* **53**
Tabulka 8: *Hodnocení konverzí metrických informací* **95**
Tabulka 9: *Hodnocení získávání bitmap* **96**
Tabulka 10: *Hodnocení získávání MF zdrojů* **96**
Tabulka 11: *Seznam příkladů (adresářů)* **104**
Tabulka 12: *Adresáře konverzního systému* **105**
Tabulka 13: *Programy konverzního systému* **105**

Úvod

Ve slovnících je typografie vysvětlována jako knihtiskařství; obor zahrnující sazbu a knihtisk; výtvarné a technické řešení tiskoviny. Dnes lze možnosti tohoto oboru realizovat na osobním počítači za pomoci speciálních typografických systémů, pro které se používá název DTP systémy.

V současnosti se na trhu vyskytují tři druhy systémů, na kterých lze připravit nějaký dokument: systémy prvoplánově koncipované pro tvorbu tiskovin, systémy původně zaměřené na emulaci psacího stroje, které rozvojem aspirují na typografické systémy, a programy fungující jako různé textové editory s cílem zvýšené funkčnosti a výkonnosti. Každý z těchto systémů je vhodný pro jiný druh práce nebo etapu vývoje tiskoviny.

Příprava tiskoviny zahrnuje mimo jiné i výběr a přípravu písma. Písma se nazývají v typografických systémech fonty. Font lze definovat jako sadu písmových znaků, číslic a symbolů jednoho typu písma. Fonty vznikají podle vzorů již existujících typů nebo jsou vytvářeny nové návrhy. Fonty můžeme rozdělit na dva hlavní druhy: vektorové a bitmapové. Bitmapový font obsahuje znaky v podobě vysvícených bodů určitého rastru. Tento druh není příliš v DTP systémech užíván, protože pro kvalitní obraz je nutné při výstupu dodržet rastr fontu. Používanější jsou vektorové fonty, kde jsou znaky popsány vektory, které se na výstupní rastr přepočítávají.

Tato práce analyzuje potřebu konverzí a úprav fontů, popisuje vektorové formáty, způsoby získání bitmapových obrazů pro různé rastry, důvody a možnosti vzájemné převoditelnosti mezi formáty a co takovými konverzemi můžeme získat nebo případně ztratit. Dále řeší problematiku přípravy kvalitního českého písma pro typografický systém — počešťování fontů. Vzhledem k opodstatněnosti a rozsahu se tento problém řeší pro použití různých formátů fontů v TEXu.

Cíl práce

Cílem práce je popis nejpoužívanějších znakových sad v typografických systémech, příprava kvalitního písma pro typografický systém, zhodnocení možností a kvality vzájemných konverzí, popis a analýza programů umožňující vzájemnou konverzi a návrh konverzního systému, který zohledňuje požadavky na českou sazbu.

Prostředí testů

Programy byly testovány v prostředí MS Windows 95 a emulovaném okně MS DOS v témž systému. Některé programy, které nejsou upraveny pro běh v tomto prostředí (např Mmpfb), byly testovány v systémech, pro které existuje spustitelná verze; většinou se jedná o různé implementace Unix (Solaris).

Testování výsledků bylo realizováno v instalaci EmTEXu šířené sdružením CSTUG CSTEX95. Všechny adresářové struktury, nastavení prostředí a práce s horní pamětí vychází z této instalace. Různé úpravy programů byly prováděny v závislosti na této instalaci.

Začlenění a použití výsledků je demonstrováno na příkladech v plain TEXu s výjimkou programů určených pouze pro formát LATEX.

1. Písmo a fonty

1.1. Písmo a jeho klasifikace

Písmo prošlo dlouhým a složitým vývojem. Původ dnešního písma užívaného v Evropě i Americe se odvozuje od písem vzniklých v Egyptě a Mezopotámii. Řecká archaická abeceda prokazuje návaznost na fénický vzor. Řekové vytvářejí znaky pro samohlásky a postupně opouštějí psaní zprava doleva. Tak dotvářejí abecedu v pevnou soustavu 24 písmen geometrického charakteru a jasného tvarového řádu — řecká klasická abeceda. Z hlediska latinky jsou dovršiteli vývoje a vlastními tvůrci naší velké abecedy staří Římané, kteří tu stejně jako v řadě jiných oborů přejímali a rozvíjeli řeckou vzdělanost. (SLEZÁK, 1985)

Písmo se vyvíjelo mnoho století podobně jako architektura; můžeme mluvit o gotickém písmu nebo renesančním, moderním, ... S vynálezem knihtisku se objevují nová písma a označení nesou často podle svého tvůrce, např. Bodoni. Historie a vývoj písma je popsán v řadě dobrých knih.

Všechna písma se dělí do čtyř tříd:

1. latinková písma
2. nelatinková písma pravosměrná (např. slovanská, řecká)
3. nelatinková písma levosměrná (orientální)
4. ostatní písma (exotická)

Latinková písma se dělí na čtyři druhy:

1. antikvová písma
2. lineární
3. psaná
4. lomená

Všechna tato písma se dělí do jedenácti klasifikačních skupin z nichž každá má tři podskupiny, a to písma základního řezu, písma vyznačovací nakloněná a písma zdobná, do nichž se řadí i všechna písma tučného a velmi tučného duktu. Rozdělení i popis např. v (JAK PUBLIKOVAT NA POČÍTAČI, 1996) nebo (SLEZÁK, 1985)

Pro každou příležitost je vhodné jiné písmo. Nějaké písmo je vhodné pro sazbu beletrie, jiné je vhodnější pro technické příručky, další na svatební oznámení a úplně jiné pro reklamní plakáty. Velmi citlivě musíme volit, pokud chceme v jednom dokumentu kombinovat více druhů písma. Některá písma se spolu snášejí dobře a některé kombinace jsou přímo proti základním typografickým pravidlům.

Písmo je třeba vlastnit ve více řezech — kromě základní varianty alespoň ještě vyznačovací a tučný řez. Samozřejmě, že čím více řezů máme dostupných, tím lépe. Pro různé části dokumentu je vhodné používat různé řezy téže rodiny písma, než volit kombinace z různých rodin.

1.2. Počítačové systémy a písmo

Počítačové technologie ovlivnily práci v grafice, designu i typografii. Nástup DTP systémů umožnil produkovat tiskoviny bez nijak zvlášť nedostupného

vybavení — počítač, tiskárna, vhodný program. Zpočátku počítač fungoval jako psací stroj nové generace, ale s rozvojem hardware a software došlo k rozšíření DTP a grafických systémů.

Samozřejmě bylo třeba vytvořit nová nebo převést již existující písma do podoby srozumitelné počítači. Vzniklo několik systémů počítačové interpretace písma. Dnes existují tisíce digitalizovaných písem. Technická realizace písmového návrhu je snadná jako nikdy předtím.

Písma vyplňují rozličné výplně, které můžeme měnit nebo přímo vytvářet několika málo úkony. Písmo lze různě naklánět, přidávat perspektivu, stíny, různé prostorové efekty, usazovat na křivku nebo dokonce přímo v textu měnit kresbu jednotlivých znaků.

Existuje množství snadno dostupných digitalizovaných písem šířených jako freeware nebo shareware, jsou stále vyvíjeny nové standardy počítačového zápisu písma, ale také došlo i k jisté degradaci řemeslné preciznosti. Mnoho písem je možná umělecky zajímavá, ale typograficky patrně omezeně použitelná. Jsou šířeny písma s nízkou čitelností nebo chybějícími znaky. Písma jsou někdy dodávána v jednom řezu a ty ostatní jsou počítačem konstruovány algoritmicky. Velmi často je používána stejná kresba písma pro všechny velikosti, někdy jsou špatné nebo dokonce úplně chybí kerningové informace. Přes všechny nedostatky, zapříčiněné spíše neznalostí uživatelů a opojením z nových, dříve netušených možností, můžeme hovořit o výrazném posunu v práci s písmy.

Kvalitní počítačové písmo musí obsahovat:

- kvalitní obrys, který si zachovává dobrou čitelnost při změnách velikosti
- propracované metrické, kerningové a slitkové informace
- úplnou sadu znaků a symbolů včetně speciálních znaků

Kvalitní typografický systém musí umět:

- používat a správně interpretovat všechny metrické, kerningové a slitkové informace pro sestavení typograficky bezchybné tiskové strany
- poskytovat způsob získání hotové tiskoviny bez změn tiskové strany s maximální kvalitou vyrastrovaných tištěných znaků a symbolů

V DTP potřebujeme kvalitní písmo i typografický systém. Kvalitní obrys písma poznáme snadno při vytištění zkušebního textu, kde testujeme chování písma při různých velikostech v zamýšleném výstupním rastru. Složitější je posouzení metrických a kerningových informací — musíme sledovat umístění znaků vedle sebe, zda nevznikají příliš těsné nebo vzdálené dvojice, které narušují plynulost textu; pro některé dvojice písmen je nutné jakési dodatečné upravení pozic (vyrovnání, kerning), kde se pomyslné obdélníky písmen překrývají nebo dodatečně vzdalují. Je zřejmé, že potřebujeme všechny znaky, které se vyskytují v dokumentu; dále by měl font obsahovat slitky a algoritmus jejich používání. Nejčastějšími slitky jsou dvojice fi a fl, ale písmo může obsahovat i další (expert encoding pro Type 1 fonty zahrnuje ještě ff, ffi a ffl). Mezi speciální znaky můžeme řadit různé varianty téhož znaku, volba se provádí podle potřeb sazby nebo podle pravidel daného jazyka.

Vlastnictví kvalitního fontu nezaručuje kvalitní sazbu. Typografický systém musí obsahovat výkonný algoritmus lámání řádek a stran, který umí používat všechny vlastnosti a informace fontu. Často se můžeme setkat s tím, že program aspirující na typografický systém sází pouze podle metrických informací a úplně ignoruje kerningové informace, i když je font obsahuje. Podobně je tomu se slitky, které jsou často v takovém systému naprosto nedostupné. Tím dochází ke znehodnocení úsilí tvůrců fontu i tiskoviny. Příkladem systému, který znehodnocuje úsilí lidí podílejících se na tvorbě dokumentu, je Microsoft Word.

Typografický systém musí obsahovat mechanismus pro získání podoby dokumentu vytištěného na papír. Jde buď o přímý tisk nebo o export do nějakého obecného grafického formátu. Nejlepší je, pokud obsahuje obě možnosti. Při výstupu nesmí docházet k dodatečným změnám tiskové strany, protože ty jsou už mimo kontrolu sazeče a jeho předchozí práci opět znehodnocují.

1.3. Konverze a úpravy fontů

Při výběru písma se řídíme určitými pravidly. Při výběru fontu musíme uvažovat v kontextu používaného typografického systému a s vlastnostmi vybraného fontu. Musíme posoudit vhodnost vybraného formátu fontu a případné úpravy nebo nutné konverze. Nejdůležitějším a prvním krokem je volba *kvalitního* fontu; čím kvalitnější font použijeme, tím méně dodatečných úprav bude nutné provést.

Hlavní důvody pro konverzi mezi formáty:

1. získat formát akceptovaný typografickým systémem

2. využít vlastností fontu, které ve výchozím formátu nejsou použitelné

Je zřejmé, že s fontem musí umět pracovat typografický systém. U většiny systémů je font užíván prostřednictvím operačního systému — fonty obsažené v OS jsou nabídnuty i v aplikaci. Jiné programy mají vlastní formát a způsob zařazení fontu do aplikace, například TEX.

Převedením fontu do jiného formátu můžeme využít více jeho vlastností. Například úspěšnou konverzí postscriptových nebo TrueType fontů do prostředí systému TEX můžeme využívat kerningové informace a ligatury, což v aplikacích typu MS Word není možné. Takovou konverzí a následným použitím dochází k plnému využití vlastností fontu a zlepšení typografické kvality tiskoviny. Často není třeba konverze celého fontu, ale pouze jeho části — nejčastěji metriky.

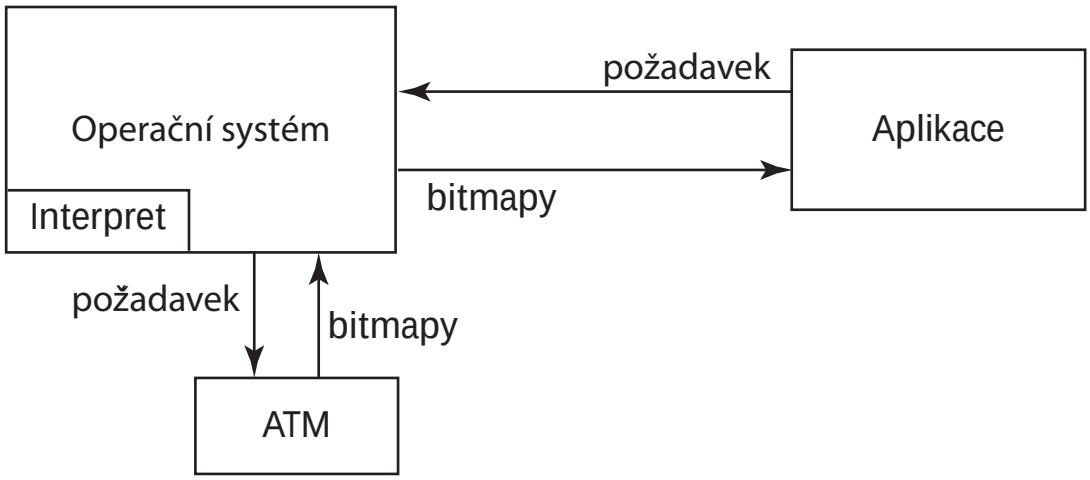
1.4. Formáty fontů

Za dobu existence počítačů vzniklo mnoho formátů fontů, často vázaných na určitý program nebo výstupní zařízení. S rozvojem hardware a operačních systémů docházelo k tomu, že fonty a jejich obsluha se stávaly nedílnou součástí OS.

V současné době můžeme mluvit o dvou dominujících formátech fontů. Oba formáty jsou vektorové a obrysové. Obrysové znamená, že je vytvořen obrys písma a ten je vyplněn podle určitých pravidel.

1. Postscriptové fonty firmy Adobe systems
2. TrueType fonty firmy Apple a Microsoft

Obrázek 1: *Odstínění interpretu fontů od aplikace*



Je vhodné zmínit formát fontů pro typografický systém TEX. Tento formát není podporován žádným OS — je pouze interním formátem prostředí aplikace. Tento formát bude popsán v podobném rozsahu jako dva výše jmenované, protože jeho znalost umožní pochopit další kapitoly této práce.

1.4.1. Postscriptové fonty

Postscriptové fonty se objevily v tiskárnách LaserWrite spolu s jazykem pro popis stran v roce 1985. Později se v roce 1989 objevila první verze ATM (Adobe Type Manager) pro počítače Macintosh, která uměla rastrovat fonty Adobe Type 1. Od té doby, podle firmy Apple, existuje 30 000 fontů ve formátu Type 1.

Druhy Postscriptových fontů:

Type 0 je kompozitní (složený) font, původně měl sloužit pro OCF (Original Composite Font) fonty, které měly pracovat s velkými znakovými sadami, v současné době tento formát není podporován a OCF fonty nahradily fonty CID.

Type 1 je jednobajtový font pro latinková písma, používá se v postscriptových tiskárnách nebo je rastrován v OS programem ATM; používá rozšířenou podmnožinu jazyka Postscript, která obsahuje hintovací informace.

Type 2 je formát kompaktního kódování charstrings (popis obrysů fontů); formát je určen pro fonty CFF (Compact Font Format), které jsou základem pro Type 1 Open Type a jsou používány v PDF souborech.

Type 3 fonty mohou používat celý postscriptový jazyk, takže mohou oproti Type 1 fontům obsahovat stínování, barvy a vzorky výplní, ale neobsahují žádné informace o hintování; ATM neumí rastrovat tyto fonty, ale lze je zobrazit každým interpretem Postscriptu; mohou obsahovat bitmapové znaky.

Type 4 je zastaralý formát pro trvalé uložení v paměti tiskárny.

Type 5 je podobný jako Type 4, ale uložení do paměti ROM tiskárny.

Type 32 je používaný pro download bitmapových fontů postscriptovými interprety verze 2016 a vyšší; účelem je šetření paměti interpretu.

Type 42 se používá pro načítání TrueType fontů do postscriptové tiskárny, která obsahuje interpret TrueType fontů; tento kód pouze obaluje TrueType font.

Multiple Masters fonty jsou rozšířením formátu Type 1; jde o způsob uspořádání souboru; font obsahuje 2 až 16 tzv. master designs, z kterých lze interpolovat mnoho přechodových fontů; např. je definován ultrabold a ultralight font a ostatní řezy jsou získávány jejich interpolací.

CID-Keyed fonty jsou určeny pro jazyky s velkou znakovou sadou; jde pouze o způsob uspořádání souboru, znaky jsou popsány jako Type 1 nebo Type 2.

CFF fonty využívají formát kódování charstrings Type 2 a jsou základem pro OpenType formát; podobně jako CID fonty jde pouze o způsob uspořádání souboru.

OpenTypeFontFormat je vyvíjen společně firmou Adobe a Microsoft; formát má podporovat vyšší typografické požadavky uživatelů.

1.4.2. TrueType fonty

TrueType fonty jsou společným dílem firem Apple a Mirosoft, poprvé se objevily v roce 1991 v systémech Mac a v roce 1992 v systému Windows 3.1. Vzhledem k rozšířenosti systémů Windows, které formát podporují, můžeme najít tyto fonty ve většině počítačů. První fonty byly TimesNewRoman, Arial a Courier od firmy Monotype. Bohužel není využíván potenciál těchto fontů vývojáři ani programy. Velké množství existujících fontů bývá označováno za typograficky nepodařené a často chybí kerningové informace.

Formát TrueType už není příliš používán a byl rozšířen na TrueTypeOpen, který sdružuje v jenom souboru znaky pro mnoho abeced a kódování (ve Windows FontDialogu volba Script).

1.4.3. Hlavní rozdíly mezi Type 1 a TrueType

1. Matematický popis křivek

TrueType fonty používají pro vyjádření křivek kvadratické B-spliny, které jsou podmnožinou kubických Bézierových křivek v Postscriptu. Proto konverze postscriptového fontu do TrueType fontu obsahuje více zaokrouhlovacích chyb, než obráceně.

2. Hintovací informace

Základní výhoda TrueType fontů je jeho možnostech hintování, což jsou speciální instrukce pro rastrování do malého (řídkého) rastru; tato situace nastává i při rastrování velmi malého písmene do hustého rastru. Postscript dovoluje *řídit* horizontální a vertikální tahy, přesahy, mělké křivky a další, zatímco TrueType umožňuje hintování i diagonální a přesun určitých bodů podle zvolené velikosti písma.

Rozdíl je i v inteligenci intepretů. postscriptový font sdělí interpretu, jaké vlastnosti budou kontrolované, a interpret vlastní inteligencí rozhodne, jak

bude hintování provedeno. Proto s každým zlepšením interpretu může být zlepšeno i hintování. Naopak TrueType obsahuje velice přesné informace o hintování,takže záleží pouze na tvůrcích fontu, jaké úsilí věnují specifikaci hintů pro všechny možné situace a jak využijí větších možností TrueType hintů.

3. Fyzický obraz fontu

Postscriptové fonty jsou uloženy ve dvou souborech — jeden obsahuje kresby znaků a druhý metrické a kerningové informace. TrueType font je celý uložen v jednom souboru. Postscriptové fonty mohou mít ještě informační soubor, který obsahuje základní informace o fontu, je textový a obsahuje pouze několik řádek. Pro všechny registrované postscriptové fonty musí existovat metrický soubor AFM, který je nezávislý na platformě. Obsahuje metrické a kerningové údaje; soubor je textový. TrueType fonty bývají větší, než postscriptové.

4. Podpora v systému

TrueType fonty jsou podporovány systémy Windows a Mac OS, postscriptové fonty OS2 a některými Unixovými implementacemi. Chceme-li používat postscriptové fonty v systémech, které nepodporují tento formát, musíme vlastnit program ATM (Adobe Type Manager), který je interpretem fontů a spolupracuje se systémem. ATM je šířen společně s písmy od firmy Adobe.

5. Množství a kvalita

Postscriptové fonty jsou starší než TrueType. Za dobu existence tohoto formátu bylo vytvořeno mnoho kvalitních a odzkoušených písem, které vyhovují nejen zobrazování na obrazovce počítače, ale i typografickým potřebám. Po vzniku formátu TrueType bylo mnoho postscriptových fontů algoritmicky převáděno a levně šířeno. Tyto fonty nedosahují kvality svých vzorů. Uvážíme-li možnosti formátů, tak lze vytvořit mnohem propracovanější TrueType font, ale za cenu většího úsilí.

6. Čitelnost kódu

TrueType fonty jsou méně čitelné a přehledné než postscriptové. Po rozkrytí postscriptového fontu „může“ odborník přímo vidět, o jaké písmo jde. TrueType fonty hýří offsetovými odkazy a flagy, zatímco postscriptové se podobají kódu napsaném v nějakém programovacím jazyku, což také jsou.

2. Postscriptové fonty

2.1. Adobe Type 1

Type 1 soubor se skládá z ASCII textu a zakódované části, dále může obsahovat speciální informace o *hintování* — což je nápověda pro ovladače, jak zobrazovat znaky při malém nebo velkém rozlišení.

Postscriptové interprety od roku 1984 procházejí neustálým vývojem. Během této doby Adobe Systems upravoval jak postscriptový jazyk, tak části Type 1 font formátu, ale obecná kompatibilita všech verzí byla zachována.

Uspořádání Type 1

Font napsaný v postscriptovém jazyku je uspořádaným souborem procedur popisující tvary znaků. Části tohoto souboru jsou zpřístupňovány kódy znaků a operátorem show jak popisuje *PostScript Language Reference Manual.* Různé fonty obsahují různé množství různých informací. Tyto informace jsou soustředěny ve **slovníku (dictionary).** Slovník obsahuje povinné a nepovinné záznamy.

2.1.1. Stavba znaků

Každý *Type 3* (user defined) font požaduje položku ve slovníku nazvanou BuildChar. Hodnota spojená s tímto jménem je procedura, která je volána postscriptovým interpretem kdykoli potřebuje sestavit znak. V Type 3 BuildChar lze použít každou metodu postscriptového interpretu na vykreslení znaku.

Oproti tomu Type 1 implicitně odkazuje na speciální BuildChar proceduru nazvanou Type 1 BuildChar, která je interní pro postscriptový interpret. Tudíž neexistuje žádná explicitní položka pojmenovaná BuildChar v Type 1 dictionary, protože Type 1 program předpokládá, že bude používat Type 1 BuildChar. Vysvětlením Type 1 font formátu je v podstatě vysvětlení funkce Type 1 BuildChar.

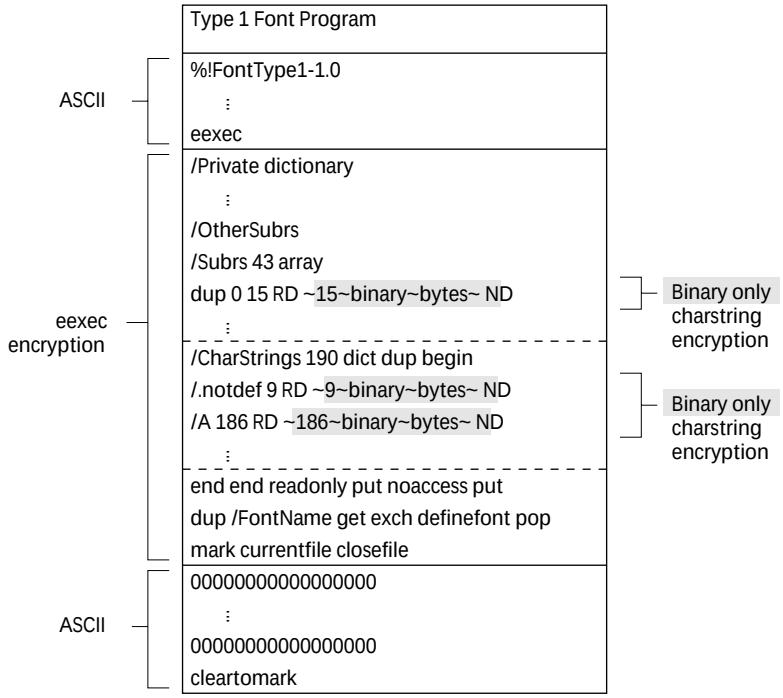
Type 1 BuildChar pracuje se znakovým kódem jako indexem v Encoding vektoru obsaženým ve slovníku fontu (font dictionary), tak získá jméno vykreslovaného znaku. Tento kódovací vektor lze uživatelsky měnit — dochází tedy k překódování (re-encoding) fontu bez žádných jiných změn. Podle tohoto jména, jako klíče, si Type 1 BuildChar vyžádá z CharStrings slovníku (obsažený ve slovníku fontu) binární řetězec, který je zakódovanou postscriptovou reprezentací obrysu znaku. K vykreslení znaku procedura volá speciální verzi stroke nebo fill, záleží na hodnotě PaintType ve slovníku fontu.

Formát Type 1 byl navržen tak, že předpokládá bezchybnost kódu — proto také neobsahuje žádné chybové zprávy.

2.1.2. Slovník fontu

Vytvoření fontu Type 1 znamená vytvoření speciálního slovníku fontu. Tento slovník je tvořen jako každý jiný slovník v jazyce PostScript, ale přidává ještě slovníky CharStrings a Private, které vyžaduje každý Type 1 formát fontu.

Obrázek 2: *Uspořádání Type 1*



2.1.3. Typický soubor Type 1

```
%!FontType1-1.0: Symbol 001.003
%%CreationDate: Thu Apr 16 1987
%%VMusage: 27647 34029
% Copyright (c) 1985, 1987 Adobe Systems
% Incorporated. All rights reserved.
11 dict begin
/FontInfo 8 dict dup begin
/version (001.003) readonly def
/FullName (Symbol) readonly def
/FamilyName (Symbol) readonly def
/Weight (Medium) readonly def
/ItalicAngle 0 def
/isFixedPitch false def
/UnderlinePosition -98 def
/UnderlineThickness 54 def
end readonly def
/FontName /Symbol def
/PaintType 0 def
/FontType 1 def
/FontMatrix [0.001 0 0 0.001 0 0] readonly def
/Encoding 256 array
0 1 255 {1 index exch /.notdef put } for
dup 32 /space put
...
... definice Encoding vektoru
...
dup 254 /bracerightbt put
readonly def
```

```
/FontBBox {-180 -293 1090 1010} readonly def
/UniqueID 6859 def
currentdict end
currentfile eexec
05f3acf73b42a65ec11a12df4c6e26
5306f37b5075f007986cdacc4cd13a
49703465ba20c83c12707f179c0586
3d27adc72767ec06a47e733401fa8d
... zakryptovaná část eexec
0000000000000000000000000000000
0000000000000000000000000000000
...
0000000000000000000000000000000
0000000000000000000000000000000
cleartomark
```

Každý font Type 1 musí začínat komentářem %!, aby bylo snadné identifikovat, že jde o postscriptový soubor. První řádek by měl mít následující syntaxi:

%!FontType1-*SpecVersion: FontName FontVersion*

Číslo *SpecVersion* je verzí Adobe Type 1 formátu. *FontName* je jméno fontu, kterému bude rozumět postscriptový interpret a *FontVersion* je číslo verze daného fontu. Některé aplikace také hledají identifikaci souboru ve následující formě:

%!PS-AdobeFont-1.0: *FontName version*

%%VMusage — zde jsou informace pro postscriptový interpret o paměťových nárocích fontu, pokud bude zaveden do paměti. Pomocí postscriptového operátoru vmstatus můžeme zjistit, jak vypadá paměťový prostor RIPu před zavedením fontu, po jeho zavedení a po opakovaném zavedení. První číslo odpovídá rozdílu výstupu vmstatus před a po zavedení fontu. Druhé číslo pak rozdílu před zavedením a po opakovaném zavedení fontu.

Po komentářích program vymezí slovník s jedenácti prvky. Tento slovník bude slovníkem fontu. Vloží do něj osm položek FontInfo, FontName, PaintType, FontType, FontMatrix, Encoding, FontBox a UniqueID.

FontMatrix je transformační matice, která upraví měřítko na 1000:1. Tento údaj se doporučuje neměnit. Důležitý údaj je také hodnota FontBBox, její přesnost by měla být co největší. Jde o pomyslný obdélník, který bychom získali naskládáním všech znaků fontu na sebe. Postscriptový interpret používá tuto informaci k řízení cashování a ořezávání. Používá-li font k vytváření akcentovaných znaků příkaz seac, měla by být tato hodnota co nejpřesnější. Jestliže font nepoužívá seac, může se hodnota skládat z nul.

FontType musí být stejná ve všech Type 1 fontech. UniqueID je identifikací fontu a pomáhá postscriptovému interpretu cashovat bitmapy znaků, které už byly jednou vytvořeny. Encoding vektor přiřazuje každé z 256 pozic jméno znaku. Hodnota kódu je dána pořadím.

Za textovou částí fontu následuje zakryptovaná část eexec, kterou ukončí 512 nul. Operátor mark v zakryptované části pokládá značku v zásobníku a operátor cleartomark maže do pozice značky a nadbytečné nuly.

2.1.4. Kódovaná část

V zakódované části fontu jsou slovníky CharStrings a Private. Slovník CharStrings obsahuje kódované příkazy na vykreslení obrysů znaků, Private slovník hintovací informace a podprogramy (subroutines). Hinty ve slovníku Private se vztahují k celému fontu. Private dictionary může obsahovat různé postscriptové procedury, které mohou modifikovat chování fontu na některých verzích postscriptového interpretu.

Znakové řetězce v CharStrings musí být dekódovány; tento úkol je interní částí Type 1 BuilChar. Kódované řetězce obrysů znaků se nazývají *charstrings.* Po dekódování je každý charstring podobný postscriptovému programu, kde je užito postfixové syntaxe — operandy předcházejí operátor a operandy jsou ukládány do paměťového zásobníku. Množina příkazů vložených do charstrings je speciální pro Type 1 BuilChar a její operandy jsou typově a rozsahově omezené. Zásobník operandů pro charstring operace je oddělen od obecného postscriptového zásobníku operandů. Některé příkazy jsou podobné vestavěným operátorům Postscriptu, jiné příkazy, jako například hintovací informace pro algoritmus zobrazení znaků, jsou specifické pro Type 1 BuildChar.

V Charstrings musí být definován znak *.notdef* , i když není definován v encoding vektoru. Po odkryptování eexec části můžeme vidět podobný kód:

```
dup /Private 8 dict dup begin
/RD {string currentfile exch readstring pop} executeonly def
/ND {noaccess def} executeonly def
/NP {noaccess put} executeonly def
/BlueValues [-17 0 487 500 673 685] def
/MinFeature {16 16} def
/password 5839 def
/UniqueID 6859 def
/Subrs 43 array
dup 0 15 RD ˜15˜binary˜bytes˜ NP
...
... 41 definic podprogramů vynecháno
...
dup 42 23 RD ˜23˜binary˜bytes˜ NP
ND
2 index /CharStrings 190 dict dup begin
/Alpha 186 RD ˜186˜binary˜bytes˜ ND
...
... 188 character definitions omitted
...
/.notdef 9 RD ˜9˜binary˜bytes˜ ND
end
end
readonly put
noaccess put
dup /FontName get exch definefont pop
mark currentfile closefile
```

Do slovníku fontu jsou přidány další dvě položky (Private a CharStrings). V příkladě kód fontu Private zahrnuje osm položek: RD, ND, NP, BlueValues, MinFeature, password, UniqueID a Subrs. CharStrings sdružuje 190 položek — každá

položka je spojena se jménem znaku (např. Alpha) a zakódovaným *charstring*. Type 1 Buildchar interpretuje charsting pouze tehdy, když je daný znak zobrazován poprvé. Položka Subrs v Private slovníku obsahuje část *charsting*, kterou mohou volat podprogramy různých znaků. *Charstrings* v běžném Type 1 fontu používá RD, ND a NP postscriptové procedury, které jsou definovány především pro zkrácení kódu fontu. Za RD je jedna mezera a sekvence binarních bytů, které jsou obsahem charstrings.

Slovník Private a všechny charstrings jsou označeny atributem noaccess — uživatel postscriptového interpretu nemůže číst nebo zapisovat jejich obsah. Interpret k těmto položkám samozřejmě přistupovat může. Operátor definefont vytvoří první slovník ve slovníku fontu a přidá položku FID.

Poslední sekvence currentfile closefile ukončí činnost eexec, pak se automaticky provede operace end, která odstraní systemdict.

2.1.5. Unique Identification Numbers a jména fontů

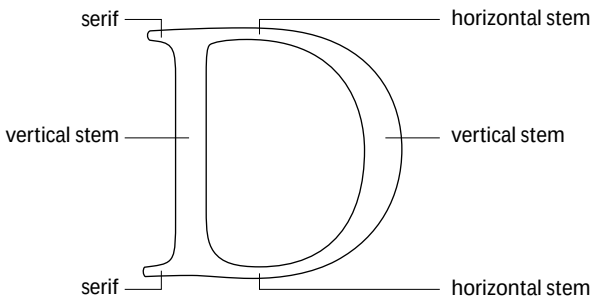
UniqueID je volitelná položka, která pomáhá identifikovat daný font. Původní smysl je jedinečná identifikace již vytvořených bitmap a cashování fontu. Položka UniqueID je definována v hlavním slovníku fontu a v Private slovníku. Jestliže se budou tato čísla lišit, bude se s fontem zacházet jako by neměl žádné UniqueID — cashování bude možné pouze pro jeden úkol.

UniqueID je celočíselný typ v rozmezí od 0 do 16 777 215 ($2^{24} - 1$). Čísla od 4 000 000 do 4 999 999 jsou pro volné použití (open range) a znamená to, že fonty nejsou u Adobe Systems registrovány.

2.1.6. Popis znaku

Znak je popsán vertikálními a horizontálními tahy (vertical, horizontal stems). Tahy mohou být rovné nebo zakřivené, zakončené serify neboli patkami. To platí však pouze pro latinku.

Obrázek 3: *Horizontální a vertikální tahy (stems), patky (serifs)*



Horizontální míry a rozměry pro definování znaku:

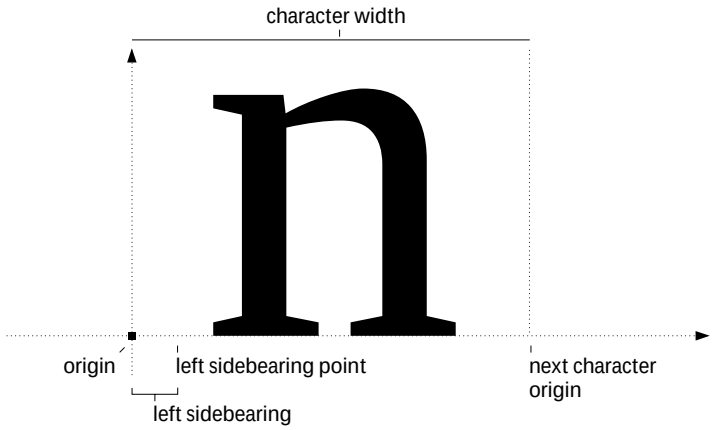
origin znaku je počáteční referenční bod, kryje se s bodem, kde bude znak zobrazen.

width znaku je vektor, obecně od počátku horizontálně doprava, tam by měl být referenční bod pro následující znak.

left sidebearing je horizontální vektor od počátku do souřadnice, kde začíná výplň znaku.

left sidebearing point je souřadnicí, která ukončuje left sidebearing vektor.

Obrázek 4: *Počátek (origin), délka znaku (width), sidebearing*



2.1.7. Alignments and Overshoots (Vyrovnání a přesahy)

Hintování Type 1 používá mnoho vertikálních rozměrů, které se aplikují na celý font. Některé napomáhají k přesné reprezentaci jemných rozdílů ve vyrovnání mezi rovnými a zaoblenými znaky. V terminologii Type 1 se zaobleným místům znaků říká *overshoots (přesahy)*.

Type 1 BuildChar přijímá alignment a overshoot informace ve dvojici čísel. Jedno číslo udává *flat position*, neboli osu *y*, na kterou dosahují rovné znaky; druhé číslo je *overshoot position* — osa *y*, kam přesahují zaoblená písmena. Dvojice čísel se nazývá *alignment zone (vyrovnávací pásmo)*, rozdíl mezi čísly *alignment zone height (výška vyrovnávacího pásma)*; její velikost se typicky pohybuje mezi 10 a 20 jednotkami.

Všechny osy jsou popsány v jednotkách *character space* a předpokládá se poměr 1000 : 1 character space k user space. Pro každý znak fontu existuje jedna vhodná *alignment zone*.

baseline — *y* osa, typograficky nazývaná účaří, typicky má hodnotu nula.

baseline overshoot position — *y* osa, nachází se pod baseline (účařím), dosahují na ni zaoblené dolní části znaků. Typicky má hodnotu – 15. Vzhledem k tomu, že je pod účařími má zápornou hodnotu.

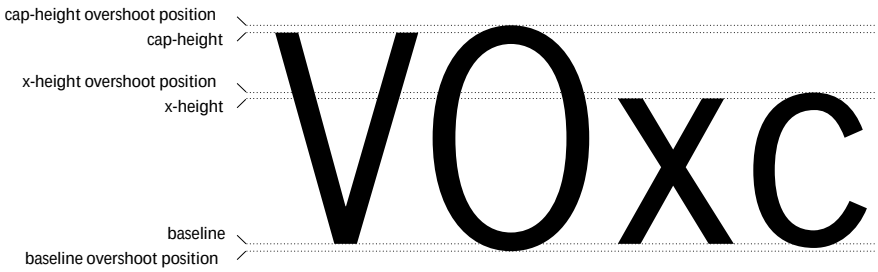
cap-height — *y* osa vrcholu rovných velkých písmen, většinou má hodnotu 700.

cap-height overshoot position — nejvyšší *y* osa, nachází se nad *cap-height*, dosahují na ni vrcholy zaoblených velkých písmen. Typická hodnota je o 10 až 20 jednotek větší než *cap-height*.

x-height — *y* osa kam dosahují malá neakcentovaná rovná písmena. Typická hodnota je blízká 450.

x-height overshoot position — nejvyšší *y* osa, kam dosahují malá neakcentovaná zaoblená písmena, její hodnota se pohybuje 10 až 20 jednotek nad *x-height*

Obrázek 5: *Baseline, x-height, cap-height a jejich přesahové osy*



Alignment zones vrcholů znaků se nazývají *top-zones (horní pásma)* a pásma pro dolní části znaků pak *bottom-zones (spodní pásma)*.

Téměř všechny latinkové fonty Type 1 používají baseline, cap-height a x-height alignment zones. Některé fonty přidávají další zones podle potřeby. Mohou popisovat figure-height, ascender-height, descender-depth, superior baseline, ordinal baseline (výšku znaku, horní a dolní dotažnici, horní účaří, řádné účaří) a další. Dílčí množina různých pásem se vybírá podle kresby daného písma; není však nutné používat žádná další pásma.

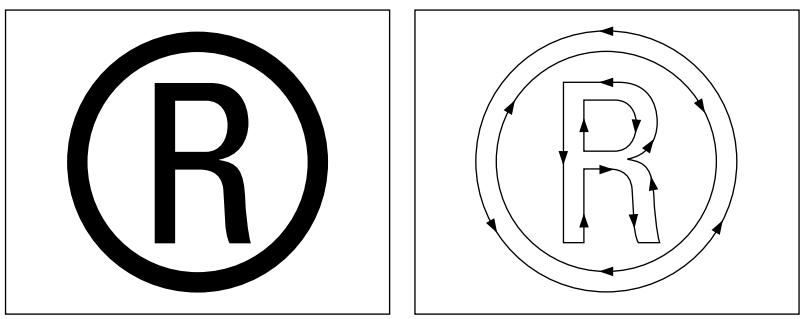
2.1.8. Character Paths (Cesty znaku)

Chceme-li v Postscriptu definovat znak, musíme popsat jeho obrys v character space. Obrys je popsán cestou (path) operátory moveto, lineto, curveto, closepath, rlineto atd. Znaky musí být definovány jako obrysy, operátory image a imagemask nejsou povoleny.

Mnoho verzí postscriptových interpretů má vnitřní limit 1500 položek path na jeden znak, po překročení této meze následuje limitcheck error. Nechceme-li použít Type 3, kde není tento limit, musíme redukovat počet složek path.

Dicrection of Paths (směry cest) Podcesty (subpaths), které mají mít výplň, musí být definovány levotočivě (v protisměru hodinových ručiček), subpaths nevyplněné pak pravotočivě.

Obrázek 6: *Tvorba cest ve správném směru*



Overlapping Paths (překrývání cest) Jednotlivé uzavřené obrysy by se neměly protínat. Budou-li dvě vyplněné podcesty protínat, může nastat problém při vyplňování znaku. Nastaví-li se PaintType na hodnotu 2, dojde k vykreslení obrysů znaku — v tomto případě budou protínající se obrysy viditelné.

2.1.9. Technika kresby

Zpočátku se může zdát, že obrysy znaku nepotřebují odlišnou grafickou techniku než jiné kresby, ale pokud mají znaky vypadat opravdu dobře, je nutné přijmout určité pravidla. Opomenutí těchto pravidel může mít za následek nerovné tahy, nechtěné body, špatné tvary křivek a nerovné přechody z rovných ploch na zakřivené.

Krajní body v extrémech Koncový bod (první nebo poslední lineto nebo curveto) by měl být umístěn v nejvyšším horizontálním nebo vertikálním extrému. To znamená, že křivky by neměly točit více než o 90°. Umístění v extrémech pomáhá zobrazovacímu algoritmu přesně reprodukovat důležité části znaků. Toto není nutné dodržovat u krátkých křivek, jako např. u zakřivených serifů.

Spojitost tečny Hladké navazování křivek zlepšuje reprezentaci znaků, pokud používáme rastrovací zařízení. Spojitost tečny (tangent continuity) popisuje metodu hladkého přechodu křivek. Směr první křivky v koncovém bodě by měl být stejný jako směr navazující křivky v jejím počátečním bodě.

Stručnost Obrys znaku by měl být definován co nejstručněji při dodržení ostatních pravidel. Dosáhneme úspory ve využití paměti, zrychlení vykreslovacího algoritmu a zjednodušení hintování.

- používat co nejmenší počet Bézierových křivek
- nepoužívat po sobě jdoucí rovné tahy na stejné přímce
- nekreslit rovné tahy pomocí curveto
- když je to možné, používat closepath k vykreslení rovného úseku
- nalézt nejmenší počet částí, ze kterých se bude skládat znak

Konzistence Protože fonty Type 1 se používají na velkém počtu různých výstupních zařízení, může docházet k různým chybám. Proto je důležité dodržovat:

- všechny tahy, které mají být stejné, by měly být opravdu stejné,
- všechny znaky, které mají mít stejnou výšku, by měly být zarovnány na stejnou osu,
- všechny další charakteristiky (sidebearing), které mají být stejné, by měly být opravdu stejné.

2.2. Adobe Font Metrics File Format

Tento formát metrických informací postscriptových fontů je zapsán jako holý ASCII text, který je srozumitelný nejen počítači, ale i člověku. Je nezávislý na platformě a rozšiřitelný. Formát je znám jako Adobe Font Metrics (AFM), Adobe Multiple Font Metrics (AMFM) a Adobe Composite Font Metrics (ACFM).

AFM soubor zahrnuje metrické informace pro font jako celek, tak i zvlášť pro každý znak. AFM formát je určen pro základní Type 1 fonty, stejně jako pro

složité fonty asijských jazyků a multiple master fonty, které jsou rozšířením Type 1 fontů.

Metriky multiple master fontů popisuje jeden AMFM soubor, který obsahuje řídící data a obecné informace o fontu, a jeden AFM soubor pro každý řez písma.

2.2.1. Struktura AFM souboru

AFM soubor se skládá z několika částí: Control Information (řídící informace), Global Font Information (celkové údaje o fontu), Writing Direction Information (informace o směru psaní), Individual Character Metrics (metriky jednotlivých znaků), Kerning Data (kerningové informace), Composite Character Data (údaje o kompozitních znacích).

AFM soubor je ohraničený klíčovými slovy:

StartFontMetrics *version*
EndFontMetrics

Tato klíčová slova jsou povinná a ohraničují celý AFM soubor (musí být na první a poslední neprázdné řádce); *version* je číslo verze specifikace AFM formátu.

2.2.2. Global Font Information

Údaje uvedené v této sekci platí pro všechny znaky fontu. V případě, že jde o kompozitní font, jsou aplikovány na všechny nižší fonty (dílčí řezy) a na všechny znaky, které obsahují.

FontName *string* jméno fontu, které používá ps. operátor findfont.

FullName *string* celé jméno fontu.

FamilyName *string* rodina písma, ke kterému font patří.

Weight *string* váha písma.

FontBBox *number number number number* definice bounding boxu fontu; levý dolní a pravý horní roh.

Version *string* identifikátor verze fontu, odpovídá řetězci ve slovníku FontInfo.

Notice *string* poznámka pro obchodní značku nebo copyright.

EncodingScheme *string* řetězec označuje implicitní encoding vektor fontu.

MappingScheme *integer* není použito v základních fontech.

EscChar povinný parametr pokud MappingScheme = 3, jinak není použit.

CharacterSet *string* řetězec popisující množinu znaků fontu.

Characters *integer* počet znaků fontu.

IsBaseFont *boolean* má hodnotu *true*, jde-li o základní font, *false* při ACFM.

IsCIDFont *boolean* vyžadováno u fontů CID-keyed.

VVector *number number* povinný, je-li MetricsSets 2; složky vektoru od počátku 0 (počátek pro směr psaní 0) do počátku 1 (počátek pro směr psaní 1); je-li definován, pak VVector je pro všechny znaky stejný (a IsFixedV má hodnotu *true*).

IsFixedV *boolean* je-li *true*, pak je VVector je pro všechny znaky stejný.

CapHeight *number* obvykle y-hodnota vrcholu písmene *H*.

XHeight *number* obvykle y-hodnota vrcholu písmene *x*.

Ascender *number* pro latinkové fonty, obvykle y-hodnota písmene *d*.

Descender *number* pro latinkové fonty, obvykle y-hodnota konce písmene *p*.

StdVW *number* udává převládající délku horizontálních hlavních tahů

StdHW *number* udává převládající délku vertikálních hlavních tahů.

2.2.3. Writing Direction Information

Tato část je ohraničena dvojicí klíčových slov:

StartDirection *integer*
EndDirection

Je-li tato část vynechána předpokládá se hodnota StartDirection 0. Některé údaje se mohou objevit i bez vymezení StartDirection a EndDirection:

UnderlinePosition *number* vzdálenost od účaří pro centrované podtrhávací tahy.

UnderlineThickness *number* síla podtrhávací čáry.

ItalicAngle *number* úhel hlavních svislých tahů fontu; pro neitalické fonty 0.

CharWidth *number number* pokud IsFixedPitch má hodnotu true, určuje *x* a *y* složky délkového vektoru (všechny znaky mají stejnou délku pro daný směr psaní).

IsFixedPitch *boolean* identifikace neproporcionálního fontu.

2.2.4. Individual Character Metrics

StartCharMetrics *integer*
EndCharMetrics

Tato část je také volitelná a ohraničena klíčovými slovy. Každá metrika znaku se skládá z posloupnosti klíčů a hodnot na jednom řádku oddělených středníkem. Znaky jsou řazeny sestupně podle číselného kódu znaku. Hodnota *integer* označuje počet znaků. Nekódované znaky následují za kódovanými a mají hodnotu kódu znaku nastavenou na -1.

C *integer* dekadická hodnota kódu znaku, pokud není znak v kódovém vektoru fontu hodnata je -1; jde-li o CID-keyed font, všechny znaky budou mít hodnotu -1, CID number bude určeno podle klíče N.

CH < *hex* > stejné jako C, ale kód znaku je hexadecimální.

WX *number* délka znaku na ose *x* pro writing direction 0; výška na ose *y* je 0;

WY *number* výška znaku na ose *y* pro writing direction 0; délka na ose *x* je 0;

W $number_x$ $number_y$ délkový vektor znaku na obou osách pro writing direct. 0;

VV $number_x$ $number_y$ stejný význam jako VVector v Global Font Program Information, ale pro jediný znak.

N *name* postscriptové jméno znaku nebo CID-keyed number.

B *llx lly urx ury* boundig box znaku, levý dolní a pravý horní roh; pokud znak nemá žádnou kresbu (např. mezera) může být B 0 0 0 0.

L *successor ligature* posloupnost dvou znaků, kde *successor* je jméno znaku, které vytvoří slitek, bude-li následovat za znakem N, tato sekvence bude nahrazena znakem *ligature*.

2.2.5. Kerning Data

StartKernData *integer*
EndKernData

Kerningová data jsou mají dvě formy: track kerning a pair-wise kerning. Obě formy jsou nepovinné, ale mohou se vyskytovat obě najednou. Track kerning se použije pro všechny znaky, zatímco pair-wise kerning má význam pouze pro určitou dvojici znaků.

Track Kerning

Track Kerning může být definovaný pro každý směr psaní, podle hodnoty uvedené MetricsSets

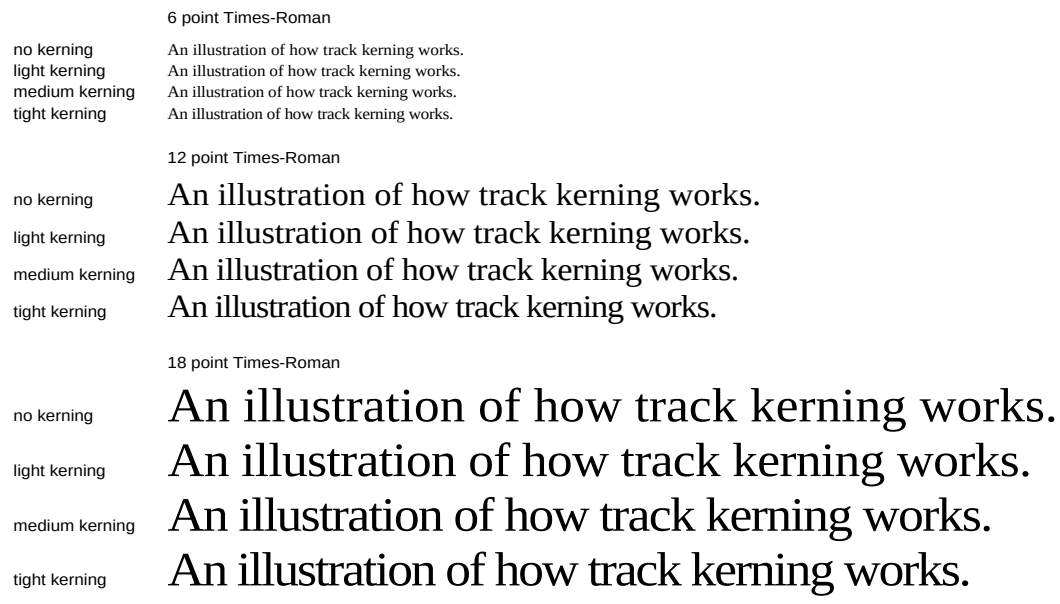
StartTrackKern *integer*
EntTrackKern

Ohraničení výše uvedenými klíčovými slovy je povinné, když je plošný kerning ve fontu přítomen. Hodnota *integer* udává počet rozdílných množin plošného kerningu fontu. Track kerning je stanoven pro různé míry těsnosti (degrees of tightness).

TrackKern *degree min-ptsize min-kern max-size max-kern*

Degree je zvyšující se záporná celočíselná hodnota odpovídající těsnosti track kerningu, pozitivní směr plošný kerning rozvolňuje. Track kerning je lineární funkce. Uvedené hodnoty pro minimální a maximální velikost písma s jejich hodnotou kernu se použijí pro odvozování plošného kernu v jiných velikostech písma.

Obrázek 7: *Stupně plošného (track) kerningu*



Pair-wise Kerning

Párový kerning lze definovat pouze pro dva směry psaní. Hodnota *integer* označuje počet kerningových párů. Každý kerningový pár je definován na samostatné řádce.

StartKernPairs *integer*
EndKernPairs

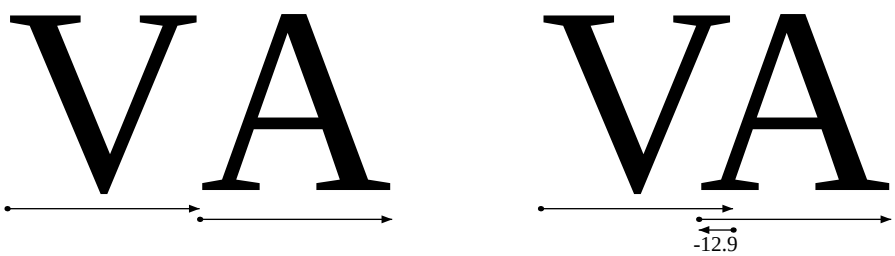
KP $name_1$ $name_2$ $number_x$ $number_y$ pro dva znaky je definovaný kerning vektorem *x, y*, což je vzdálenost $znaku_1$ od $znaku_2$.

KPH < hex_1 > < hex_2 > $number_x$ $number_y$ stejné jako KP, ale znaky jsou určeny jejich hexadecimálním kódem.

KPX $name_1$ $name_2$ $number_x$ kerning pouze ve směru osy *x*

KPY $name_1$ $name_2$ $number_y$ kerning pouze ve směru osy *y*

Obrázek 8: *Párový (pair-wise) kerning*



Characters printed without kerning | Pair-wise kerning applied

2.2.6. Composite Chracter Data

Kompozitní znaky jsou nové znaky, které se skládají ze znaků již existujících (definovaných). Příkladem mohou být akcentované znaky. Počet složených znaků udává hodnota *integer*.

StartComposites *integer*
EndComposites

Data pro každý kompozitní znak jsou definovány klíčovými slovy a jejich hodnotami oddělených středníkem. Každý složený znak je definován na zvláštní řádce.

CC *name integer* jméno kompozitního znaku a počet částí, z kterých se skládá.

PCC *name deltax deltay* definice jedné části kompozitního znaku; název části a jeho posun po osách *deltax, deltay* od počátku.

Příklad 1: *Soubor AFM (ITC Garamond Bold Condensed)*

```
StartFontMetrics 2.0
Comment Generated by Fontographer 3.5 Sat Oct 31 15:16:29 1998
FontName ITC-GaramondBoldCondensed
FullName ITC Garamond Bold Condensed
FamilyName ITC Garamond
Weight Bold
Notice Copyright (c) 1990 Adobe Systems Incorporated.
ItalicAngle 0
```

```
IsFixedPitch false
UnderlinePosition -100
UnderlineThickness 50
Version 1.000
EncodingScheme Windows
FontBBox -167 -236 1000 884
CapHeight 630
XHeight 456
Descender -232
Ascender 700
StartCharMetrics 229
C 5 ; WX 278 ; N breve ; B 0 522 278 685 ;
C 6 ; WX 278 ; N dotaccent ; B 75 532 203 660 ;
...
C 99 ; WX 389 ; N c ; B 16 -16 373 468 ;
C 100 ; WX 500 ; N d ; B 26 -22 502 700 ;
C 104 ; WX 500 ; N h ; B 5 -4 480 700 ;
...
C 126 ; WX 600 ; N asciitilde ; B 90 140 510 312 ;
C 128 ; WX 500 ; N fi ; B 10 -4 482 708 ;
C 129 ; WX 500 ; N fl ; B 10 -4 482 708 ;
C 130 ; WX 222 ; N quotesinglbase ; B 32 -130 174 126 ;
...
C 255 ; WX 389 ; N ydieresis ; B -14 -220 412 660 ;
C 256 ; WX 444 ; N Lslash ; B 4 -4 432 630 ;
C 257 ; WX 278 ; N lslash ; B -2 -4 298 700 ;
EndCharMetrics
StartKernData
StartKernPairs 92
KPX A T -55
KPX A V -74
KPX A W -74
KPX A Y -55
...
KPX quoteright quoteright -37
EndKernPairs
EndKernData
StartComposites 58
CC Scaron 2 ; PCC S 0 0 ; PCC caron 66 150 ;
CC Zcaron 2 ; PCC Z 0 0 ; PCC caron 94 150 ;
...
CC yacute 2 ; PCC y 0 0 ; PCC acute 56 0 ;
CC ydieresis 2 ; PCC y 0 0 ; PCC dieresis 56 0 ;
EndComposites
EndFontMetric
```

3. TrueType fonty

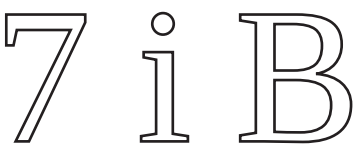
3.1. TrueType

TrueType font je sice mladší než Type 1, ale jeho rozšíření je větší díky tomu, že je součástí operačních systémů Windows.

3.1.1. Obrysy

Znak (glyph) se skládá z obrysů, kontur (contours). Jednoduché znaky mohou být popsány jedním obrysem. Složitější více obrysy a složené kombinací. Existují řídící znaky, které nemají žádné obrysy.

Obrázek 9: *Znaky s jednou, dvěma a třemi konturami*



Kontury se skládají z rovných čar a křivek. Křivky jsou definovány jako řady bodů popsaných druhým stupněm Bézierových splajnů (Bézier-splines). Body tvořící křivku musí být očíslovány pořadově. Vzestupné nebo sestupné pořadí ovlivní vzorek výplně znaku. Tam kde je pořadí rostoucí, tam bude po pravé straně použito černé výplně.

3.1.2. FUnits, em square a souřadnicový systém

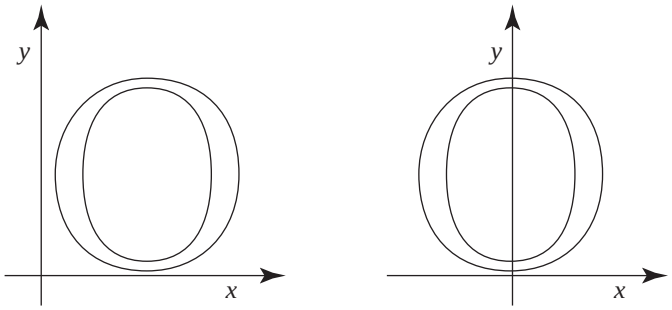
V TrueType fontu je umístění bodů popsáno v jednotkách fontu neboli FUnits. FUnit je nejmenší měřitelnou jednotkou em square — imaginární čtverec, který je používán k popisu velikosti a zarovnání znaků. Em square má typicky rozměry jako výška fontu plus dodatečná mezera pro případ, že by se sázelo bez prokladů.

Souřadnicová síť je dvojrozměrná a počátek má v bodě (0,0). Má omezenou velikost, body se musí pohybovat v rozmezí -16384 a $+16384$ FUnits. Hustota rastru souřadnicového systému se nazývá *units per em (upem)*. Zvětšování bude nejrychlejší, bude-li upem mocninou čísla dvě (např. 2048).

Počátek em square nemusí mít žádný zásadní vztah k obrysu znaku. Ve skutečnosti však existují některé zvyklosti, podle kterých jsou umísťovány znaky do fontu. Pro latinkové (roman) fonty se předpokládá horizontální sazba, a proto pro y osu bude typická hodnota 0 a předpokládá se, že bude totožná s účařím (baseline) fontu. Pro osu x není nutná rovnost nule, ale stardardně se používá také.

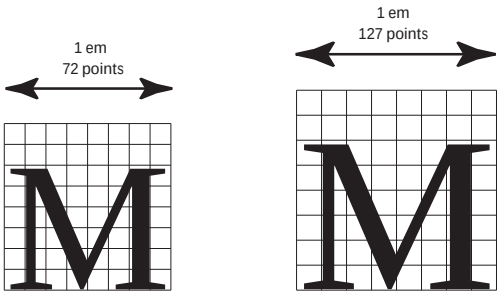
Další možností je umístění každého znaku tak, že jeho krajní levý bod obrysu má x hodnotu stejnou jako left-side-bearing znaku. Fonty vytvořené tímto způsobem mohou urychlit tisk na postscriptových tiskárnách. Na následujícím obrázku je v prvním případě je left-side-bearing nula a v druhém je znak zarovnán na optický střed.

Obrázek 10: *Dva možné způsoby umístění počátku v latinkových fontech*



Hustota rastru em square je dána počtem jednotek (FUnits) na em. Em square má svůj souřadnicový systém, kde jedna jednotka je rovna FUnit. Čím větší počet jednotek na em, tím je větší přesnost adresování uvnitř em square. FUnits jsou relativní jednotky, ale upem jsou konstatní pro daný font bez ohledu na velikost bodu. Em square je přesně 9 bodů vysoký, když je znak zobrazován v devíti bodech, přesně 10 bodů vysoký, když je zobrazován v deseti bodech atd. Zatímco počet upem se nemění s velikostí bodu, v kterém je font zobrazován, absolutní velikost FUnit se mění s touto velikostí.

Obrázek 11: *72 a 127 bodové M v jejich em squares; upem se rovná 8 v obou případech*



Předtím než je znak zobrazen na výstupním zařízení musí byt zvětšen (zmenšen) na požadovanou velikost v absolutních jednotkách — pixel je jeden bod výstupního zařízení. Někdy nás může také zajímat počet pixelů na jedno em (ppem).

ppem = velikost znaku × dpi / 72.

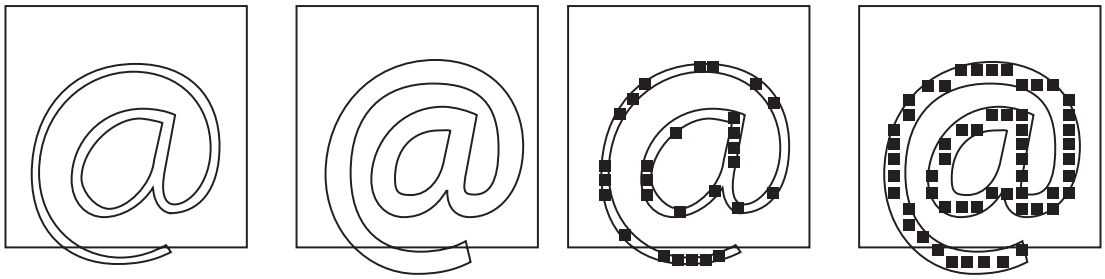
Potom převod mezi FUnits a pixely je:

$$\text{souřadnice pixelu} = \text{souřadnice em} \times \frac{\text{ppem}}{\text{upem}}$$

3.1.3. Grid-fitting (úprava rastru) obrysu

Aby na různých zařízeních byly znaky zobrazovány s co největší věrností ve všech velikostech, existuje speciální instrukce úpravy rastru. Cílem je dosažení stejně silných tahů, barvy, mezer a vyloučení vzniku tzv. *pixels dropouts (zmizelých bodů)*. Tyto instrukce je nutné použít v době rastrování znaku — dochází k změnám nebo deformacím originálního obrysu, aby byla dosažena co nejvyšší kvalita vyobrazení. Tyto deformace obrysu se nazývají *grid-fitting (úprava rastru)*.

Obrázek 12: *Obrys bez grid-fitting (vlevo), s grid-fitting (vpravo) a výsledek rastrování*



Grid-fitting je proces natahování obrysu znaku podle instrukcí s ním spojených. Může se stát, že bude změněn počet bodů obrysu nebo některé body budou posunuty. Množina TrueType instrukcí obsahuje množství příkazů na upřesnění způsobu vykreslení znaku. Můžeme říci, že řídí proces grid-fitting podle toho na jakém zařízení a v jaké velikosti bude znak zobrazen.

TrueType fonty mohou obsahovat instrukce, ale také nemusí. Obecně lze říci, že fonty bez instrukcí dosahují vyšší kvality při dostatečně vysokých rozlišení a velikostech znaků. Záleží ovšem také na dané kresbě písma. Při použití fontů v malých velikostech a nízkých rozlišení jsou dodatečné instrukce rozhodující. Použití instukcí je složitý proces, který mimo jiné zahrunuje analýzu klíčových rysů kresby písma.

3.1.4. TrueType interpret

Interpret zpracovává tok nebo posloupnost instrukcí. Tyto instrukce si berou své argumenty z *instruction stream (tok instrukcí)*. Všechny akce interpretu se přenášejí do kontextu Graphics State (Grafický stav — množina proměnných; její hodnoty doprovázejí činnost interpretu a určují přesný účinek dílčích instrukcí.

Shrnutí činnosti interpretu:

1. Interpret vezme instrukci z instrukčního toku, v pořadí kód instrukce (op-code) a data. Kód instrukce má velikost jednoho bytu, data jednoho bytu nebo dvou bytů (word). Bere-li word, skládá jej ze dvou bytů v pořadí hight byte, low byte.

2. Instrukce je vykonána. Jde-li o instrukci *push*, vezme si své argumenty z instruction stream. Instrukce si vezme data (jestliže je potřebuje) ze zásobníku *pop*. Data, která jsou výsledkem instrukce se vloží (push) do zásobníku interpretu. Efekt provedení záleží na hodnotách proměnných, které tvoří Graphics State. Instrukce může modifikovat proměnné Graphics State.

3. Proces se opakuje dokud nebudou všechny instrukce provedeny.

3.1.5. Používání instrukcí

Instrukce se mohou objevit na mnoha místech v tabulkách fontu, které tvoří TrueType font. Mohou být částí Font Programu, CVT Programu nebo v datech znaku. Mohou se aplikovat na celý font nebo jen na konkrétní znak.

Font Program se skládá z množiny instrukcí, které se vykonají pouze jednou — při výběru fontu aplikací. Vytvoří se *function definitions FDEFs (definice*

funkcí) a *instruction definitions IDEFs (definice instrukcí)*. Funkce a instrukce definované Font Programem mohou být použity kdekoliv v celém fontu.

CVT Program je sekvencí TrueType instrukcí vykonávaných vždy, když se mění velikost písma nebo dochází k jiné transformaci — je výhodnější změnit parametry celého fontu, než jednotlivých znaků. CVT Program inicializuje hodnoty v Control Value Table. CVT je číslovaný seznam hodnot, na který může být odkazováno jednou ze dvou nepřímých instrukcí: MIRP a MIAP. Položky CVT soustřeďují hodnoty, které musí být stejné pro každý znak v celém fontu.

Příklad 2: *Příklad položek CVT*

Entry #	Value	Description
0	0	upper and lower case flat base (base line)
1	-39	upper case round base
2	-35	lower case round base
3	-33	figure round base
4	1082	x-height flat
5	1114	x-height round overlap
6	1493	flat cap
7	1522	round cap
8	1463	numbers flat
9	1491	numbers round top
10	1493	flat ascender
11	1514	round ascender
12	157	x stem weight
13	127	y stem weight
14	57	serif
15	83	space between the dot and the i

Instrukce, které odkazují na hodnoty v CVT jsou volány nepřímo jako protiklad přímým instrukcím, které si berou hodnoty přímo z obrysu znaku. Jako součást TrueType fontu jsou hodnoty CVT vyjádřeny v FUnits, při konverzi obrysu na pixely se stejně převedou i hodnoty CVT. Při psaní CVT lze použít hodnoty, které jsou vyjádřeny v souřadnicovém systému znaku WCVTP nebo použít přímo původní jednotky FUnits WCVTF. Hodnoty se čtou z CTV vždy v pixelech (F26Dot6).

3.1.6. Storage Area

Interpret má také Storage area (ukádací prostor) — část paměti, která se využívá pro dočasnou úschovu dat ze zásobníku interpretu. Existují instrukce pro čtení těchto uložených dat a ukládání nových. Rozsah Storage area je od 0 do n – 1, kde n je hodnota položky maxStorage v maxProfile table. Hodnoty jsou 32 bitová čísla.

3.1.7. Graphics State

Graphics State (grafický stav) je tabulkou proměnných a hodnot. Tato tabulka má standardní hodnoty, které se mohou modifikovat příslušnými instrukcemi. Grafický stav nemá svou vlastní paměť, proto si uchovává informace pouze pro jeden znak.

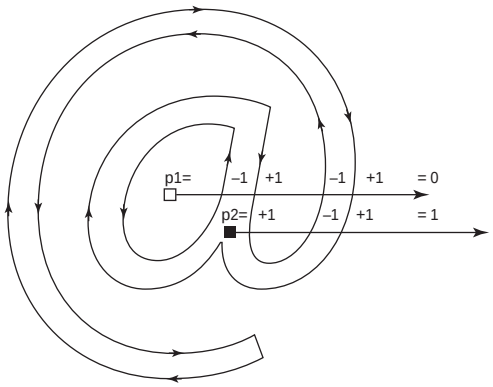
3.1.8. Scan Converter

TrueType Scan converter (obrazový převodník) převádí popis obrysu znaku na bitmapový obraz. Scan converter pracuje ve dvou režimech: v prvním jednoduchý algoritmus určuje body, které jsou částí znaku. Používá následující pravidla:

- jestliže střed pixelu spadá do obrysu, pixel bude částí znaku
- jestliže obrys prochází středem pixelu, pixel nebude částí znaku

V druhém je bod brán v úvahu, jestliže má nenulové winding number. Winding number se určí podle tahu přímky vedené z daného bodu do nekonečna (směr je nepodstatný). Na začátku tahu je Winding number rovno nule, odečteme jedna, když obrys protne přímku zprava doleva nebo zezdola nahoru *(on transition)*, přičteme jedna, když obrys protne přímky zleva doprava nebo shora dolů *(off transition)*. Je-li výsledné číslo nenulové, bod je vnitřním bodem a bude zobrazen.

Obrázek 13: *Výpočet Winding number pro body p1 a p2*



3.1.9. Dropouts

Dropouts (ztracené body) vznikají, když spojení dvou bodů uvnitř znaku není možné rovnou čarou. TrueType instrukce umožňují takovou úpravu rastru znaku, že označené pixely budou rozsvíceny bez ohledu na velikost nebo transformaci znaku, ale je těžké předvídat všechny transformace, kterými může znak projít. Tento problém nastává zejména při malém počtu pixelů na em a u komplikovaných písem. K prevenci vzniku dropouts, lze použít pro scan converter další dvě pravidla:

- jestliže scan přímka mezi sousedními pixely prochází jejich středem (vertikálně nebo horizontálně), je protnuta *on transition* i *off transition* obrysem a ani jeden z bodů nebyl rozsvícen předchozími pravidly, bude rozsvícen bod více vlevo (horizontální scan přímka) nebo bod více dole (vertikální scan přímka).
- použije se předchozí pravidlo, jestliže dvě kontury protnou další scan přímky v obou směrech. Tento bod nebude vysvícen, protože jde o tzv. „stubs“ (pahýly, zuby). Výseče scan přímky, které svírají pravý úhel s výsečí protnuté scan přímky, jsou prohlédnuty, neprotínají-li ji dvě kontury. Je možné, že jde o dva různé obrysy.

Tvůrci fontů si mohou vybrat, jaká pravidla aplikují na svoje písmo. Lze určit různá pravidla pro každý znak zvlášť.

3.1.10. TrueType soubor

Soubor TrueType je soubor dat v tabulkové formě; tabulky popisují obrysy znaků. Pro rastrování se používá kombinace dat z různých tabulek na vykreslení znaku. Snadnější průchod tabulkami nastane při zarovnání tabulek tak, že každá začíná na čtyřbytové hranici. Tabulky lze zarovnat nulami.

TrueType font definuje jedenáct datových typů. Jsou to: byte, char, ushort, short, ulong, long, fixed, funit, fword, ufword, f2dot14. Začíná-li typ písmenem *u* jde o čísla bez znaménka. Typ funit je nejmenší měřitelná velikost v em. fword je 16 bitové celočíselné číslo pro popis jednotek v FUnits. F2dot14 je 16 bitové číslo s pevnými 14 desetinnými čísly.

Table Directory

TrueType soubor začíná na bytu 0 Offset Table.

Jméno	hodnota	popis
sfnt version:	1.0	číslo verze fontu
numTables	16	počet tabulek
searchRange	256	maximální mocnina dvou $<$ numTables $\times$ 16
entrySelector	4	$\log_2$(max. mocnina dvou $<$ numTables)
rangeShift	0	numTables $\times$ 16 $-$ searchRange

Tabulka 1: *Offsetová tabulka TrueType fontu*

Na bytu 12 začínají položky Table Directory. Položky v Table Directory musí být řazeny vzestupně podle tagů. Obsahem jedné položky jsou: tag 4 bytový identifikátor, checkSum kontrolní součet, offset offset od začátku souboru, length délka tabulky.

Příklad 3: *Příklad Directory table*

```
 0. 'LTSH' - chksm = 0x3810FCDD, off = 0x0000010C, len =    250
 1. 'OS/2' - chksm = 0x74AF571A, off = 0x00000208, len =    86
 2. 'cmap' - chksm = 0x420370CE, off = 0x00000260, len =    808
 3. 'cvt ' - chksm = 0x7DD555E5, off = 0x00000588, len =   1094
 4. 'fpgm' - chksm = 0xF5775FD3, off = 0x000009D0, len =   1084
 5. 'glyf' - chksm = 0x3A78FF96, off = 0x00000E0C, len =  62148
 6. 'hdmx' - chksm = 0x24196C4E, off = 0x000100D0, len =   7696
 7. 'head' - chksm = 0xB7F81A71, off = 0x00011EE0, len =   54
 8. 'hhea' - chksm = 0x0DD80602, off = 0x00011F18, len =   36
 9. 'hmtx' - chksm = 0xEABD3F01, off = 0x00011F3C, len =   984
10. 'kern' - chksm = 0x58C35F3B, off = 0x00012314, len =   3006
11. 'loca' - chksm = 0x629B2816, off = 0x00012ED4, len =   494
12. 'maxp' - chksm = 0x04B00810, off = 0x000130C4, len =   32
13. 'name' - chksm = 0x22DF71E9, off = 0x000130E4, len =   1320
14. 'post' - chksm = 0xC6D3E233, off = 0x0001360C, len =   555
15. 'prep' - chksm = 0x2A3279BC, off = 0x00013838, len =   1652
```

Table Directory obsahuje pouze ty tabulky, které daný font skutečně potřebuje. Důsledkem je, že neexistuje standardní hodnota numTables. Tagy jsou jména

tabulek TrueType fontu. Všechny jména tagů skládají ze čtyř znaků, ačkoli to není nutné. Jména s menším počtem znaků musí být doplněny mezerami.

Tag	jméno
cmap	character to glyph mapping
glyf	glyph data
head	font header
hhea	horizontal header
hmtx	horizontal metrics
loca	index to location
maxp	maximum profile
name	naming table
post	PostScript information
OS/2	OS/2 and Windows specific metrics

Tabulka 2: *Povinné tabulky TrueType fontu*

Tag	jméno
cvt	Control Value Table
EBDT	Embedded bitmap data
EBLC	Embedded bitmap location data
EBSC	Embedded bitmap scaling data
fpgm	font program
gasp	grid-fitting and scan conversion procedure (grayscale)
hdmx	horizontal device metrics
kern	kerning
LTSH	Linear threshold table
prep	CVT Program
PCLT	PCL5
VDMX	Vertical Device Metrics table
vhea	Vertical Metrics header
vmtx	Vertical Metrics

Tabulka 3: *Nepovinné tabulky TrueType fontu*

Další tabulky mohou být definovány pro jiné platformy a pro budoucí rozšíření fontu. Tyto tabulky nebudou mít žádný efekt na scan converter. Tagy těchto tabulek musí být registrovány u *Apple Developer Technical Support.* Jména tagů, která jsou složena z velkých písmen jsou rezervována pro Apple. Číslo 0 není platné jméno tagu.

3.1.11. cmap (Character To Glyph Index Mapping Table)

Tato tabulka definuje mapování kódů znaků na indexy obrysů (glyph index) používané v daném fontu. Může obsahovat více než jednu podtabulku, aby bylo možné používat více znakových kódovacích schémat (encoding scheme). Kódy znaků, které neodpovídají žádnému obrysu daného fontu, by měly být mapovány s indexem nula. Obrys na této pozici musí být speciální obrys zastupující chybějící znak.

Záhlaví tabulky udává, které kódování je aktuální. Každá podtabulka je v jednom ze čtyř možných formátů a začíná kódem formátu, který je použit. Platform ID a platform-specific encoding ID jsou užívány na upřesnění podtabulky; to znamená, že každý pár Platform ID / Platform-specific encoding ID se může jednou objevit v tabulce cmap. Každá tabulka může specifikovat rozdílné kódování znaků. Položky musí být řazeny nejdříve podle platform ID a potom podle platform-specific encoding ID.

Když píšeme *Unicode font* pro Windows, Platform ID by mělo být tři a Encoding ID jedna; jde-li o font pro Macintosh, Platform ID bude 1 a Encoding ID nula. Všechna kódování Microsoft Unicode (PID = 3, EID = 1) musí používat *Formát 4* pro jejich cmap podtabulku. Microsoft doporučuje používat Unicode cmap pro všechny fonty.

Platform ID	Encoding ID	Popis
3	0	Symbol
3	1	Unicode
3	2	ShiftJIS
3	3	Big5
3	4	PRC
3	5	Wansung
3	6	Johab

Tabulka 4: *Platform a Encoding ID*

Formát 0: Byte encoding table

Toto je Apple standard cmap. Formát je nula, Length je délka podtabulky v bytech, Version je číslo verze (začíná nulou), GlyphAray[256] je pole, které mapuje kódy znaků na indexy obrysů. Lze indexovat více než 256 pozic, ale pouze 256 jich bude dostupných.

Formát 2: High-byte mapping through table

Tato tabulka je užitečná pro národní kódování znaků používaných v Japonsku, Číně a Koreji. Tento kódovací standard používá smíšené 8/16 bitové kódování, které určuje hodnota prvního bytu. Kód znaku je vždy určen jedním bytem. Množina obrysů je limitována 256 pozicemi. Tabulka začíná polem, které mapuje první byte na *subHeader.* SubHeader obsahuje dílčí pole, kam se mapuje druhý byte. Je-li subHeader roven nule, pak jde o 8 bitové kódování a druhý byte není potřeba.

Formát 4: Segment mapping to delta values

Formát 4 je standardem Microsoftu. Formát se používá, když kódy znaků pro obrysy reprezentované fontem se rozpadnou na několik styčných oblastí. Data jsou rozdělena na tři části v následujícím pořadí:

- header (hlavička) obsahující parametry pro optimalizaci vyhledávání segmentu
- čtyři paralelní pole popisující segmenty (jeden segment pro každou styčnou oblast)
- proměnlivé délky polí glyph ID.

Příklad 4: *Záhlaví podtabulky 2*

```
Subtable  2.
Platform ID:  3
     Specific ID:  1
     'cmap' Offset: 0x0000011A
  ->Format: 4 : Segment mapping to delta values
Length: 1436
Version: 0
segCount: 94  (X2 = 188)
searchRange: 128
entrySelector: 6
rangeShift: 60
```

Formát 6: Trimmed table mapping

Hodnoty firstCode a entryCount specifikují subpole jako řadu možných znakových kódů. Kódy mimo toto subpole jsou mapovány jako glyph index 0. Offset kódu v tomto subpoli je použit jako index pro glyphIdArray.

3.2. TrueTypeOpen

TrueType Open je rozšíření standardu TrueType fontů. TrueType Open obsahuje dodatečné informace, které umožňují:

- větší možnosti mapování mezi písmeny a znaky (glyphs) — podpora ligatur, pozičních tvarů, variant znaků a další
- vložit informace pro dvojrozměrné umístění a spojení znaků
- disponovat explicitními skripty a jazykovými informacemi
- definovat vlastní typografické vlastnosti fontu (features)

TrueType Open fonty lze použít pro nelatinkové písma a pro smíšené dokumenty. Může obsahovat několik variant jednoho znaku a mechanismus výběru varianty například podle místa výskytu ve slově. TrueType Open pomáhají správně orientovat znaky podle směru sazby, umožňují složení ligatur i jejich rozložení na jednotlivé znaky. Lze definovat body pro místo napojení znaků.

Skriptem se rozumí skupina příbuzných znaků, které mohou být použity jedním nebo více jazyky, například Latinský skript. Skript může být rozdělen na jazykové systémy. Latinský skript je použit pro psaní anglického, francouzského nebo německého textu.

Tag	jméno
GSUB	Glyph Substitution Table
GPOS	Glyph Positioning Table
BASE	Baseline Table
JSTF	Justification Table
GDEF	Glyph Definition Table

Tabulka 5: *Tabulky TrueType Open*

4. Metafontové fonty

TEX a METAFONT tvoří dvojici systémů, která je schopná poskytnout prostředky pro získání kvalitní tiskoviny. METAFONT je nástroj pro tvorbu fontů, TEX s těmito fonty sází a příslušné ovladače umí realizovat vlastní tisk.

4.1. Program METAFONT

METAFONT je systém na vytváření fontů pro rastrové výstupní zařízení — obrazovka nebo tiskárna. Jeho autorem je stejně jako u TEXu Arthur Donald Knuth. TEX umísťuje znaky do správných pozic na stránce, METAFONT znaky vytváří. Název systému autor vysvětluje přibližně takto: slovem FONT jsou označovány soubory příbuzných znaků, které se při sazbě nazývají znakové sady písmen (fonts of type); předpona META naznačuje, že nás zajímá vysoká úroveň deskripce, která přesahuje popis jednotlivého fontu.

METAFONT umožňuje schematický popis tvarů v rodině příbuzných fontů; změna tvarů znaků odpovídá změně jejich základních parametrů. Chceme-li použít znaky v různých velikostech, asi nás napadne možnost zvětšení nebo zmenšení základního písma; je to sice rychlé řešení, ale důsledkem je vážné snížení kvality. Mnohem lepší výsledek obdržíme připojením parametrických změn do meta-designu. Dobrým dokladem výše uvedeného jsou fonty Computer Modern, které jsou součástí instalace TEXu.

Příklad 5: *Zvětšené písmo 8pt a 10pt*

```
Písmo čtrnáct      % csr8 at 14dd
Písmo čtrnáct      % csr10 at 14dd
```

Definice dokonalého písma se v METAFONTu skládá ze tří hlavních částí. První je spíše množinou nezbytných administrativních úkonů jako přiřazení čísla kódu znaku nebo umístění znaku v boxu. Druhá část obsahuje podprogramy rozvržení základních charakteristických tahů písma (patky, dříky, ...). Třetí částí jsou rutiny zvláště pro každý znak.

4.2. Jazyk METAFONTu

METAFONT není pouhým souborem omezeného počtu příkazů, které slouží k nakreslení písmene, jako je tomu např. u Type 1 fontů, ale spíše se podobá fontům Type 3, kde lze plně využívat jazyka Postscript. Jazyk METAFONT si složitostí a promyšlenou koncepcí nezadá nejen s jazykem TEXu, ale i s jinými grafickými nástroji. Mluvit o METAFONTu jako o nástroji pro tvorbu písma není celá pravda. Svým matematickým a grafickým aparátem je stejně vhodný pro kreslení geometrických objektů, obrázků, pérovek, grafů, schémat apod. Vnějškově se však vždy jeví jako fonty a jednotlivé objekty se vysazují jako písmena.

4.2.1. Souřadnicový systém, body a cesty

METAFONT používá kartézský souřadnicový systém, každý bod je definován svou x a y souřadnicí. Počátek *origin* má souřadnice (0,0) a kryje se s referenčním bodem písmene (obrázku). Body lze v METAFONTu definovat nejen jako dvojici uspořádaných čísel, které označují polohu pomocí souřadnic, ale

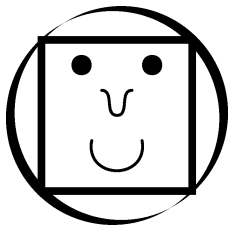
také jako obraz komplexního čísla v Gaussově rovině. METAFONT tedy umí pracovat s úhly, k vyjádření používá stupně.

Cesty jsou spojité čáry vytvořené jedním tahem. METAFONT chápe cesty také jako proměnné. Při vytváření cesty lze určit, jakým způsobem budou jednotlivé body cesty spojeny. Může je spojit úsečkami nebo křivkami. K vytváření křivek METAFONT používá Bézierových křivek. Ty můžeme definovat buď přímým zápisem nebo si je METAFONT dopočítá sám z údajů, kterými vyjadřujeme přání, jak bude naše křivka vypadat (můžeme např. určit směr tečny v bodě).

Příklad 6: *Body a cesty v Metafontu*

```
mode_setup; %nastavení délkových jednotek
u#:=1mm#; %oblíbená jednotka Karla Horáka
define_pixels(u); %přepočet jednotek na pixely
beginchar(1,50u#,50u#,0); %znak s kódem 1 a jeho velikost
path p[]; %deklarace cesty
z1=(0,0); z2=(30u,0); z3=(30u,30u);
z4=(0,30u); z5=(10u,10u); z6=(20u,10u);
z7=(6u,25u); z8=(10u,25u); z9=(12u,20u);
z10=(18u,20u); z11=(15u,15u); %definice bodů
p1=z1--z2--z3--z4--z1; %cesta je spojená úsečkami
pickup pensquare scaled 1.5u; % hranaté pero
draw p1; %vykreslení
pickup penrazor scaled 2u rotated 30; %pero s rotací 30 stupňů
draw z1..z2..z3..z4..z1; %cesta je spojena křivkami
pickup pencircle scaled .6u; %kulaté pero
draw z5{dir-100}..z6; %úsměv
p2=z7..z8..z7..cycle; %oči
fill p2; %oči vyplněné
fill p2 shifted(14u,0); %druhé oko posunutím
draw z9{dir0}..z11..z10{dir0}; %nos
endchar; %konec definice znaku
end.
```

Obrázek 14: *Body a cesty v Metafontu*



4.2.2. Algebraické výrazy a proměnné

METAFONT umí pracovat s následujícími typy proměnných:

- **boolean** true / false
- **string** posloupnost znaků
- **path** cesta, úsečka, křivka
- **pen** vzorek výplně
- **transform** operace změny velikosti, rotace, posunutí, zrcadlení a naklánění

• **pair** uspořádaná dvojice čísel (bod nebo vektor)

• **numeric** samostatné číslo

Numerické proměnné a body není nutno deklarovat, cesta se například deklarovat musí p[]. Přiřazení má symbol := jako např. v Pascalu. METAFONT umí řešit soustavy lineárních rovnic, takže může dopočítat neznámé proměnné — y1=x2-x3+y2 je definice vztahů mezi proměnnými.

Ve výrazech lze používat množství operátorů: sčítání, násobení, mocniny, goniometrické funkce, zaokrouhlování, výběr maximální a minimální hodnoty a další.

4.2.3. Pera, výplně a gumy

Než bude METAFONT něco kreslit je nutné mu sdělit jaké pero má použít. Máme celkem tři možnosti — kruhové pero, čtvercové pero, úsečka. Pera lze různě zvětšovat a naklánět. Máme možnost si vyrobit i vlastní pero, jediné omezení je, že musí mít konvexní tvar.

METAFONT umí vyplnit oblast barvou, pokud je ohraničená uzavřenou křivkou — příkaz fill. Program si také pamatuje počet překrytí barvy každého bodu, což se dá využít při gumování — příkaz unfill.

4.2.4. Podmínky, cykly a makra

Jako každý jazyk obsahuje METAFONT podmínky. Pokud není splněna podmínka if, mohou následovat další testy elseif, pokud není splněna ani jedna z podmínek provede se else.

Cyklus obsahuje pouze jeden. Jde o cyklus for, který má tři varianty: pro hodnoty ve výčtu, pro hodnoty od do s krokem (jako Basic) a nekonečný cyklus forever.

Makra jsou definována příkazem def *jméno* = *text* enddef. Vzniká tedy nový příkaz *jméno*. Makra mohou mít své parametry.

Příklad 7: *Příklad podmínky, cyklu a makra*

```
def makro(bod,uhel) = z1{dir uhel}--z2..bod..z3--{dir uhel}z4
enddef;

for i=0 step 1u until 30u:
  if z1 < i : draw p1;
  else: draw p2;
endfor;

for i=0 step 1u until 8u:
  draw(i,0)--(i,8u);
  draw(0,i)--(8u,i);
endfor; % šachovnice
```

METAFONT je složitý a mocný grafický jazyk, jeho význam překračuje tvorbu fontů. Existuje podobný program, který vychází z METAFONTu, ale jeho výstupem nejsou fonty, ale postscriptový kód. Program se jmenuje METAPOST.

4.2.5. Běžná práce s METAFONTem

Program METAFONT čte kód s příkazy jazyka METAFONT popisující font (soubor .mf) a výstupem je metrika tohoto fontu a bitmapy pro danou velikost, rozlišení a zařízení. Program lze spouštět z příkazové řádky, z menu prostředí MNU nebo je automaticky volán programem MFjob prostřednictvím dvi-ovladače.

O existenci tohoto programu se většinou začneme zajímat až v okamžiku, kdy dvi-ovladač nemůže zobrazit čitelně náš dokument ani po volbě Y na dotaz:

Warning 1209: do you want to call MFjob to generate 3 missing fonts now?

Type Y, N, ?: _

Děje se zhruba toto — příkazem \font zavedeme nový font, kterým TEX může sázet, pokud najde požadovanou metriku např. pplr8z.tfm; požadujeme-li jinou velikost at nebo scaled, TEX pronásobí velikost boxu daným koeficientem a sází. Problém nastane až při zobrazení dvi-ovladačem. Ovladač se nejdříve podívá nejde-li o virtuální font — adresář FONTS\VF. Když najde soubor pplr8z.vf, bude se řídit pokyny pro sestavení znaků podle definic v tomto souboru. Pak ovladač prohledá adresář FONTS a jeho podadresáře, kde hledá pplr8z.pk v požadované velikosti a rozlišení. Když ho najde, zobrazí dokument.

Neexistuje-li soubor pk ani vf, tak se podívá do adresáře MFINPUT, kde jsou METAFONTové zdroje fontů (pplr8z.mf). Z tohoto souboru se po volbě Y při zobrazení generují potřebné bitmapy. Většina ovladačů volá program MFjob s parametry chybějících bitmap. MFjob spouští METAFONT a získané bitmapy ukládá do adresářů: PIXEL.*zařízení**rozlišení*DPI.

S programem METAFONT lze samozřejmě pracovat i přímo z příkazové řádky nebo prostřednictvím menu METAFONT v prostředí MNU. Jako parametry je nutno zadat soubor MF, jméno báze, zvětšení a mód. Jméno báze (přípona .bas) je soubor s předdefinovanými makry (příkazy), pomocí kterých je popsán font soubor.mf — analogické s formátem plain nebo LATEX v TEXu. Většinou jde také o bázi s názvem plain. Mód je pojmenování seznamu parametrů, které ovlivňují chování METAFONTu. Určuje rozlišení a různé další „finesy“ pro výstupní zařízení i pro program METAFONT. V EmTEXu jde o MFINPUT\ETC\local.mf soubor se seznamy parametrů pojmenovaných podle výstupních zařízení. Významy jednotlivých položek jsou popsány např. v (OLŠÁK, 1995).

Výstupem je bitmapa formátu GF (Generic Font). Jde o překonaný formát bitmap, ovladače pracují s úspornějším formátem PK (PacK), který lze získat pomocí programu gftopk. Program MFjob provádí tuto konverzi automaticky, z prostředí MNU je nutné použít volbu Convertor.

4.2.6. Metafontový soubor

Následující příklad ukazuje část METAFONTového kódu písma Bodoni, které bylo převedeno z formátu Type 1 pomocí programu MF4PS. Po inicializaci a definici měrných jednotek následují popisy znaků. Nejdříve je určen kód znaku, jeho šířka, výška a hloubka beginchar.., pak jsou určeny souřadnice bodů z1=(.. a vykreslena cesta FuF, která je vyčerňována fill (definice makra v souboru ps2mfbas.mf). Na závěr jsou údaje pro metriku fontu.

Příklad 8: *Příkad MF souboru Bodoni*

```
% Bodoni-Bold-Bold
mode_setup;
if unknown FontSize: FontSize := 10pt#; fi
FX# := FontSize * 0.0010;
FY# := FontSize * 0.0010;
input ps2mfbas
...
beginchar(77,867FX#,668FY#,0FY#);
"M";
z1=(348FX,0FY); z2=(383FX,0FY); z3=(619FX,626FY);
z4=(619FX,19FY); z5=(541FX,19FY);
...
z24=(203FX,19FY); z25=(189FX,18FY); z26=(116FX,149FY);
z27=(116FX,640FY); z28=(348FX,0FY);
FuF (z1 -- z2 -- z3 -- z4
...
-- z19 {0,-110} .. {-58,0}z20 -- z21 -- z22  -- z23 -- z24 {-5,-1}
.. {-5,0}z25 {-64,0} .. {0,115}z26 -- z27 -- z28);
lbl (range 1 thru 28);
endchar;
...
ligtable 33:
96=:60;
...
font_slant := 0.0000; font_normal_space := 220 * FX#;
font_normal_stretch := 110 * FX#; font_normal_shrink := 73 * FX#;
font_x_height := 420 * FY#; font_quad := 867 * FX#;
designsize := FontSize;
font_coding_scheme := "TeX text";
font_identifier := "Bodoni-Bold-Bold";
end.
```

4.3. Produkty METAFONTu

METAFONTový soubor spojuje metrické informace a kresbu znaků. Tento formát je pouze zdrojem pro další soubory, použitelné systémem TEX a ovladači. Jde o formáty TFM a PK, jejich textové varianty a starší bitmapový formát. Začleněn byl i popis virtuálních fontů, přestože se jejich existence neváže na METAFONT, ale na ovladače a výstupy METAFONTu.

4.3.1. Soubor TFM a PL

Při sazbě zajímají TEX pouze metrické informace fontu. Tyto metriky jsou v souborech s příponou tfm (TEX Font Metrics). Formát těchto souborů je binární, ale existuje konverzní program, který tyto soubory převádí do čitelné podoby a zpět. Čitelnou verzí metrik jsou soubory pl (Property List). Konverzní rutiny tftopl a pltotf jsou součástí instalace.

Základním fontem pro českou sazbu je csr10.tfm. Vezmeme si ho z adresáře TFM\CS někam „stranou“, abychom s ním mohli dělat případné pokusy, a spustíme program tftopl a necháme se provést jeho *wizárdem,* nebo přímo z příkazové řádky tftopl \emtex\tfm\cs\csr10.

Příklad 9: *Soubor TFM — hlavička*

```
(FAMILY CMR)
(FACE O 352)
(CODINGSCHEME TEX CS TEXT)
(DESIGNSIZE R 10.0)
(COMMENT DESIGNSIZE IS IN POINTS)
(COMMENT OTHER SIZES ARE MULTIPLES OF DESIGNSIZE)
(CHECKSUM O 15736045506)
(FONTDIMEN
  (SLANT R 0.0)
  (SPACE R 0.333334)
  (STRETCH R 0.166667)
  (SHRINK R 0.111112)
  (XHEIGHT R 0.430555)
  (QUAD R 1.000003)
  (EXTRASPACE R 0.111112)
  )
```

Hlavička obsahuje základní informace o fontu. FAMILY označuje rodinu fontu, zde jde o Computer Modern Roman. FACE specifikuje font uvnitř rodiny, tento oktalovým čílem 352. CODINGSCHEME říká o jaké jde kódování. DESIGNSIZE je velikost písma v bodech. COMMENT je komentář. Kontrolní součet CHECKSUM je uveden jak v souboru TFM, tak i v bitových mapách a mělo by být shodné.

Údaje FONTDIMEN:

SLANT sklon fontu; informace pro umístění akcentů a kurzívní korekce
SPACE základní mezera mezi slovy
STRETCH maximální doporučená roztažitelnost mezery
SHRING maximální doporučená stlačitelnost mezery
XHEIGHT výška písmene X; pro umístění akcentů; jednotka *ex*
QUAD čtverčík, emsize, šířka písmene M; jednotka *em*

V matematických fontech bývá množství údajů mnohem větší: umístění mocnin, odmocnin, indexů, ...

Před číselnými údaji se nachází jedno písmeno formátu čísla: O oktalové, R reálné, D decimální, před znakem C chracter.

Příklad 10: *Soubor TFM — ligatury a kerning*

```
(LIGTABLE
  (LABEL O 40)
  (KRN C I R -0.277779)
  (KRN C L R -0.319446)
  (STOP)
  (LABEL C f)
  (LIG C i O 14)
  (LIG C f O 13)
  (KRN O 77 R 0.077779)
  (KRN O 41 R 0.077779)
  (KRN O 51 R 0.077779)
  (KRN O 135 R 0.077779)
  (STOP)
  ...
```

Za hlavičkou následují informace o kerningu a ligaturách. Uvozeny jsou slovem LIGTABLE a říkají nám: znak oktalově 40 (mezera) má se znakem *l* záporný kerning 0,277779, se znakem *L* −0,319446. Znak *f* následovaný znakem *i* tvoří ligaturu a dává znak s oktalovým číslem 14 (*fi*).

Příklad 11: *Soubor TFM — informace o znacích*

```
...
(CHARACTER C f
  (CHARWD R 0.305557)
  (CHARHT R 0.694445)
  (CHARIC R 0.077779)
  (COMMENT
    (LIG C i O 14)
    (LIG C f O 13)
    (LIG C l O 15)
    (KRN O 47 R 0.077779)
    (KRN O 77 R 0.077779)
    (KRN O 41 R 0.077779)
    (KRN O 51 R 0.077779)
    (KRN O 135 R 0.077779)
    )
  )
...
```

Po ligační a kerningové tabulce následují definice znaků. Znak (CHARACTER) je označen kódem (C f) a může mít čtyři údaje CHARWD, CHARHT, CHARDP a CHARIC u skloněných fontů. Údaje udávají šířku znaku, výšku znaku nad účařím, přesah pod účaří a doporučená mezera za znakem pokud bude následovat přímé písmo. V TEXu příkaz \/. Dále se dovíme, že tvoří ligaturu s písmenem *i*, *f* a *l*, což dá znaky s oktalovým číslem 14, 13 a 15 (*fi, ff, fl*).

4.3.2. Soubor VF a VPL

Soubor VF (Virtual Font) a VPL (Virtual Property List) jsou párové soubory, z nichž VPL je čitelná podoba binárního VF. Samozřejmě existují převodní programy vftovp a vptovf.

Vidíme, že hlavička souboru VPL je stejná jako u metrik PL. Je tomu tak proto, že při obrácené konverzi vptovf se generuje jak soubor VF, tak i soubor TFM.

Příklad 12: *Soubor VPL — hlavička*

```
(VTITLE Smyšlený soubor, základem Times New Roman)
(FAMILY TEX-RPTMR)
(FACE F MRR)
(CODINGSCHEME XL2ENCODING + ADOBESTANDARDENCODING)
(DESIGNSIZE R 10.0)
(COMMENT DESIGNSIZE IS IN POINTS)
(COMMENT OTHER SIZES ARE MULTIPLES OF DESIGNSIZE)
(CHECKSUM O 7575461244)
(FONTDIMEN
  (SLANT R 0.0)
  (SPACE R 0.25)
  (STRETCH R 0.2)
  (SHRINK R 0.1)
  (XHEIGHT R 0.448)
```

```
  (QUAD R 1.0)
  (EXTRASPACE R 0.111)
  )
(MAPFONT D 0
  (FONTNAME rptmr)
  (FONTCHECKSUM O 30202316533)
  (FONTAT R 1.0)
  (FONTDSIZE R 10.0)
  )
(LIGTABLE
 (LABEL O 47)
  (LIG O 47 O 42)
...
```

Příkaz MAPFONT vymezuje fonty, z nichž bude složený virtuální font. Zde jde o font Times New Roman (rptmr). Položka FONTAT určuje zvětšení fontu; v tomto případě nebude žádné. FONTDSIZE určuje velikost písma v bodech. Sekce MAPFONT může být opakovaná podle toho kolik budeme potřebovat vstupních fontů. Následující číslo označuje číslo fontu. Lze se odvolat i na virtuální fonty, ale nesmí docházet k cyklům.

Příklad 13: *Soubor VPL — definice znaků*

```
(CHARACTER O 344
  (CHARWD R 0.444)
  (CHARHT R 0.6305)
  (CHARDP R 0.0085)
  (MAP
    (SETCHAR C a)
    (MOVERIGHT R -0.389)
 (MOVEUP R 1.12)
    (SETCHAR O 302)
    ))
  ...
  (CHARACTER O 301
    (CHARWD R 0.722)
    (CHARHT R 0.675)
    (COMMENT
      (KRN O 335 R -0.092)
      (KRN C T R -0.111)
      (KRN C V R -0.129) )
  (MAP
    (PUSH)
 (SPECIAL ps: gsave 1 0 0 setcolor)
 (SELECT D 2)
 (SETCHAR C A)
 (SPECIAL ps: grestore)
 (POP)
 (MOVERIGHT R 0.309)
 (MOVEUP R 0.685)
 (SETCHAR O 302)
 (MOVERIGHT R 0.312)
 ))
...
```

Základní parametry písmene jsou shodné jako u formátu TFM. Deklarace MAP říká, z čeho se bude výsledný znak skládat. Zde to bude znak *a* SETCHAR, pak se posune aktuální bod do leva o 0,398 MOVERIGHT a nahoru o 1,12 MOVEUP a na toto místo se namapuje znak oktalově 302 (čárka). Tak je realizováno písmeno *á*.

Druhý znak *Á* je realizován červeně pomocí postscriptových příkazů, a to z fontu 2 (SELECT). Operátory PUSH a POP uloží a obnoví aktuální bod sazby, příkaz grestore nastaví původní barvy tisku.

4.3.3. Soubor GF, PK, PXL a PKedit

Soubor GF (Generic Font) je spolu se souborem TFM výstupem METAFONTu a formátem bitmapového obrazu fontu. V DOSu nemusí mít příponu gf, ale pouze číslo označující rozlišení, např. 300. Soubor formátu PK, s kterou pracují dvi-ovladače, je komprimovaná verze formátu GF. Pro převod se používají programy gftopk a pktogf. Zastaralá forma PXL je formát, s kterým sice ovladače nepracují, ale používají ho některé utility jako svůj výstup, např. Rumgraph. Programem pxltopk, lze tento soubor připravit k použití pro běžné dvi-ovladače.

Pro prohlížení a přímou editaci může posloužit program PKedit. Soubory lze editovat buď myší, jako v PaintBrushi, anebo klávesnicí.

5. Fonty v TEXu

5.1. TEX

TEX je typografický systém napsaný Donaldem E. Knuthem ze Stanfordské univerzity. Je určen pro velmi kvalitní sazbu knih s množstvím matematiky, ale to není podmínkou. Svůj produkt D. E. Knuth daroval světu zdarma a od roku 1983, kdy byla dokončena verze 3, nebyla v systému provedena žádná zásadní změna v jeho koncepci.

Název TEX se vyslovuje „tech“, protože anglické slovo *technology* pochází z řeckého kořene začínající písmeny $\tau\epsilon\chi$; toto slovo v řečtině znamená nejen technologii, ale i umění.

Program lze používat i pro nelatinkové abecedy, lze v něm sázet zprava doleva, nebo i vertikálně. TEX je implementován na mnoha platformách (DOS, Windows, Linux, ...). Většina těchto implementací je freeware, ale existují i komerční produkty.

Uživatelé TEXu mají svá sdružení (většinou podle národnosti); prvním byl americký TUG (TEX User's Group), u nás vznikl CSTUG (Czech and Slovak TEX User's Group). Tato organizace vytváří a šíří instalační balík TEXu, který je přizpůsoben české a slovenské sazbě.

5.1.1. Práce TEXu

Jádro systému tvoří překladač jazyka TEXu, jehož vstupem je textový soubor pořízený libovolným editorem. Do tohoto souboru se vedle vlastního textu zapisují také příkazy, které určují, jak má být text vysázen. Hlavní prací překladače je rozmístění jednotlivých znaků do sazebního zrcadla. K tomu potřebuje znát rozměry znaků. Všechny znaky jsou v tomto okamžiku chápány jako obdélníky, jejichž rozměry jsou soustředěny v souborech s rozšířením .tfm (TEX Font Metric). Průběh překladu a všechna varovná a chybová hlášení jsou překladačem vypisována na obrazovku a současně i do textového souboru s rozšířením .log.

Výstupem překladače je soubor s vysázeným textem (.dvi), který je vytvořen tak obecně, aby jej bylo možno zpracovat na různých finálních zařízení. Jeho obsah je tedy nezávislý na zobrazovacím zařízení (DeVice Independent). Abychom mohli vidět vysázený text, je nutné tento soubor zpracovat dalším programem, který jej na daném zařízení zobrazí. (RYBIČKA, 1995)

5.1.2. Kódování v TEXu

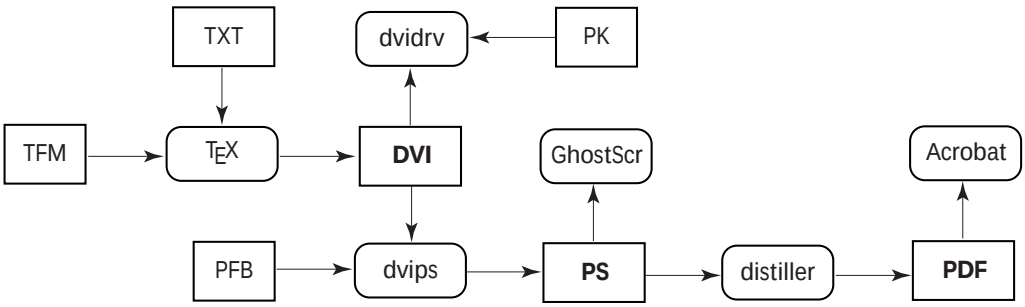
Kódování v TEXu můžeme rozdělit do dvou oblastí: kódování dokumentu a kódování fontů. TEX pracuje ve svém vnitřním kódování, do kterého jsou převáděny znaky z kódování dokumentu pomocí dvou (vstupní a výstupní) vektorů. Vnitřní kódování TEXu je nezávislé na operačním systému. Nastavení kódovacích vektorů je často možné jen před kompilací TEXu ze zdrojových souborů. Některé implementace umožňují tyto vektory měnit i po kompilaci — v EmTEXu tcp tabulkami v podobě inicializačního formátu nebo na Unixových instalacích prostřednictvím EncTEXu až za běhu TEXu.

Kódování fontů je stejné jako kódování v dvi souboru, které lze modifikovat pomocí virtuálních fontů. Virtuální fonty obsahují návod, jak namapovat znaky ze souboru s jiným kódováním do našeho kódování nebo jak sestavit znak na dané pozici ze znaků jiných fontů. Možnosti virtuálních souborů jsou využívany při používání postscriptových a TrueType fontů v TEXovské sazbě. Podrobně je tato problematika popsána v (OLŠÁK, 1993) a (OLŠÁK, 1997).

5.2. Jak TEX pracuje s fonty

Jak bylo uvedeno výše, TEX nepracuje s vlastními obrazy písmen (fontů), ale pouze s jejich *metrikami*. Tyto metriky jsou uloženy v souborech *.tfm (TEX fonts metrics). Pokud chceme, aby TEX sázel v nějakém nestandardním fontu, který nebývá součástí TEXovských instalací, pak principiálně stačí získat z tohoto fontu metrické informace a transformovat je do formátu TFM. Pak musíme použít dvi-ovladač, který s těmito fonty umí pracovat. (OLŠÁK, 1995)

Obrázek 15: *Typiká práce TEXu, ovladačů a převodníků*



Instalace TEXu implicitně nabízí fonty Computer Modern, jejichž autorem je opět D. E. Knuth. Bitmapové obrazy písem mají formát PK (PacK bitmap), s kterými umí pracovat každý dvi-ovladač. Bitmapy je nutné generovat pro každé rozlišení zvlášť, např.: jehličkové tiskárny 180 DPI, inkoustové a laserové 300 a 600 DPI, atd. Protože jde o malé soubory, mohou být vytvořeny knihovny těchto bitmap — soubory fli.

5.2.1. Základní příkazy pro práci s fonty

Při zavedení TEXu je implicitní font Computer Modern, antikva ve velikosti 10 pt. Plain TEX poskytuje následující příkazy pro změnu řezu:

\rm	normální písmo	Roman
\sl	skloněné písmo	*Slanted*
\it	kurzíva	*Italic*
\tt	psací stroj	`Typewriter`
\bf	tučné písmo	**Bold face**

Příklad 14: *Zavedení fontu v TEXu*

```
\font\malyroman=csr10
\font\vetsiroman=csr12
\font\velikyroman=csr12 at 24pt
\font\zvetsenyroman=csr10 scaled 2000
\font\geomerickyroman=csr10 scaled\magstep2
```

Zavedení nového fontu v plain TEXu:

\font*jméno=externí jméno fontu* at *požadovaná velikost*

Příkaz \font zavede do paměti font ze souboru *.tfm (např. csr10.tfm) a přiřadí mu jméno (malyroman), které bude použito jako přepínač pro jeho aktivaci. Chceme-li zvětšený font, použijeme příkaz at a za ním požadovanou velikost. Násobnou změnu velikosti lze provést příkazem scaled *číslo*; scaled 1000 znamená 1:1 (tedy žádné zvětšení), scaled 2000 provede dvojnásobné zvětšení, scaled 750 dodá 75 % původní velikosti.

V typografii je časté odstupňování písma koeficientem 1,2. Příkaz \magstep2 vyjadřuje *základní velikost* × 1,44. Změna velikosti fontu znamená pro TEX pouze pronásobení metriky.

Velikosti písma se udávají číslem a označením pro měrný systém. V našem geografickém prostoru je domovský **didotův** měrný systém.

pt	point
pc	pica (1 pc = 12 pt)
in	inch (1 inch = 72.27 pt)
bp	big point (72 bp = 1 inch)
mm	milimetr (10 mm = 1 cm)
cm	centimetr (2,54 cm = 1 inch)
dd	didotův bod (1157 dd = 1238 pt)
cc	cicero (1 cc = 12 dd)
sp	scaled point (65536 sp = 1 pt)

Tabulka 6: *Tabulka zkratek měrných systémů*

V původních (sedmibitových) Computer Modern (cm) fontech nejsou obsažena akcentovaná písmena české abecedy. Tyto znaky je nutné sestavovat.

Pro české akcentované znaky jsou implementovány následující příkazy:

\'	čárka	\'a	→á
\v	háček	\v{c}	→č
\r	kroužek	\r{u}	→ů

Lze získat i další akcenty, příkazy jsou popsány např. v (RYBIČKA, 1997).

Instalace CSTEXu obsahuje akcentované české fonty tzv. CSfonty (cs). CSfonty jsou osmibitové a obsahují všechny české a slovenské znaky, které mají vhodné umístění akcentů, naopak od algoritmického usazování pomocí příkazů.

Seznam CSfontů je v příkladové části, adresáři CSFONTS.

5.2.2. Virtuální fonty

Virtuální fonty vznikly dlouho po vzniku TEXu spolu s vzrůstem nového jazyka pro popis tiskové strany — Postscriptu. Virtuální fonty umožňují nazývat písmenem, znakem i větší část vysázeného textu, např. znak *č*, což je v Knuthových CM fontech sekvence \v{c}. Virtuální font neobsahuje žádné křivky, jde vlastně o popisy, jak skládat písmena (virtuálního fontu) z jiných (několika normálních) písmen fontů na základě metrických informací o znacích. (SOJKA, 1994)

Virtuální fonty pracují mimo TEX (překladač o nich neví a nepracuje s nimi); jde o jakousi „falešnou“ metriku k neexistujícímu fontu, podle které TEX sází. Až na úrovni dvi-ovladačů dojde ke zjištění, že bylo sázeno virtuálním fontem, a pak už jen záleží na schopnostech ovladače, jak si poradí. Virtuální fonty mají příponu vf a obsahují informace pro dvi-ovladač jak sestavit tvary znaků virtuálního fontu.

Virtuálním fontem lze provést:

1. **Nahrazení znakem z jiného PK fontu.** Tato možnost nabízí sestavit virtuální font jinak kódovaný, než je fyzický font ve tvaru PK.
2. **Vytvoření znaku pomocí jazyka výstupního zařízení.** Znak virtuálního fontu může být realizován jako sekvence příkazů srozumitelná jen pro konkrétní výstupní zařízení.
3. **Sestavení z komponent.** Znak virtuálního fontu může být sestaven z většího množství elementárních znaků, z nichž každý může být realizován jiným způsobem.

V praxi se virtuální fonty nejvíce používají při práci s postscriptovými fonty, kde slouží k sestavení kompozitních (akcentovaných) znaků a sladění rozdílných kódování.

5.2.3. Makro NFSS

Makro NFSS (New Font Selection Scheme) umožňuje zavádět a použít v TEXu rozsáhlé skupiny písem přehledným a dobře strukturovaným způsobem. Toto makro vytvořil Frank Mittelbach a Rainer Schöpf. Hlavní myšlenka spočívá ve vytvoření přepínačů mezi jednotlivými fonty, které nepřepínají absolutně, ale změní jen některé parametry písma, které jsou na sobě nezávislé. Každý font zavedený makrem NFSS je určen vektorem s pěti parametry. (OLŠÁK, 1995)

1. Kódování fontu (encoding):

OT1	Knuthovy CM fonty
T1	kódování DC fontů (Cork)
XL2	kódování CS fontů
OML OMS OMX	matematické fonty
L<*xx*>	jiná lokální kódování

2. Rodina písma (family):

cmr	Computer Modern Roman
cmss	Computer Modern Sans Serif
cmbx	Computer Modern Boldface
ptm	Adobe Times
phv	Adobe Helvetica
...	další

3. Duktus (series), váha: m medium (střední), b bold (tučné), bx bold extended (tučné rozšířené), sb semibold (polotučné), c condensed (zúžené).

4. Varianta (shape), tvar: n normal (románské písmo), it italic (kurzíva), sl slanted (skloněné), sc small caps (kapitálky).

5. Stupeň (size): udává se jako libovolná míra (12pt, 6mm, 10dd).

Přepínání slouží následující příkazy: \fontencoding{*kódování*}
\fontfamily{*rodina*}
\fontseries{*váha*}
\fontshape{*varianta*}
\fontsize{*stupeň*}
\selectfont aktivace nastaveného fontu (následuje za výše uvedenými příkazy)

NFSS definuje další příkazy např. \rmfamily, \sffamily, ...

Vazby mezi pěti parametry a soubory metrik jsou uvedeny v souborech fd (font definition). Názvy těchto souborů jsou pevně stanoveny: název kódování a rodina (OT1cmr.fd).

5.2.4. Standardní postscriptové fonty

V balíku instalace CSTEXu můžeme používat standardní počeštěné postscriptové fonty. Pro tyto fonty jsou vytvořeny nejen metrické informace, ale i nejběžnější bitmapy (*.pk soubory), takže výsledek můžeme sledovat přímo v integrovaném dvi-prohlížeči. TEXu nebude vadit, když bude sázet i v neběžných velikostech, ale pak v dvi-prohlížeči uvidíme místo písmen pouze jakési „boxy“. Tato situace není neřešitelná; výsledný dvi-soubor může exportovat do Postscriptu pomocí DVIPS a vlastní prohlížení a tisk provést jinými ovladači a programy.

Jde o tyto fonty:

soubor	písmo
cavantga	Avantgarde Book
cbookman	Bookman
chelvet	Helvetica
cncent	New Century Schoolbook
cpalatin	Palatino
ctimes	Times Roman

Tabulka 7: *Postscriptové fonty v instalaci TEXu*

Pro zavedení těchto písem v plain TEXu stačí zapsat \input cavantga nebo \input ctimes a nastaví se automaticky normální řez daného písma. Jsou implementovány základní přepínače \tenrm, \tenit, \tenbf, \tentt a \tensl. Část přepínače \ten.. překvapivě neznamená nic jiného než velikost deseti bodů. Chceme-li používat jinou velikost např. 12pt použijeme \magnification\magstep1 \input cpalatin. Druhá možnost se nám nabízí použitím příkazu \font a souboru tfm. Název daného souboru můžeme zjistit v souboru pro zavedení písma:

Příklad 15: *Soubor cpalatin.tex (Palatino).*

```
\font\tenrm=pplr8z at 10pt
\font\tenbf=pplb8z at 10pt
\font\tenit=pplri8z at 10pt
\font\tentt=pcrr8t at 10pt
\let\tensl=\tenit
\tenrm
```

Budeme-li chtít sázet písmem Palatino velikostí 10 didotových bodů napíšeme:

Příklad 16: *Použití neběžných velikostí postscriptových fontů.*

```
\font\palrm=pplr8z at 10dd % normální 10 didotů
\font\palbf=pplb8z at 10dd % tučné 10 didotů
\font\palit=pplri8z at 10dd % skloněné 10 didotů
\font\palnadpis=pplb8z at 12dd % písmo pro nadpis zvětšené tučné 12dd

\palnadpis Pozor! \par
\palrm Budeme-li chtít tento příklad vidět pomocí dvi-ovladače, tak
uvidíme, že \palbf nic neuvidíme. Musíme sehnat někde bitmapy fontů
v˜požadovaných velikostech např. pomocí PS2PK nebo export do \PS{}u.
```

5.2.5. TEX a Postscript: DVIPS, PSTricks, GhostScript

Postscript přináší nový rozměr do sazby TEXem a řeší množství problematických oblastí sazby s tímto systémem. Postscript je určitým standardem ve světě DTP a zároveň získáváme možnost práce s barvou, různé efekty, vkládání obrázků, používání postscriptových fontůa tisk na postscriptových zařízení.

Tomas Rokicki napsal volně šířitelný program **DVIPS**, který umožňuje převod výstupního DVI souboru TEXu do Postscriptu. Tento balík je v instalaci CS TEXu, a má svou vlastní položku v prostředí MNU (Others). Program do výsledného postscriptového souboru *zapisuje přímo bitmapové obrazy* písmen, proto musíme zadat parametr rozlišení fontů, aby program věděl, jaké bitmapy má zařadit do postscriptového souboru.

Bude-li mít cílové zařízení jiné rozlišení než náš výstupní soubor, dojde ke snížení kvality. Vhodné je zadat rozlišení souboru podle cílového zařízení; popř. generovat více souborů v různých běžných rozlišeních.

V případě sazby postscriptovým fontem se do postscriptového souboru nemusí zapisovat bitmapy, ale lze nastavením příslušných parametrů programu sdělit, že chceme tento font přibalit na začátek výstupního souboru nebo že tento font dodáme ovladači sami nebo ho přímo ovladač obsahuje (např. postscriptové tiskárny). Rastrování písmen provádí z postscriptových fontů při tisku nebo prohlížení postscriptový RIP (Raster Image Processor).

Získáme-li TEXovskou metriku a počítáme s prohlížením dokumentu pomocí intepretu Postscriptu, zapíšeme do souboru DATA\DVIPS\psfonts.map následující řádek:

soubor_metriky_bez_přípony FontName < cesta_a_soubor.pfb

Tím jsme řekli programu DVIPS, že při konverzi dvi na Postscript má zapsat odkaz na FontName do Postscriptového souboru a přibalit soubor.pfb, bude-li dokument obsahovat font soubor_metriky_bez_přípony. Ne shodou okolností část řádku (až do <) je shodná s výpisem programu Afm2tfm po úspěšném provedení. FontName je totožné s hodnotou položky FullName v souboru AFM. Kratším zápisem: soubor_metriky_bez_přípony FontName říkáme, že font dodáme výstupnímu zařízení sami nebo ho bude obsahovat samo zařízení.

V TEXu je možné používat přímo příkazy Postscriptu. To zajistí primitivní příkaz \special{*blok*}; blok je přenášen do výstupního souboru.

Pro uživatele cizích maker existuje balík **PSTricks,** jehož autorem je Timothy Van Zandt. PSTricks je soubor postscriptově založených maker, které umož-

ňují práci s barvou, grafikou, můžeme provádět různé rotace a překrývání objektů. Autor v úvodu User's Guide říká: *PSTricks put the icing (Postscript) on your cake (TEX) !*

Výsledný dvi soubor musíme opět převést do Postscriptu pomocí DVIPS, teprve na zařízení, které pracuje s Postscriptem, bude vidět výsledek našeho snažení. Nás bude místo kreslení bude spíše zajímat možnost naklánění a transformace textu, abychom získali pozměněný řez písma nebo psali zrcadlově, obráceně, vertikálně atp.

Vertikální a obrácený text

```
\rotateleft{Left} % rotace vlevo
\rotateright{Right} % rotace vpravo
\rotatedown{Down} % obrácený text
```

Natahování textu

```
\scalebox(2 1){Natažený}   % parametry označují horizontální
\scalebox(1 2){Protažený.}  % a vertikální natažení
\scalebox{2}{Jednou takový} % je-li jenom jeden parametr, dojde ke
\scalebox{-1 1}{Zrcadlový}  % konstatnímu zvětšení
```

Text na křivce

```
\psset{linestyle=none}
\pstextpath[c]{\psarcn(0,0){43pt}{180}{0}}{První národní Medvědárium}
\pstextpath[c]{\psarc(0,0){43pt}{180}{0}}{Pět Medvědů s Cibulkou}
```

Barva textu

```
\red George Gershwin: \blue Rhapsody in blue.
```

GhostScript je název pro program, který je interpretem Postscriptu a PDF (Portable Document Format). Umožňuje prohlížení těchto dokumentů na obrazovce, tisk postscriptových souborů na nepostscriptových tiskárnách a převod mezi těmito formáty. Dále lze použít k získání metriky AFM ze souboru PFM pomocí „chytré dávky" (bude uvedeno níže) nebo k extrakci holého (ASCII) textu z postscriptového souboru.

Dalším přínosem je schopnost interpretovat Type 3 fonty. Pokud není font přibalen do PS souboru, musíme umístit tento font do podadresáře FONTS a do souboru Fontmap zapsat řádek: /FullName (soubor.pfa);. To samé platí i pro fonty Type 1.

GSview je grafický interface GhostScriptu, který přináší sympatické ovládání a prostředí. Důležité je, že oba programy jsou freeware. Jsou umístěny na: http://www.cs.wisc.edu/~ghost/.

5.2.6. Adresáře instalace EmTEX

metriky	TFM
bitmapy	FONTS\PIXEL.xxx
virtuální popisy	FONTS\VF
metafontové zdroje	MFINPUT
zavedení Postscriptových fontů	TEXINPUT\CSPLAIN
programy	BIN

6. Převody fontů do prostředí TeXu

Tato kapitola rozebírá postup při přípravě fontů pro použití v jednom konkrétním typografickém systému. Zvolen byl systém TEX, protože vlastnosti prostředí, kde příprava fontu probíhá, jsou transparentní a dobře srozumitelné. Druhým důvodem pro zvolení systému TEX byl fakt, že tato problematika je dosud nezmapovaná a proniknutí do ní vyžaduje vynaložení velkého úsilí při shromažďování materiálů, studia dokumentací, testování a úpravy programů.

Členění této kapitoly bylo zvoleno podle logických celků: získání metrik, bitmap, metafontových zdrojů a popis důležitých utilit. Podkapitoly, které popisují programy mají členění: základní údaje, syntaxe, příklad a popis detailních vlastností. Příklady mají demonstrovat možnosti programu.

6.1. Teoretický základ

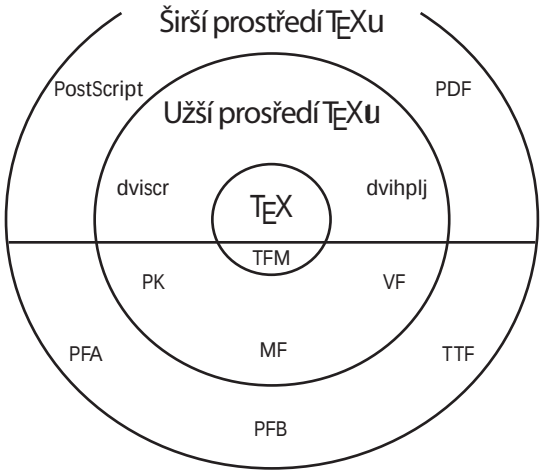
Před vlastní přípravou fontu je třeba si uvědomit, *čeho* chceme dosáhnout a *jakou* cestu zvolíme. Málokdy narazíme na situaci, kdy se nám nabízí pouze jedno řešení problému. Dobrou volbou si můžeme ušetřit mnoho úsilí i zklamání.

6.1.1. Principy konverzí z hlediska prostředí TEXu

TEX je pro určitý typ tiskovin ideálním typografickým nástrojem. Naprosto odlišná koncepce programu dovoluje mnohem vyšší přenositelnost výsledků do jiných operačních systémů, jiných grafických formátů a jiných časových horizontů, než většina komerčních aplikací. S odlišnou koncepcí souvisí i to, že lze mluvit o převodech formátů fontů *do prostředí TEXu* a ne pouze *do TEXu*.

Jak již bylo několikrát zdůrazněno, program TEX potřebuje pouze metrické informace (samozřejmě i kerningové a slitkové). Zobrazení nebo tisk je záležitostí ovladačů nebo převodníků do jiných grafických formátů. Konverzí do TEXu rozumíme pouze získání TEXovské metriky TFM. Za konverzi do prostředí TEXu v *užším smyslu* můžeme považovat soubory fontů MF, PK a VF, které tvoří spolu s dvi-ovladači přirozené prostředí TEXu; v *širším smyslu,* pak všechny možnosti zpracování DVI souboru, které vedou k využití původních formátů fontů v jejich přirozeném prostředí.

Obrázek 16: *Prostředí TEXu*



S TEXem je spjatý pouze grafický formát DVI, jehož zpracování je závislé na existenci ovladačů. Lze naprogramovat dvi-ovladač, který bude sám rastrovat font z původního formátu, nebo jako ostatní aplikace žádat o vyrastrování fontu operační systém. Tato myšlenka je realizována v implementaci TEXu pro Windows BaKoMaTEX.

Při výstupu z užšího prostředí TEXu může docházet k transformaci DVI souboru na jiný grafický formát. Nejčastější variantou je Postscript a progresivní PDF (Portable Document Format). Tato otevřenost systému je znázorněna na obrázku neuzavřeným kruhem, protože širší prostředí TEXu je limitováno pouze existencí ochoty a smysluplnosti vytváření nových ovladačů a převodníků.

Důvody konverzí fontů do TEXu:

- malý počet fontů v metafontových zdrojích
- velké množství odzkoušených a kvalitních, zejména postscriptových fontů
- snadné počeštění fontu
- využití kvalitní sazby, kterou poskytuje TEX
- využití a vytváření chybějících ligačních informací
- využití a snadná úprava kerningových informací
- editace a úpravy fontů bez asistence komerčních produktů

Při konverzi fontů se můžeme zvolit různou úroveň převodu. Ta je charakterizována získanými soubory a vazbou na původní vzor. Můžeme rozlišit čtyři úrovně konverzí do prostředí TEXu:

Lehká — pouze na úrovni metrik a virtuálních fontů; u této varianty bude grafický výstup realizován jinými programy, často i jinými grafickými formáty (např. Postscript) nebo dvi-ovladači se schopností rastrovat použité fonty (touto cestou se ubírá již zmíněný BaKoMaTEX).

Silná — získání metriky, virtuálních popisů a PK bitmap pro dvi-ovladače; kladem je, že zůstáváme v užším prostředí TEXu a můžeme využívat výhody s tím spojené (např. znalost systému, vhodný dvi-ovladač, rychlá kontrola výstupu); problematická může být kvalita získaných bitmap, vázaná na použité konverzní nástroje a kresbu písma.

Úplná — získání metafontových zdrojů; jde o ideální metodu, která umožní z jednoho (nebo více) souborů generovat potřebné bitmapy; tato varianta zaručuje spolupráci existujících ovladačů na všech implementacích TEXu; problém s kvalitou generovaných bitmap je ještě vystupňován oproti předchozí úrovni.

Konečná — vzniká úpravami, které přerušují zpětnou vazbu s původním formátem; jde o situaci, kdy je font přizpůsoben užšímu prostředí TEXu tak, že jeho nové vlastnosti nelze v původním prostředí vyvolat — úpravy tahů písma v MF souboru, vlastní úprava bitmap, využití netradičních postupů při vytváření virtuálních fontů (např. vazba na služby dvi-ovladače).

Ve většině případů je nutná změna kódování — pro české a slovenské podmínky jde o XL2, což je rozšířená verze ISO Latin 2 (pro variantu psacího

stroje jde o XT2). Bez ohledu na úroveň konverze i cílové formáty fontů musíme vyvinout snahu o maximální věrnost původnímu vzoru. Pro prostředí TEXu i jiné konverze můžeme stanovit následující požadavky:

- minimalizace nepřesností a ztrát
- minimalizace kroků
- následná plná funkčnost

Při každém převodu může docházet k zaokrouhlovacím chybám nebo chybám algoritmů; z toto vychází i druhý bod — čím více konverzí, tím větší pravděpodobnost výskytu a nahromadění chyb. Transformovaný font by měl vykazovat alespoň všechny vlastnosti, jaké měl před konverzí a musí v prostředí TEXu fungovat zamýšleným způsobem.

6.1.2. Principy konverzí z hlediska počeštění fontů

Prostředí TEXu můžeme ocenit při počešťování fontů. Počeštění lze provést bez použití profesionálních a většinou komerčních nástrojů. Existují programy, které umožní počeštit postscriptový nebo TrueType font, třeba jen se znalostmi struktury a syntaxe TEXovských metrik. Zatímco pro jiné aplikace není možné tento font počeštit bez speciálních programů — tedy nelze ani použít pro českou sazbu — TEX nám tuto cestu otvírá.

S mnoha programy dostáváme spoustu kvalitních a vhodných fontů, ale s chybějícími akcentovanými znaky. Proto se při volbě písma musíme omezit na malý počet, často nekvalitních fontů poskytovaných operačním systémem. Někdy sice vlastníme český font, ale neodborný způsob počeštění je jen výzvou k naší vlastní, kvalitnější verzi.

Český font můžeme získat několika způsoby:

1. český kvalitní font (není třeba provádět počeštění)
2. počeštění fontu v původním formátu
3. počeštění metriky postscriptového fontu
4. počeštění metriky TFM pomocí virtuálních fontů
5. získání českých znaků příkazy TEXu

Nejlepší situací je první varianta. Počeštění původního formátu je vhodné pokud zamýšlíme použití i v jiných systémech nebo dáváme přednost vizuální úpravě znaků (např. akcentů). Někdy je třeba v původním fontu vytvořit některé znaky, často jde o ligatury, háček, čárku a kroužek. Vhodná je samostatná přítomnost i ostatních akcentů. Někdy pouze stačí opravit názvy znaků a máme funkční český font. V této části nesmíme opomenout doplnění kerningových informací — nestačí pouze vytvořit obrysy.

Algoritmický, a tím i rychlý je postup počeštění původní metriky, která je dále transformována do TEXovských formátů. Požadavkem je přítomnost samostatných akcentů. Výhodou je i možnost automatického doplnění chybějících kerningových párů. Stejného efektu lze dosáhnou i v užším prostředí TEXu, ale ztrácíme vazbu na původní metriku. Tyto varianty jsou pracnější na odladění.

Počeštění na úrovni TEXovských sekvencí není vhodné, protože je z větší části typograficky nevhodné a ztrácíme některé vlastnosti TEXu (dělení slov). Výhodou je přenositelnost do zahraničí, kde se české prostředí neočekává.

Požadavky na počeštěný font:

1. musí obsahovat potřebné znaky nebo komponenty k sestavení znaků,
2. musí existovat odpovídající metrika fontu,
3. počeštění musí být typograficky únosné.

6.2. Získání metrik

Získání kvalitních metrik fontu je základním požadavkem pro sazbu TEXem. Pokud jde o postscriptové fonty, využívá se metriky AFM, která je podobná TEXovskému property listu. U TrueType fontů neexistuje metrická informace oddělená od kresby, proto se pro konverzi musí použít binární forma fontu.

Do této podkapitoly byly zařazeny i programy, které počešťují nebo jinak upravují metrické informace pro prostředí TEXu. Pro funkci v systému je třeba zařadit získané soubory do adresářové struktury, která se může lišit podle implementace systému. Metrické informace stačí pro sazbu TEXem, ale pro vizuální podobu je třeba různých transformací souboru dvi nebo dodat PK soubory.

6.2.1. Program Afm2tfm

Program Afm2tfm je součástí instalace CSTEXu, balíku dvips, jehož autorem je Tomas Rokicki. Afm2tfm převádí postscriptovou metriku AFM na metriku TFM a popřípadě i virtuální font. S písmem lze provádět postscriptové transformace a měnit kódování výstupních souborů. Program je společně s DVIPS a PS2PK základním nástrojem pro práci s postscriptovými fonty. Posledním řádkem po úspěšné konverzi je text zapisovaný do psfonts.map. Syntaxe:

```
afm2tfm foo[.afm] [-O] [-u] [-v -V bar[.vpl]] [-e expansion] [-s slant]
                [-c capheight] [-p -t -T encodingfile] [foo[.tfm]]
```

Program Afm2tfm má množsví přepínačů:

-e ratio	tvorba rozšířené a zúžené varianty písma
-c scale	tvorba kapitálek, s volbou -V
-O	výstup VPL bude v oktalových číslech
-p file	soubor pro Postscriptové kódování
-s slant	tvorba skloněné varianty
-t file	soubor pro TEXovské kódování
-T file	soubor pro kódování Postscriptové i TEXovské
-u	do výstpního souboru vkládá pouze znaky výstupního kódování
-v file	výstupem je i soubor VPL
-V file	výstupem je i soubor VPL a vytváří kapitálky

Příklad: Postscriptový font ITC-GaramondBoldCondensed, soubory pgbc.afm a pgbc.pfb v AdobeStandardEncoding, font obsahuje ligatury a některé akcentované znaky, má samostatné znaky pro háček, čárku a kroužek. Chceme jej připravit pro sazbu v TEXu:

```
afm2tfm pgbc -t xl2.enc -v pgbc rpgbc
```

Získali jsme soubor pgbc.vpl, kde jsou přemapovány znaky podle kódování xl2.enc (najdeme ho v DATA\A2AC) a soubor rpgbc.tfm v původním kódování a bez ligatur a kerningů (r znamená raw). Převedeme soubor VPL na VF:

```
vptovf pgbc.vpl pgbc.vf
```

Vzniknou soubory pgbc.vf a pgbc.tfm. Soubor pgbc.tfm obsahuje ligarury i kernigové informace, naopak od souboru rpgbc.tfm. Zkopírujeme soubory do správných adresářů a do psfonts.map přidáme řádek:

```
rpgbc ITC-GaramondBoldCondensed <pgbc.pfb
```

Nyní použijeme DVIPS a v GhostView vidíme výsledek. Sazba je kvalitní, ligatury fungují a *některé české znaky,* které byly v původním fontu se objevují také správně (většinou jde o písmena s čárkou a š, Š, ž, Ž), ale některé akcentované znaky chybí. Tento nedostatek vyřešíme programem A2Ac níže.

Afm2tfm nám nabízí možnost změny kódování postscriptové metriky; potom můžeme odkazovat ve virtuálním fontu na znaky s kódem nad 255 (někdy tam lze najít např. ligaruty fi, fl, česká písmena ž, Ž, polské Lslash. K vlastnímu překódování musí dojít až při převodu do Postscriptu, a proto je to nutné DVIPS sdělit a dodat příslušný kódový soubor. Do souboru psfonts.map zapíšeme opět poslední řádek výpisu Afm2tfm:

```
afm2tfm pgbc -p xl2.enc -v pgbc rpgbc
rgbc ITC-GaramondBold "XL2encoding ReEncodeFont" <xl2.enc <pgbc.pfb
```

Volba -V je pro tvorbu kapitálek:

```
afm2tfm pgbc -t xl2.enc -V pgbc rpgbc
afm2tfm pgbc -t xl2.enc -c .50 -V pgbc rpgbc
```

Standardně jsou malá písmena kapitálek 80 % velikosti velkých, tento poměr se dá měnit parametrem -c. Pro vytvoření rozšířeného nebo zúženého písma je parametr -e (zúžené < 1 < rozšířené), zápis do souboru psfonts.map musí informovat o naší změně (opět totožná zpráva s výpisem Afm2tfm. Skloněné písmo vytvoříme parametrem -s:

```
afm2tfm pgbc -t xl2.enc -e 1.6 -v pgbc rpgbc
afm2tfm pgbc -t xl2.enc -s .22 -v pgbc rpgbc
rpgbc ITC-GaramondBoldCondensed " 1.6 ExtendFont " <pgbc.pfb
rpgbc ITC-GaramondBoldCondensed " .22 SlantFont " <pgbc.pfb
```

Obrázek 17: *Afm2tfm — tvorba různých variant písma*

„Příliš žluťoučký kůň úpěl vysoké melodie.“

„PŘÍLIŠ ŽLUŤOUČKÝ KŮŇ ÚPĚL VYSOKÉ MELODIE.“

„Příliš žluťoučký kůň úpěl vysoké melodie.“

„Příliš žluťoučký kůň úpěl vysoké melodie.“

„Příliš žluťoučký kůň úpěl vysoké melodie.“

Příklady mají pouze znázornit možnosti programu; typografická úroveň algoritmicky rozšířených nebo skloněných variant písma je nízká a v praxi se doporučuje používat těchto deformací po velmi pečlivé úvaze.

Vidíme, že program Afm2tfm je opravdu silný nástroj. Stačí získat font a jeho metriku a můžeme využívat k sazbě systém TEX. Deformované řezy samozřejmě ve skutečnosti neexistují, DVIPS pouze pomocí postscriptových příkazů vytváří požadovaný vizuální efekt.

6.2.2. Ttf2tfm

Teprve nedávno se na CTANu objevil program srovnatelný s Afm2tfm pro fonty TrueType. Spolu s programem Ttf2pk tvoří dvojici, která může zajistit obsluhu TrueType fontů pro TEX. Ttf2tfm je součástí balíku nástrojů pro práci s fonty FreeType. Program je schopnostmi i syntaxí podobný Afm2tfm, ale bez možnosti transformace výstupu do obecného formátu. Soubor, kde jsou evidovány konvertované metriky, ttfonst.map slouží jako zdroj parametrů pro Ttf2pk a je složen z posledních řádků výpisu programu. Program má následující syntaxi a přepínače:

ttf2tfm file[.ttf] [options] [file[.tfm]]

-c real	použije real pro výšku kapitálek vytvořených -V (0.8)
-e real	rozšířené písmo, koeficient real (1.0)
-E int	volba encoding ID fontu (1)
-l	vytvoří první a druhý byte ligatur (pro korejská písma)
-n	použije postscriptová jména v TrueType fontu
-N	použije pouze Postscriptová jména, nepoužije cmap
-O	kódování znaků ve VPL souboru bude oktalové
-p encfile[.enc]	kódový soubor pro raw font
-P int	volba platform ID fontu (3)
-q	potlačení hlášení při výstupu
-r oldname newname	nahradí jméno znaku oldname za oldname
-R rplfile[.rpl]	označí soubor náhrad názvů znaků
-s real	skloněné písmo, koeficient real (0.0)
-t encfile[.enc]	kódový soubor pro VPL soubor
-T encfile[.enc]	ekvivalentní -p encfile -t encfile
-u	na výstupu budou pouze znaky z kódování
-v file[.vpl]	vytváří VPL soubor
-V scfile[.vpl]	jako -v, ale vytváří kapitálky
--help	vypíše informace o programu
--version	vypíše číslo verze programu

Program použije jedno z možných kódování v tabulce cmap TrueType fontu. Standardně je nastavena kombinace Microsoft/Unicode (platform/encoding popsáno v kapitole o TrueType fontech). Bude-li nastavena volba -N, budou ignorovány všechny údaje z tabulky cmap a program použije postscriptová jména v TrueType fontu; čerpá z tabulky post, která je povinná. Přepínač -n způsobí, že implicitní názvy vestavěné do Ttf2tfm jsou nahrazeny postscriptovými jmény znaků ve fontu. Tato volba není příliš vhodná, protože názvy jsou často nesprávné. Volba -l vytváří ligatury v subfontu mezi prvním a druhým bytem všech znakových kódů. Tato volba se používá pro korejské fonty.

Vstupní a výstupní kódování

Při konverzi je nutné stanovit dva kódovací vektory — jeden pro raw font a druhý pro virtuální soubor, podobně jako u programu Afm2tfm, ale zde se nám nabízí více možností. Můžeme adresovat přímo znaky v TrueType fontu pomocí desítkového, osmičkového nebo šestnáctkového zápisu:
/.c<decimal-number>, /.cO<octal-number> a /.cOx<hexadecimal-number>
Pro přímý přístup do glyph indexu můžeme použít zápis /.g místo /.c.

Další možností jak změnit kódování je použití přepínače -r, který provede záměnu starého názvu znaku za nový. Abychom šetřili příkazovou řádku, je vhodné tyto náhrady umístit do souboru .rpl a odkázat hodnotou parametru -R. Soubor náhrad obsahuje na každém řádku jednu náhradu (nový a nahrazovaný znak) oddělenou mezerou. Komentáře se značí znakem procento a uvozují komentář od místa výskytu až do konce řádky.

Pro dvojici pid/eid (1,0) a (3,1) program umí rozpoznat postscriptová jména znaků. Pokud nebude zvoleno vstupní kódování, bude prvních platných 256 znaků TrueType fontu namapováno podle zvolené tabulky cmap do TEX raw fontu a za ní budou následovat všechny ostatní znaky obsažené ve zvolené cmap. Vynechány budou znaky, které nemají význam pro TEX (null, backspace, horizontal tabulation, carriage return a group separator). Neplatné znaky s glyph index 0 budou také vynechány. S parametrem -N bude vybráno prvních 256 znaků s platným postscriptovým jménem.

Nebude-li zvoleno výstupní kódování, program použije mapovací tabulku pro typewriter. Prázdné pozice budou zaplněny znaky, které jsou ve vstupním kódování, ale ve výstupním použity nejsou. Volba -u způsobí, že budou zapsány pouze znaky ve výstupním kódování (prázdné pozice nebudou vyplněny).

Program Ttf2tfm má schopnost, která umožňuje sestrojení znaku Germandbls (ostré S). Tento znak se objeví na konci výpisu — řádek označený hvězdičkou. Lze jej používat pouze skrze virtuální font.

Pro obě kódování je prázdná pozice reprezentována názvem znaku .notdef. V kódovém souboru lze použít na konci řádku symbol \, který říká, že definice bude pokračovat i na dalším řádku.

Příklad 17: *Soubor náhrad RPL*

```
...
.c0x0041
A .c0x0042 B
.c0x0043 C
.c0x0044 D
.c0x0045 E
.c0x0046 F
.c0x0047 G
.c0x0048 H
.c0x0049
I ...
.c0x00b4 Iacute
.c0x00b5 Igrave
.c0x00b6 ocircumflexdotbelow
.c0x00b8 Itilde
.c0x00b9 Oacute
.c0x00ba uhornhookabove
...
```

Český font s kerningovými informacemi lze vytvořit pomocí virtuálního a raw fontu, které potom můžeme smazat a používat pouze TFM soubor vytvořený konverzí z VPL. Řádek v ttfonts.map samozřejmě nesmí obsahovat rtimes, ale pouze times. Soubor ttfonts.map je tvořen podobně jako u Afm2tfm posleními řádky výpisů Ttf2tfm.

```
ttf2tfm times -q -T xl2.enc -v times rtimes
vptovf times
del rtimes.tfm
del times.vf
ttf2pk times
```

Je nutné upravit soubor xl2.enc o řádek s informacemi o ligaturách, protože tato informace nemusí být obsažena v metrice fontu:
% LIGKERN f i =: fi ; f l =: fl ; a do ttfonts.map zapsat
times times.ttf Encoding=xl2.enc, místo varianty rtimes ....

Obrázek 18: *TrueType TimesNewRoman*

„Příliš žluťoučký kůň úpěl vysoké melodie.“
nabodeníčka: ě š č ř ž ý á í é ú ů ň ť ó ď
Ě Š Č Ř Ž Ý Á Í É Ú Ů Ň Ť Ó
pomocí příkazů TEXu: á Á č Č ť Ť ď ň
kerny a ligatury: ti ťi Tě Te fi fl ffi ffl – —
test interpunkce: ?!,.()’¿¡:;=+-&% *

TrueType font, který neobsahuje české znaky, ale obsahuje všechny kompozity, můžeme převést do kódování OT1 (sedmibitové CM fonty) a programem L2Accents vytvořit počeštěné virtuální fonty.

6.2.3. Program Af2pl

Existují programy (na CTANU jsou tři), které převádějí AFM soubor na property list. Všechny mají název Aftopl a využívají se k získání TFM souborů s kerningovými informacemi. Program Afm2tfm přidává kerningové informace pouze do virtuálního souboru, takže chceme-li získat úplnou metriku pro TEX, musíme postupovat přes virtuální property list nebo použít Aftopl. Pro DOS lze bez problémů zkompilovat pomocí GNU C verze pana Claytona M. Elwella, která pracuje se standardním vstupem a výstupem.

6.2.4. Program Accents a L2Accents

Program Accents (L2Accents) napsal Jiří Zlatuška. Má dvojí použití:

1. konverze postscriptové metriky v kódování Adobe Standard Encoding do osmibitového, českého fontu a současné *umístění akcentů* nad písmena.

2. konverze tradičního sedmibitového 128 znakového textového fontu v kódování TEXovských textových fontů do osmibitového virtuálního fontu. Virtuální font obsahuje na prvních 128 pozicích znaky v CM rozložení textových fontů.

Program pozná obě uvedené varianty automaticky. Nesloží však samozřejmě znaky, ke kterým nemá ve vstupním souboru komponenty — pracuje pouze na úrovni metrických informací. Umístění háčků a čárek dělá podle standardních definic maker \v a \', pokud není řečeno jinak ve speciálním (adjustačním) souboru. (SOJKA, 1994)

Obsahuje-li nečeský font potřebné akcenty, lze pomocí virtuálních fontů tento font „počeštit“. Jak bylo uvedeno výše, program Accents nám ušetří tuto ruční práci, ale jsou zde důležité podmínky: kódování musí být v **Adobe Standard Encoding** nebo **OT1!** Tady je vhodné upozornit, že ne všechny fonty Type 1, které je možné najít na internetu, mají toto kódování nebo obsahují tolik potřebný háček a čárku. Při prvních pokusech s počešťováním je toto jeden z důvodů neúspěchu a případně i následného „zatracení“ tohoto programu.

Syntaxe spuštění programu:

accents [<tfm> [<vf> [<adj>]]]

Pro seznámení s tímto programem je vhodné použít nějaké osvědčené fonty např. cmb10.tfm (TEXovské sedmibitové kódování) a třeba hvb.afm (Helvetica Bold v kódování Adobe Standard).

Font cmb10 najdeme v instalaci CSTEXu v adresáři TFM\CM. Tento font je v kódování OT1. Překopírujeme tento soubor do adresáře pro naše pokusy a převedeme tuto metriku příkazem:

l2accent cmb10.tfm

Máme tedy už tři soubory; původní metrika cmb10.tfm, vcmb10.tfm virtuální metrika (touto budeme sázet) a virtuální font vcmb10.vf. Po překopírování do příslušných adresářů vytvoříme testovací soubor a porovnáme český a počeštěný font.

Obrázek 19: *Porovnání výstupů csb10 a vcmb10*

Český text obsahuje háčky a čárky — csb10

Český text obsahuje háčky a čárky — vcmb10

Font hvb.pfb a hvb.pfm lze nalézt v každé instalaci AcrobatReaderu, metriku hvb.afm musíme získat buď na internetu, nebo si ji vygenerujeme sami, například programem GhostScript. K převodu na soubor TFM je vhodné použít program Af2pl, abychom neztratili kerningové a ligační informace, což by se v případě afm2tfm cob.afm stalo. Provedeme následující operace:

```
af2pl <hvb.afm >hvb.pl % získáme hvb.pl
pltotf hvb.pl hvb.tfm  % získáme hvb.tfm
l2accents hvb.tfm vhvb.vf vhvb.tfm % získáme virtuální font a metriku
hvb Helvetica Bold <hvb.pfb   % řádek v psfonts.map
```

Obdrželi jsme opět tři soubory (raw metriku, virtuální font a virtuální metriku). Překopírujme soubory do určených adresářů, připíšeme příslušný řádek do psfonts.map (DATA\DVIPS), přeTEXovaný soubor převedeme na Postscript a v GhostView uvidíme výsledek našeho snažení.

Obrázek 20: *Počeštění Helvetiky Bold prostřednictvím L2accents*

„Černý zuřivý pošťák zvoní u dveří.“

Sice již píšeme česky, ale estetický cit může urážet usazení akcentů některých písmen. Tuto vadu pomáhá odstranit adjustační soubor. Vhodnými příkazy lze definovat umístění nabodeníček — stejně jako ruční editací VPL.

Příklad 18: *Příklad adjustačního souboru hvb.adj*

```
(COMMENT Adjustacni soubor hvb.adj)
(COMMENT Neni nutne zadavat do prikazove radky,)
(COMMENT pokud je jmeno .adj shodne s .tfm)
(DESIGNUNITS R 10)
(CHARACTER O 322 (DOWN R .5) (RIGHT R .42)) (COMMENT - N-hacek)
(CHARACTER O 331 (DOWN R .36))        (COMMENT - U-krouzek)
(CHARACTER O 370 (RIGHT R .48))       (COMMENT - r-hacek)
(CHARACTER O 357 (RIGHT R .62))       (COMMENT - d-hacek)
(CHARACTER O 273 (RIGHT R .77))       (COMMENT - t-hacek)
(CHARACTER O 363 (UP R .2))          (COMMENT - o-carka)
```

Z příkladu je patrné, že zápis je stejný jako v souboru VPL, když si před příkazy pro umístění akcentů odmyslíme slovo MOVE. Tento příklad neslouží jako vzor pro typografickou úpravu, ale pouze pro ilustraci možností programu L2Accents. Ladění je poněkud zdlouhavé a je nutné nezapomenout žádný z šesti kroků — změna souboru ADJ, program L2Accents, kopírování souborů, TEXovská kompilace, převod DVIPS, prohlédnutí v GSView. Dalším doporučením je přesměrovat výstup L2Accents do nějakého souboru, protože nám „utečou“ chybová hlášení o ADJ souboru, např. l2accents hvb >a.log. Při používání nového fontu můžeme shledat, že některé znaky neobsahuje. Pak je nutné ručně editovat VPL soubor a font si doladit.

Na závěr by bylo vhodné uvést rozdíl mezi Accents a L2Accents. Rozdíl je ve výstupním kódování. Accents produkuje EC neboli DC neboli Cork kódování; výstup L2Accents je v kódování CSFontů neboli ISO-Latin-2.

6.2.5. Fontinst

Autorem tohoto programu je Alan Jeffrey. Fontinst je množina TEXovských maker, které umožňují instalaci virtuálních fontů. Balík může konvertovat fonty z AFM nebo property listu na virtuální property list. Fontinst má následující vlastnosti:

- program je napsán v TEXu, takže teoreticky může fungovat všude, kde pracuje TEX
- podporuje kódování OT1 (Computer Modern) a T1 (Cork), ale umožňuje generovat fonty v libovolném kódování
- může vytvářet kapitálky
- umožňuje sdílení kerningových informací mezi znaky
- lze připravovat matematické fonty
- do jednoho TEXovského fontu můžeme zařadit znaky z více Postscriptových fontů
- automaticky generuje FD soubor pro LATEX

Program Fontinst umí všechno, co lze implementovat pomocí virtuálních fontů. Prakticky vše co nám nabízí, jsme už dříve zrealizovali výše uvedenými programy (kromě tvorby FD souborů).

Fontinst pracuje s třemi druhy souborů:

- fontinst soubory, které obsahují příkazy pro instalaci fontu (např. fonttime.tex)
- soubory s kódováním (kódovací tabulky, ligatury, údaje o rozměrech fontu)
- metrické soubory (soubory s příponou mtx)

1. Fontinst soubory

Fontinst soubor je každý soubor, do kterého vstupují makra balíku Fontinst. Možné jsou následující příkazy:

\installfonts
install commands
\endinstallfonts

Posloupnost příkazů *install commands* instaluje rodinu písma.

Install commands:

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

Vytváří LaTeXovskou rodinu písma v daném kódování. Instalace rodiny Times v kódování Cork se provede: \installfamily{T1}{ptm}{}

\installfont{*font-name*}{*file-list*}{*ext*}{*encoding*}{*family*}{*series*}{*shape*}{*size*}

Příkaz produkuje TeXovský virtuální font nazvaný *font-name* z *file-list*u, který může obsahovat AFM, MTX nebo PL soubory s požadovaným zvětšením (scaled). Každý AFM soubor je překonvertován na raw PL soubor. Výsledný TeXovský font je kódován podle specifikovaného *etx* a může se k němu přistupovat z LaTeXu pomocí *encoding, family, series* a *shape*; *size* je buď deklarována \declaresize nebo specifikována ve FD.

```
\installfont{ptmrq}{ptmr0, latin}{T1}{T1}{ptm}{m}{n}{}
\installfont{cmro10}{cmr10, cmmi10}{OT1o}{OT1}{cmr9}{m}{n}{<10> <10.95>}
```

Takto lze instalovat Times Roman v kódování Corku, nebo vytvořit verzi cmr10 s číslicemi old style digits.

\substitutesilent{*from*}{*to*}
\substitutenoisy{*from*}{*to*}

Deklarace LaTeXovské substituce fontů, lze nahradit series a shape (*from*) jiným series a shape (*to*). Druhé makro pracuje stejně, ale při substituci zobrazí varování.

```
\substitutenoisy{sl}{it}
\substitutesilent{bx}{b} %nahradí bold extended tučným písmem
```

\transformfont{*font-name*}{*transformed font*}

Vytváří transformovaný raw font; např. rozšířený, skloněný nebo překódovaný. Jak bude tato tranformace provedena, záleží na výstupním zařízení.

```
\transformfont{ptmrol}{
    \slantfont{167}{
        \reencodefont{latin1}{
            \fromafm{ptmr0}
}}}
```

Do souboru psfonts.map pro DVIPS zapíšeme:

ptmrol Times-Roman " .167 Slant Font ISOLatin1Encoding ReEncodeFont"

\declaresize{*size*}{*fd-size-range*}
\declareencodiing{*string*}{*ext*

Příkaz deklaruje novou velikost a vytvoří příslušné FD příkazy, druhý určuje, který ETX soubor odpovídá danému kódování.

```
\declaresize{10}{<10>}
\declaresize{11}{<10.95>}
\declaresize{17}{<17.28>}
\declareencodiing{EXTENDED TEX FONT ENCODING - LATIN}{T1}
\declareencodiing{TEX TEXT}{OT1}
```

2. Soubory s kódováním

Kódovací soubor, má příponu .etx, je TEXovským dokumentem, který má následující strukturu:

\relax
ignored material
\encoding
encoding commands
\endencoding
ignored material

Encoding commands:

\nextslot{*number*} nastaví číslo následujícího slotu, není-li definováno bude jím následující

\setslot{*glyph*}
slot commands %popis znaku
\endsetslot

\inputetx{*file*} načte soubor s *encoding commands* nebo soubor ETX

Encoding proměnné:
boundarychar slot krajního znaku
fontdimen(*n*) rozměr znaku *n*
letterspacing dodatečná mezera, která bude přidána mezi každé znaky
codingscheme kódování fontu

Slot commands:

\usedas{*type*}{*control sequence*} nastaví řídící sekvenci pro daný slot

\nextlarger{*glyph*} nastaví položku nextlarger podle daného slotu

\varchar
varchar commands % nastaví položku varchar
\endvarchar

Varchar commands:

\vartop{*glyph*} nastaví horní *glyph*
\varmid{*glyph*} nastaví prostřední *glyph*
\varbot{*glyph*} nastaví dolní *glyph*
\varrep{*glyph*} nastaví opakující se *glyph*

3. Metrické soubory

Metrický soubor (přípona .mtx) je TEXovský soubor, který má následující strukturu:

\relax
ignored material
\metrics
metric commands
\endmetrics
ignored material

Metric commands:

\setglyph{*name*}
glyph commands
\endsetglyph

Pokud znak se jménem *name* není definován, bude použita nová definice.

\resetglyph{*name*}
glyph commands
\endsetglyph

Definuje znak bez ohledu na to, byl-li již definován.

\unsetglyph{*name*} znak *name* bude nedefinovaný.

\setrawglyph{*name*}{*font*}{*dimen*}{*integer*}{ vyzinteger}{*integer*}{*integer*}{*integer*}

Nastaví znak *name* z *font*u s parametry: velikost, slot, délka, výška, hloubka a kurzívní korekce. Tento příkaz je obvykle generován automaticky z AFM nebo PL souboru.

\setnotglyph{*name*}{*font*}{*dimen*}{*integer*}{ vyzinteger}{*integer*}{*integer*}

Pracuje stejně jako \setrowglyph s tím rozdílem, že jde sice o znak, který se vyskytuje v AFM souboru, ale není v default encoding. Slot nastaví na -1. Příkaz je generován automaticky z AFM souboru.

\setkern{*glyph*}{*glyph*}{*integer expression*}

Definuje kern mezi znaky; zvětšený o aktuální hodnotu rawscale, pokud byla nastavena.

\setleftkerning{*glyph*}{*glyph*}{*integer expression*}

\setrightkerning{*glyph*}{*glyph*}{*integer expression*}

První znak převezme pravý nebo levý kerning od podle druhého znaku s určitým zvětšením.

\setleftkerning{aacute}{a}{1000} *aacute* bude mít stejné kerny jako *a*.

\inputmtx{*file*}

Načte metrické příkazy *metric commands* ze souboru *file*.mtx.

Metrické proměnné:

ascender výška nejvyššího malého písmene
capheight výška nejvyššího velkého písmene
descender hloubka nejvyšší verzálky

italicslant poměr kurzívní korekce
minimumkern každý kern menší než tato hodnota bude ignorován
monowidth tato proměnná je nastavena u fontů neproporcionálních
underlinethickness síla podtržení
xheight výška písmene *x*
gliph-spacing mezera mezi písmeny
disignsize velikost fontu

Glyph commands:

\glyph{*glyph*}{*integer expression*} definuje znak v určitém zvětšení (implicitně 1000)

\glyphrule{*integer expression*}{*integer expression*} nastavení čáry dané délky a výšky

\glyphspecial{*text*} předá povel *text* výstupnímu zařízení.

\glyphwarning{*text*} při každém použití znaku bude zobrazeno varování *text*.

\movert{*integer expression*} a \moveup{*integer expression*} posun doprava nebo nahoru o hodnotu *integer expression*.

\push
glyph commands
\pop

Vykoná *glyph commands* bez posunu aktuálního bodu sazby.

\glyphpcc{*glyph*}{*integer expression*}{*integer expression*}

Zkratka pro:

```
\push
\movert{...}
\moveup{...}
\glyph{...}{1000}
```

\resetwidth{*integer expression*}
\resetheight{*integer expression*}
\resetdepth{*integer expression*}
\resetitalic{*integer expression*}
\samesize{*glyph*}

Nastaví délku, výšku, hloubku a kurzívní korekci pro aktuální znak nebo všechny rozměry podle znaku *glyph*.

Příklad 19: *Fontinst příklady:*

```
\setglyph{fi}
 \glyph{f}{1000}
\glyph{i}{1000}
 \endsetglyph

 \setglyph{moje_č}
  \glyphspecial{Filename: moje_č.eps}
 \endsetglyph

 \setglyph{onehalf}
  \moveup{500}
```

```
 \glyph{one}{700}
 \moveup{-500}
 \glyph{slash}{1000}
 \moveup{-200}
 \glyph{two}{700}
 \moveup{200}
\endsetglyph

\resetglyph{A}
 \movert{25}
 \glyph{A}{1000}
 \movert{25}
 ...
```

Závěr: Balík fontinst byl patrně programován se snahou komplexně postihnout a řešit maximum problémů při instalaci postscriptových fontů do TEXu. Jeho problémem bude patrná složitost a jistá „upovídanost“. Pro české podmínky zatím chybí příslušný soubor xl2.etx s definicemi českých znaků.

6.2.6. Cspsfont

Program Cspsfont napsal Zdeněk Wagner a slouží automatizovanému vytváření FD (Font Description) souborů pro LATEX. LATEX má velkou výhodu v tom, že lze v jednom dokumentu používat současně fonty s různým kódováním. Zavedení počeštěných fontů do LATEXu vyžaduje dvě akce. Především je nutno vytvořit virtuální skripty fontů v požadovaném kódování. Druhým krokem je vytvoření souboru s popisem fontu. Tyto soubory jsou velmi podobné, proto je lze snadno vytvářet automaticky. To je důvod proč byla vytvořena utilita Cspsfont. (WAGNER, 1998)

Je nutné vytvořit soubor, kde budeme definovat jak má být vytvořen a co bude obsahovat soubor s popisem fontu. Makra jsou uložena v souboru cspsfont, takže na začátek zadáme: \input cspsfont.

Globální příkazy:
\Author jméno autora se zapisuje do generovaných souborů
\Comment text komentáře by měl být pokud možno stručný
\FontPath definuje adresář, kde se nacházejí metriky fontů

Prostředí pro zápis definice fontu

fd Jeden soubor může být použit pro generování více souborů současně. Definice prostředí slouží k vymezení začátku a konce:

\begin{fd}{*kód*}{*rodina*} Příkazy pro definici fontů \end{fd}

PkgWrite Prostředí pro zápis příkazů, pro něž nemáme symbolický skript.

\exclam Makro pro vykřičník. Tento znak (!) je používán v případě, že chceme, aby nějaké makro expandovalo, protože v prostředí PkgWrite expanze neprobíhá.

Prostředí definice fontu

fontdecl Prostředí uzavírá popis vlastností fontu.

\begin{fontdecl}[*default*]{*base name*}[*suffix base*] Definice fontu \end{fontdecl}

Standardní font, kterým se nahradí nenalezená kombinace series/shape, určuje parametr *default*; *base name* je základní část jména souboru s metrikou fontu; *suffix base* je přípona, standardně je suffix prázdný.

\SeriesSuffix a \ShapeSuffix každému \fontseries a \fontshape musíme přidat písmeno nebo skupinu písmen.

\FontDef je speciální definice series/shape, které přiřadí určitý soubor.

\FontSub definuje substituci pro kombinaci series/shape.

\SeriesAlias nahradí series zadaným alias

BasBX a BXasB určuje, zda budeme series bold považovat za series bold extended nebo obráceně.

\MakeSeries a \MakeShape uvádějí seznam series a shape, pro něž se mají vytvářet definice fontu.

\FamilySetup a \ShapeSetup makra načítají a uschovávají příkazy, které se provedou vždy při zavedení rodiny písma nebo fontu. Více objasní následující příklady:

Příklad 20: *FD soubor fontu Bookman*

```
\ProvidesFile{XL2pbk.fd}[1996/01/05\space CS\space Bookman]
\typeout{\space CS\space Bookman}
\typeout{(C)\space Z.\space Wagner, 1996/01/05}
\typeout{Created\space by\space CsPsFont\space utility\space v.1.0d\space
(C)\space Z.\space Wagner,\space 1995/12/02}
\DeclareFontFamily{XL2}{pbk}{}
\DeclareFontShape{XL2}{pbk}{m}{n}{<-> pbkl8z}{}
\DeclareFontShape{XL2}{pbk}{m}{it}{<-> pbkli8z}{}
\DeclareFontShape{XL2}{pbk}{m}{sc}{<-> pbklc8z}{}
\DeclareFontShape{XL2}{pbk}{bx}{n}{<-> pbkd8z}{}
\DeclareFontShape{XL2}{pbk}{bx}{it}{<-> pbkdi8z}{}
\DeclareFontShape{XL2}{pbk}{bx}{sc}{<-> pbkdc8z}{}
\DeclareFontShape{XL2}{pbk}{b}{n}{<-> pbkd8z}{}
\DeclareFontShape{XL2}{pbk}{b}{it}{<-> pbkdi8z}{}
\DeclareFontShape{XL2}{pbk}{b}{sc}{<-> pbkdc8z}{}
\DeclareFontShape{XL2}{pbk}{m}{sl}{<-> sub * m/it}{}
\DeclareFontShape{XL2}{pbk}{b}{sl}{<-> sub * b/it}{}
\DeclareFontShape{XL2}{pbk}{bx}{sl}{<-> sub * bx/it}{}
```

6.2.7. Program A2Ac

Petr Olšák napsal program A2Ac (Afm To Afm add Composites), který podobně jako Accents přidává akcentované znaky do fontu, ale o úroveň výše — do souboru AFM.

Program umožňuje sázet texty postscriptovým fontem v jazycích, ve kterých se používají akcentované znaky. Výchozí fonty nemusejí obsahovat celou abecedu daného jazyka; stačí, když jsou přítomny jednotlivé akcenty samostatně. Konfigurační soubory programu A2Ac jsou nezávislé na kódovaní postscriptového fontu i na kódování, které je použito sázecím systémem. Program vytváří aleternavivu k programu Fontinst od Alana Jeffreyho. (Olšák, 1996)

A2Ac načítá AFM soubor, přidává nové kompozitní znaky a kerningové informace podle definičního souboru a vytváří nový AFM soubor. Výhodou tohoto

programu oproti Accents je jeho nezávislost na kódování; program pracuje se symbolickými názvy znaků (rcaron, aacute, ...). V instalaci EmTEXu je program A2Ac v adresáři BIN, definiční soubor cscorr.tab a kódovací soubory jsou v DATA\A2AC.

Použití programu:

```
a2ac input.afm corr.tab output.afm
```

S novou metrikou AFM souboru (output.afm)můžeme postupovat již známou cestou konverze na TEXovskou metriku pomocí Afm2tfm.

Můžeme se vrátit k příkladu v kapitole o Afm2tfm. Font Garamond se nám podařilo převést do prostředí TEXu a dvi soubor konvertovat do Postscript, ale některé české akcentované znaky font neobsahoval. Pomůžeme si tedy novým nástrojem následovně:

```
a2ac pgbc.afm cscorr.tab cpgbc.afm >a.log
```

Prohlédneme si a.log, kde jsou zaznamenány změny v počeštěném AFM oproti původní metrice; zkontrolujeme nový AFM soubor, případně doplníme ligatury fi a fl a pokračujeme stejně jako v předchozím příkladě:

```
afm2tfm cpgbc.afm -t xl2.enc -v cpgbc rpgbc
vptovf cpgbc.vpl
```

Zkopírujeme cpgbc.tfm, rpgbc.tfm a cpgbc.vf do příslušných adresářů, nezapomeneme dopsat příslušné informace do psfonts.map:

```
rpgbc ITC-GaramondBoldCondensed <pgbc.pfb
```

Obrázek 21: *Český ITC-GaramondBoldCondensed*

„Příliš žluťoučký kůň úpěl vysoké melodie.“
nabodeníčka: ě š č ř ž ý á í é ú ů ň ť ó ď
Ě Š Č Ř Ž Ý Á Í É Ú Ů Ň Ť Ó
pomocí příkazů TEXu: á Á č Č ť Ť ď ň
kerny a ligatury: ti ťi Tě Te fi fl ffi ffl – —
test interpunkce: ?!,.()’¿¡:;=+-&% *

Definiční soubor:

Všechny změny prováděné v souboru AFM jsou uveny v definičním souboru. Autor programu doporučuje vycházet z dodávaného souboru cscorr.tab. A2Ac pracuje s jakýmsi vlastním jazykem — můžeme definovat proměnné, volat funkce, provádět výpočty.

1. Definice proměnných má následující formát >> Nazev = vyraz; cscorr.tab obsahuje např.:

```
>> Captop = b(l,4)
>> distance = 0.0916 Captop
>> Acutetop = Captop + h(acute) + distance
>> Carontop = Captop + h(caron) + distance
>> acutewidth = w(acute)
```

```
>> acuteshift = 0.1 acutewidth
>> Acuteshift = 0.19 acutewidth
>> vshift = 0
>> diershift = b(acute,4) - b(dieresis,4)
>> Ccorrection = 0.1 w(C)
>> ccorrection = 0.05 w(c)
>> quotwidth = w(quoteright)
>> quotshift = .1quotwidth-b(quoteright,1)
```

2. Funkce jsou předem definovány a pouze se volají.

b(znak,i)	vrací hodnotu BoundingBoxu znaku; i nabývá hodnot 1–4.
b(znak,1)	levé dolní *x*
b(znak,2)	levé dolní *y*
b(znak,3)	pravé horní *x*
b(znak,4)	pravé horní *y*
w(znak)	šířka znaku; b(znak,3) – b(znak,1)
h(znak)	výška znaku; b(znak,4) – b(znak,2)
W(znak)	hodnota WX; délka znaku, posun bodu sazby
k(znak1,znak2)	hodnota kernu dvou znaků

3. Definice kompozitních znaků

NC New Composite, nový kompozitní znak; použije se pokud není v originální metrice definován

RC Rewrite Composite, nový kompozitní znak; přepíše původní definici kompozitního znaku

!C Nový kompozitní znak; přepíše původní definici i je-li znak definován jako nekompozitní

PCC znak x y Položí část znak kompozitního znaku posunutý od počátku o x a y

PAC znak x y Posun vůči ose základního (prvního) kompozitu

PAT znak x y y udává horní bod usazované části, posun po ose *x* vůči ose základního kompozitu

PCT znak x y y je horním okrajem usazované části, posun po ose *x* vůči počátku

```
NC Aacute 2 ; PCC A 0 0 ; PAT acute Acuteshift Acutetop ;
RC Zcaron 2 ; PCC Z 0 0 ; PAT caron 0 Carontop ;
NC ncaron 2 ; PCC n 0 0 ; PAC caron 0 vshift ;
```

Definice písmene Á (Aacute) se skládá ze dvou částí; na počátek se položí základní část A; bod sazby čárky (acute) se posune o hodnotu Acuteshift po ose *x* vůči počátku a horní bod na ose *y* bude roven Acutetop. Znak ncaron (ň) se vytvoří položením písmene n na počátek a háčkem (caron), který bude mít posunuty obě souřadnice vůči počátu o 0 a vshift.

4. Korekce parametru WX

Parametr WX v AFM souboru udává posun sázecího bodu pro každý znak a pro TEX se stane velikostí boxu písmene (tento údaj nelze vypočítávat z hodnot BoundingBoxů). Pro nové kompozitní znaky se WX stanovuje podle základní části. Určitá korekce WX je vhodná např. u písmen ť a ď.

RWX znak vyraz

RWX dcaron W(d)+0.7quotwidth
RWX tcaron W(t)+0.7quotwidth

WX písmene dcaron bude mít hodnotu součtu WX písmene dcaron a 70 % quotwidth, podobně je tomu s písmenem tcaron a lcaron.

5. Snížení počtu kerningových informací

ReduceKerns vyraz způsobí vymazání kerningových informací ≤ hodnotě vyraz. Je vhodné použít tuto funkci na začátku a na konci definičního souboru. Některé kerningové informace jsou tak malé nebo nulové, že jejich význam je také příliš malý nebo nulový.

V cscorr.tab je použito ReduceKerns 9.

6. Vytváření nových kerningových informací

NK první druhy vyraz New Kern, nový kern bude použit pokud tato kerningová informace neexistuje.

RK první druhy vyraz Rewrite Kern, nový kern bude použit bez ohledu na existenci této kerningové informace.

Definice kerningových informací má širokou paletu možností, využívá masek a výčtů. Princip lze snadno pochopit na příkladech:

NK á b 13 — kerning mezi písmeny bude 13

NK * tcaron : * t — všechny znaky budou mít stejný kerning s písmenem tcaron jako s písmenem t.

NK tcaron * : f * -.3quotwidth — tcaron bude mít stejné kerningové informace jako písmeno f, ale zmenšené o 30 % quotwidth.

RK (F,P,T,V) zcaron 0 — písmena F, P, T a V budou mít s písmenem zcaron nulový kerning.

NK Zcaron : Z — Zcaron a Z budou mít stejné kerningové informace

Vidíme, že A2Ac je účinný nástroj k přípravě české metriky AFM. Důležitou podmínkou je, aby výchozí metrika obsahovala všechny znaky pro sestavení znaků kompozitních. Jedná se o breve, dotaccent, ring, hungarumlaut, cedila, caron a dotlessi. Některé asi nebudeme pro českou sazbu potřebovat (hungarumlaut, breve, cedila), ale nemusíme vždy připravovat ryze český nebo slovenský font.

6.3. Získání bitmap

Máme-li instalaci TEXu, pak většinou neobsahuje pouze samotný program TEX, ale i řadu ovladačů a pomocných programů. Například instalace EmTEX obsahuje ovladače pro obrazovku i pro tiskárny. Pokud se v instalaci nachází ovladač, který lze používat na našem stroji, pak můžeme vidět výsledky své práce bez různých konverzí a nastavování rozličných řídících parametrů.

Získávání bitmap (soubory PK) je mnohem komplikovanější záležitost, než získávání metrik. Zatímco metrika AFM je velmi podobná složení TFM, pak obrysový font Type 1 nebo TrueType nemá se souborem PK mnoho společného.

Otázkou zůstává zdůvodnění potřeby těchto bitmap. PK bitmapy, metriky TFM a jejich metafontové zdroje spolu s METAFONTem tvoří přirozené prostředí TEXu. Všechny ovladače umí s těmito formáty dat pracovat a jsou implicitně očekávány i samotným TEXem (Computer Modern). Na druhou stranu jiné fonty mají své přirozené prostředí zcela jinde a TEX, jak bylo již řečeno, stejně potřebuje pouze metrické informace. Není potom lepší sestavit svůj dokument a nechat výstupní zařízení, aby se samo vyrovnalo s problematikou rastrování? Například převod do Postscriptu a následný tisk na postscriptovém zařízení?

Důvodů pro získávání bitmap může být celá řada. Například nejsme vlastníky postscriptové tiskárny nebo chceme pouze předběžné kontrolovat, jak se nám daří tvořit dokument.

6.3.1. PS2PK

Program PS2PK rastruje postscripttový font Type 1 na TEXovský font PK. Autorem je Piet Tutelaers a v archívu CTAN existuje již řada verzí jak ve zdrojových kódech, tak kompilovaných spustitelných verzí. PS2PK je součástí balíku, který by měl zvládnout kompletní obsluhu postscriptových fontů v naší instalaci TEXu. K převodu potřebujeme soubor PFB i AFM. Syntaxe programu:

pk2pk [options] type1 [pkfont]

Options:

-aAFMfile určí jméno AFM souboru
-eenc změna kódování oproti AFM souboru, například xl2.enc
-Eextension rozšíření písma (1.0)
-Ppointsize požadovaná velikost v bodech (10.0) pt
-Sslant sklon písma
-Xxres rozlišení v horizontálním směru (300 dpi)
-Yyres vertikální rozlišení (xres)
-O vytváří starou formu kontrolních součtů
-v program říká co právě provádí
type1 název Postscriptového fontu, akceptuje PFB i PFA, předpokládá i AFM
[pkfont] název výstupního PK souboru

PS2PK navazuje na program Afm2tfm, proto pokud generujeme skloněné nebo rozšířené varianty písma, měly by parametry -E a -S odpovídat těm hodnotám parametrů, které jsme použili při přípravě metrik. Podobně kódování by mělo být stejné jako u metriky, pokud používáme virtuální fonty, pak stejné (tedy žádné) kódování jako raw font. Všechny tyto hodnoty parametrů jsou zaznamenány v psfonts.map, pokud je tam přidáváme.

Obrázek 22: *Rastrování Garamond-Bold PS2PK (navazuje na Afm2tfm a A2Ac)*

„Černý zuřivý pošťák zvoní u dveří.“

„Černý zuřivý pošťák zvoní u dveří.“

„Černý zuřivý pošťák zvoní u dveří.“

„Černý zuřivý pošťák zvoní u dveří.“

„Černý zuřivý pošťák zvoní u dveří.“

Chceme-li pracovat s PS2PK, je vhodné z archívu stáhnout celý balík, protože obsahuje program mkpsres, který vytváří databázi fontů psres.dpr. S touto databází spolupracuje jak PS2PK, tak i ostatní programy v balíku. Program někdy odmítá převést některý font a vypisuje soubor core. Někdy stačí změnit název souboru, například zkrátit, nebo načíst do textového editoru a zase uložit.

Různé prameny uvádějí, že PS2PK generuje bitmapy, které jsou položeny o jeden bod výše, než účaří. Toto lze pozorovat, když položíme vedle sebe znaky vytvořené PS2PK a Computer Modern. Při zvětšení v prohlížeči DVISCR je tento jev patrný.

6.3.2. Ttf2PK

Program Ttf2PK tvoří společně s Ttf2tfm programovou dvojici pro konverzi TrueType fontů do prostředí TEXu. Oba programy jsou součástí balíku FreeType. Ttf2tfm získáváme potřebné metriky TrueType fontu a Ttf2PK zase bitmapy. Program Ttf2PK čte údaje z ttfonts.map a podle nich vytváří bitmapu. Použití:

```
ttf2pk [-q] [-n] FONT DPI
ttf2pk -t [-q] FONT
```

Volby:

-q potlačení vypisování zpráv na obrazovku
-n PK soubory budou generování s rozšířením .pk
-t testuje FONT, při úspěchu vrací 0
--help vypisuje stručný návod
--version vypíše verzi programu

Parametr FONT musí odpovídat položce v souboru ttfonts.map, jinak vrací chybový kód 2. Volba -t vypíše řádku z ttfonts.map, která odpovídá volbě FONT a vrací kód 0. DPI určuje cílové rozlišení. Program předpokládá velikost fontu 10pt, takže pokud chceme získat větší font, musíme změnit rozlišení. Vzorec pro výpočet tohoto rozlišení může být např.:

$$\mathrm{DPI} = \frac{\text{Velikost v pt}}{\frac{10}{\text{rozlišení}}}$$

Obrázek 23: *Bodoni.ttf ve formátu PK*

„Černý zuřivý pošťák zvoní u dveří."

„Černý zuřivý pošťák zvoní u dveří."

„Černý zuřivý pošťák zvoní u dveří."

„Černý zuřivý pošťák zvoní u dveří."

Programu není třeba sdělovat hodnoty parametrů pro rozšířené a skloněné varianty písma, jako je tomu u PS2PK. Posledním řádkem (řádky) výstupu Ttf2tfm jsou všechny hodnoty parametrů, které jsme použili pro vytvoření metriky ve formátu pro ttfons.map. Název fontu použité jako parametr FONT

při vytváření bitmap odpovídá TEXovské metrice a zároveň sdružuje údaje o názvu TrueType fontu a provedených transformacích.

6.3.3. GsftoPK

Původem Unixová utilita od Paula Vojty GsftoPK vytváří fonty formátu PK konverzí z Postscriptových fontů za pomoci programu GhostScript. V balíku je i dávka GetAfm.bat, která vytvoří potřebný AFM soubor pro další převod na metriku TFM. Použití:

gsftopk font-name dpi

font-name musí být krátký název postscriptového fontu, např. rptmr. Krátká jména užívaná v TEXu jsou v souboru psfonts.map, který používá program DVIPS.

Program používá postscriptový soubor render.ps, kde je naprogramována celá transformace a soubory pro vstup a výstup (gs.ps, gsin a gsout). Soubor gs.ps obsahuje údaje o celém názvu fontu, jeho umístění, požadovaném rozlišení a umístění souboru render.ps. Soubor gsin obsahuje údaj o velikosti bitmap a kódy znaků, které budou generovány.

GsftoPK ve verzi pro DOS je rozdělen do dvou souborů. Celá dávka pracuje tak, že první soubor (gsftopk1.exe) otestuje přítomnost potřebných komponent, vytvoří gs.ps a podle metriky TFM vytvoří soubor gsin. Pak je spuštěn GhostScript, který vytvoří soubor gsout, ten je zpacován gsftopk2.exe na PK font jméno.dpi.

Práce s tímto nástrojem v DOSu není tak snadná jako na Unixových systémech. Zatímco v OS Solaris program po nastavení systémových proměnných pracoval bez problémů, v DOSu tomu tak zdaleka není.

6.3.4. TTFtoGF

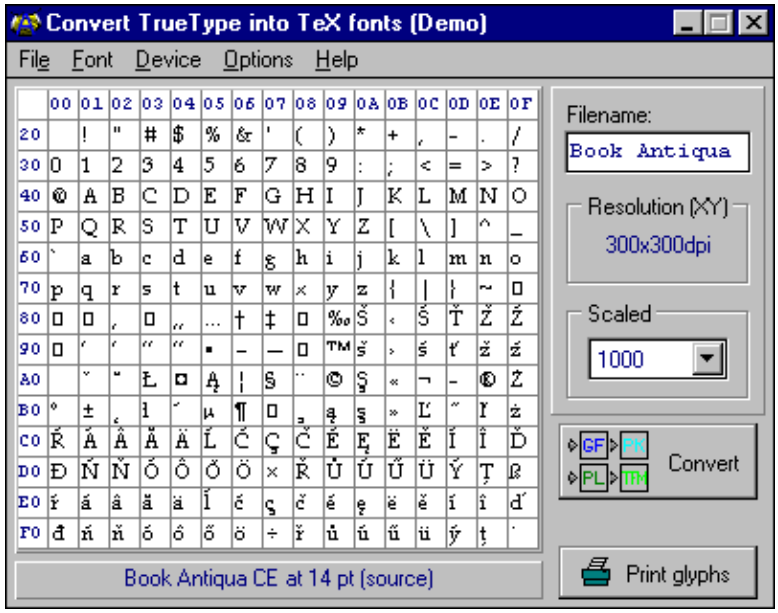
TTFtoGF je program, který umožňuje převod TrueType fontů do prostředí TEXu. Autorem je Konstantin Vasiľev. Stejně tak může konvertovat i Postscriptové fonty instalované v systému. Program vytváří soubory bitmapy GF a metriku PL, které odpovídají vybranému fontu. Po převodu na PK a TFM,lze fonty používat standardními DVI prohlížeči. Program běží v prostředí Windows 3.x a vyšší.

TTFtoGF načítá informace o fontu přímo ze systému a převádí znaky na bitmapy TEXovského formátu. Stejně získává metriky a kerningové informace. Programu lze nastavit, aby konverze na TFM a PK byla provedena automaticky — musíme umístit gftopk a pltotf do pracovního adresáře nebo do PATH.

Font, který chceme konvertovat se vybírá ve standardním systémovém Font Dialogu. Rozlišení je určeno předvolenou tiskárnou, takže pokud chceme jiné rozlišení než má instalovaná tiskárna v systému, je nutné přiinstalovat další s požadovanými vlastnostmi. V menu Options lze nastavit kódování fontu a cesty pro uložení PK a TFM. Další funkcí je možnost vytištění zkušební stránky a zvětšení výstupního fontu podle TEXovského parametru \magstep.

Program vytváří výstupní jméno souboru z názvu fontu, který nemusí být programy jako pltotfm akceptováno (obsahují mezeru), např. Book Antiqua bude reprezentován jménem Book An.pl.

Obrázek 24: *Program TTFtoGF*



Kódovací soubor je textový soubor s příponou CTF a každý řádek má následující formát:

c<source character code>=[<destination character code>] [-1]

Volba -1 znamená, že znak nebude ve výstupním GF souboru přítomen. Vstupní i výstpuní kódování musí mít hodnoty v rozmezí 0 až 255.

Příklad 21: *Kódový soubor TTFtoGF CTF*

```
[Encoding]
c123=-1
c124=-1

c128=-1
c129=-1
c130=-1
...
c192=128
c193=129
c194=130
...
c254=238
c255=239
```

TTFtoGf není volně šířitelný program, v archívech se vyskytuje pouze demo verze, která má omezení pro výstupní znakovou sadu.

6.4. Získávání metafontových souborů

Fonty formátu METAFONTu spolutvoří přirozené prostředí TEXu. Bohužel koncepce digitalizovaného písma v METAFONTu je zcela odlišná od myšlenky obrysových fontů. METAFONT také neobsahuje hintovací informace, které jsou samozřejmou součástí postscriptových a TrueType fontů. Přesto existují dobré důvody pro tento typ konverze. Získáme-li metafontový zdroj potřeb-

ného písma, můžeme kontrolovat vizuální podobu sázeného dokumentu bez nutnosti převodu do Postscriptu nebo dodatečného generování PK souborů.

6.4.1. PS4MF

PS4MF konvertuje Type 1 fonty na formát METAFONTu. Autorem je Markus Neteler. Type 1 fonty se vyskytují v binární nebo ASCII podobě. Binární podoba je kódovaná forma ascii formy. PS4MF čte úplně rozkrytou (rozkódovanou) podobu postscriptového fontu a vytváří metafontový zdroj. Při konverzi je PFB font nejdříve převeden do rozkryté ASCII podoby — program volá program T1disasm, který má jako vstup tento font. PS4MF načítá font.ps a font.afm a kódový vektor výstupního MF souboru. Převod Type 1 do METAFONTu podle PS4MF:

```
t1disasm font.pfb >font.ps
ps2mf -C -p font.ps -a font.afm -m font.mf -c encoding.cfg
```

Parametry:

-p následuje název rozkrytého ascii Type 1 fontu
-a určuje jméno příslušného AFM souboru
-m název výstupního metafontového souboru
-c výstupní kódování
-C umožní vytvoření metafontového souboru za použití tzv. „curls“, což je možnost měnit tvar křivky znaku v jeho koncových bodech (křivky se lépe „vlní“)

Program také volá Afm2tfm se zvoleným kódováním a kopíruje všechny soubory do příslušných adresářů (nutno upravit dávku).

Znaky, které obsahuje výstupní kódování a jsou definovány v PFB fontu jako kompozitní, obsahují v metafontové definici znaku řádek začínající postscriptovým operátorem seac (.... Tato nepodařená konverze definice znaků způsobí nejen jejich nevykreslení, ale i konec práce METAFONTu. Tyto řádky je třeba ručně umazat.

Bohužel toto není jediný problém. Spolu s programem je šířeno několik dávek a kódování pro různé národní uživatele TEXu. Dokonce se pod názvy conv_es.bat, conv_es2.bat, east.cfg a east2.cfg skrývají dávky a kódování pro „Czechoslovak TEX users“. Ani jedno kódování však neodpovídá xl2; chybějí znaky (např. Uring) nebo jsou špatně označeny (tquoteright). V balíku se nacházejí dávky pro konverze do dalších kódování.

Kvalita znaků vzrůstá podle hustoty výstupního rastru a velikosti znaků. Čím větší, tím kvalitnější. I přesto dochází k jistým nepřesnostem. Tady je potřeba zvážit důvod převodu do METAFONTu a účel připravované tiskoviny, protože různé poruchy tahů mohou způsobit zajímavé efekty nebo oživení písma.

Pro získání písem použitelných v českém a slovenském prostředí se nabízejí dvě cesty. První je překreslení nebo programová dekompozice kompozitních znaků, aby nedocházelo k chybám typu seac. Druhá cesta vede přes konverzi do kódování původních CM fontů k použití dalších nástrojů pro počeštění virtuálními fonty (Accents). První metoda předpokládá navíc úpravu chybného

kódového vektoru east.cfg podle xl2.enc. Syntaxe kódovacích konfiguračních souborů vychází z AFM formátu, takže je srozumitelná.

Obrázek 25: *Písmo Times, převedené PS4MF*

„Příliš žluťoučký kůň úpěl vysoké melodie.“
nabodeníčka: ě š č ř ž ý á í é ú ů ň ť ó ď
Ě Š Č Ř Ž Ý Á Í É Ú Ů Ň Ť Ó
kerny a ligatury: ti ťi Tě Te fi fl ffi ffl
test interpunkce: ? !,.()';=+&% *

6.4.2. TTF2MF

TTF2MF je program, který převádí Windows True Type fonty do formátu METAFONTu. Autorem je Oleg V. Motygin. Program je volně šiřitelný a lze ho bez poplatku používat pro nekomerční účely. Program pracuje pod systémem Windows 3.x, 9x a NT.

Po spuštění programu ttf2mf.exe je nutné vybrat font, který hodláme konvertovat: menu Font → Choose. Výběr fontu probíhá ve standardním font dialogu systému. Vybraný font se zobrazí v tabulce původního kódování. U TrueType Open je vhodné vybrat příslušný skript, např. středoevropský.

Pro vlastní převod zvolíme: Font → Convert. Před tímto úkonem je potřeba zvolit výstupní kódování: Encoding → Change encoding. Program bohužel nepoužívá názvy znaků, ale přímo kódy TrueType fontu. To sice nevadí při převodu spodní poloviny ASCII tabulky, ale v horní polovině není vždy stejné rozložení znaků, a tak nelze používat jeden kódový soubor pro všechny TrueType fonty. Při vytváření souboru kódování je nutné nahlédnout do TrueType souboru a zjistit umístění znaků a kódový soubor případně upravit.

Výsledkem je pouze metafontový soubor, proto je třeba před použitím vygenerovat metriku. V metafontovém souboru může být potřeba upravit následující položky:

1. font_unlin_height
2. font_normal_space
3. font_normal_stretch
4. font_normal_shrink

Tuto úpravu je vhodné provést podle chování fontu v TEXu.

Kódový soubor fontu

S programem je šířen kódový soubor 1251_866.enc, který může posloužit jako příklad. Na každém řádku je jedna položka kódovacího vektoru METAFONTu a jedna položka kódu znaku jaký má v TrueType fontu. Obecný formát řádky:

m [.. n] t

Znaky od kódu m do n ve výstupním kódování mají obraz v TrueTypu na pozici t. Bude-li mít t hodnotu −1, znak bude vynechán. Zápis lze provádět jak v decimálním, tak v oktalovém zápisu nebo smíšeném, podobně jako např. property listu. Změna kódování se provede i v okně programu.

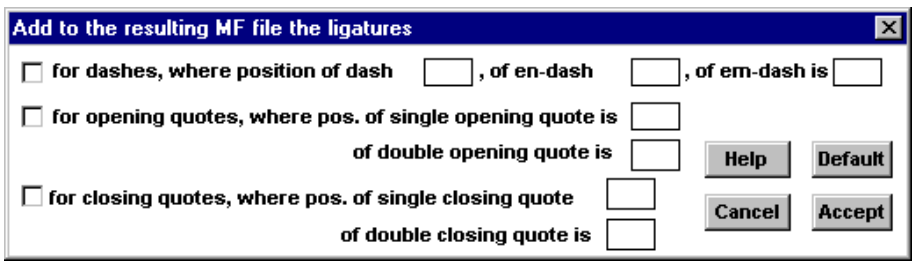
Příklad 22: *Kódový soubor pro TTF2MF*

```
000     148
001     138
002     143
004     175
005     170
006     128
007     142
...     ...
'340..'357  240
'360        168
'361..'362  184
'363        171
'364        187
```

Definice ligatur

TTF2MF umožňuje definici některých ligatur, konkrétně pomlček, otevíracích a uzavíracích uvozovek. Jde o určení, které ligatury budou ve výstupním souboru a polohy jejich složek v TrueType fontu. Definice jiných ligatur není možná v programu, ale lze ji provést až dodatečně, takže nemusíme být o nic ošizeni.

Obrázek 26: *Definice ligatur v programu TTF2MF*



Program nepracuje přímo s vlastním fontem, ale používá definované funkce systému (GetGlyphOutLine, ...), což je důvod proč pracuje pouze s instalovanými fonty. TTF2MF nezohledňuje ani hintovací informace.

Obrázek 27: *Bookman.ttf převedený pomocí TTF2MF*

„Příliš žluťoučký kůň úpěl vysoké melodie.“
nabodeníčka: ě š č ř ž ý á í é ú ů ň ť ó ď
Ě Š Č Ř Ž Ý Á Í É Ú Ů Ň Ť Ó
kerny a ligatury: ti ťi Tě Te fi fl ffi ffl
test interpunkce: ?!,.()'><:;=+-&% *
Ligatury a některá interpunkční znaménka nefungují!

V příkladech je pouze soubor booka.mf a booka.tfm. metafontový zdroj byl získán TTF2MF bez změny kódování a ručně byly provedeny změny kódů znaků š,ž,ť a uvozovek.

6.4.3. Fog2MF

Autor TEXu připravil program, který jak sám říká je „rychlý a špinavý švindl“ pro konverzi Fontographer Type 3 fontů do METAFONTu. Autor předpokládá, že vstupní soubor bude ručně sestaven z AFM a postscriptového výstupu Fontographeru. Pak bude ještě nutné ručně upravit metafontový soubor. První vstupní řádek je definice znaku v AFM souboru, pak následuje postscriptový zápis znaku. Nejlépe vše objasní příklad.

Příklad 23: *Vstup a výstup programu Fog2MF*

Vstup:

```
C 33 ; WX 220 ; N exclam ; B 34 442 187 664 ;
/exclam{220 0 4.0426 422.858 217.66 683.983 Cache
187.702 623.194 moveto
-0.297607 -29.4158 -111.106 -124.124 -24 -28 rrcurveto
-19 10 rlineto
16.42 45.8054 35.7433 110.536 23 36 rrcurveto
8.12329 12.7366 11.0454 6.84058 15.4363 -0.159424 rrcurveto
23.3605 -0.159424 21.7024 -18.0087 -0.297607 -23.8059 rrcurveto
closepath
0 FillStroke
}def
```

Výstup:

```
beginchar(33,220u#,664u#,0u#);
stroke (188,623)
 ...(187,594,76,470,52,442) --(33,452)
 ...(50,497,85,608,108,644)
 ...(117,657,128,664,143,663)
 ...(166,663,188,645,188,621) --cycle;
endchar;
```

Program se nachází na Knuthových WWW stránkách ve formátu cweb. Proto je pro kompilaci potřeba nejdříve programem ctangle získat kód v jazyku C a teprve potom se pokusit o překlad. V DOSu se podařilo kompilaci zvládnout GNU C. Fog2MF pracuje se standardním vstupem a výstupem.

Problematika širšího rozšíření Fontographerovských Type 3 fontů a pracnost přípravy vstupního a úpravy výstupního souboru odsouvá tento program na pokraj zájmů hromadného převádění fontů, bez ohledu na to, že Fontographer verze 3.5.2, který byl používán při testování a úpravách fontů, neměl možnost uložit výstup ve dříve zmiňovaném formátu Type 3.

6.4.4. Ega2MF a Vga2MF

Ega2Mf je utilita, která generuje kód METAFONTu z obrazovkových bitmap rozměru 8 × 14. Cílem programu je vytvoření fontů, které emulují výstup obrazovky počítačového displaye se všemi jeho nedostatky na příslušné výstupní zařízení. Pro převod jiných než EGA bitmap je třeba určitých úprav. Autorem je Thomas Ridgeway.

Vstupem je jméno souboru s bitmapami a výstupem je metafontový soubor. Program nekontroluje správnost vstupního souboru. Spolu s programy je šířeno i několik souborů s bitmapami z různých kódovacích stránek.

6.5. Utility pro práci s fonty

6.5.1. Qdtexvpl

Qdtexvpl je podle autora (Eberhard Mattes) nástroj pro rychlou a „špinavou“ tvorbu virtuálních fontů. Nejdříve vytvoříme TEXovský soubor. Po zpracování souboru TEXem podsuneme DVI soubor programu Qdtexvpl, jehož výstupem je soubor VPL. Ten převedeme na virtuální font a metriku.

V souboru, kde připravujeme virtuální font, musíme načíst soubor qdtexvpl.tex s makrem \texvpl. Makro má dva parametry: první je znak, který budeme používat pro označení kompozitního znaku a druhý TEXovská sekvence sestavení tohoto znaku.

Příklad 24: *Použití Qdtexvpl (test.tex)*

```
\font\f=cmr10
\input qdtexvpl
\texvpl{Ď}{\f\v D}
\texvpl{Ö}{\f\" O}
\texvpl{\ss{}}{\f\ss}
\texvpl{ä}{\f\" a}
\texvpl{È}{\f\' L}
\end
```

Po získání DVI souboru použijeme program Qdtexvpl. Použití:

qdtexvpl -d<disign_size> [-i<insert.pl>] <input.dvi> <outpout.vpl>
Pro náš příklad: qdtexvpl -d10 -icmr10.pl test.dvi novy.vpl

Příklad 25: *Získaný VPL soubor (novy.vpl)*

```
(MAPFONT D 16
  (FONTNAME cmr10)
  (FONTCHECKSUM O 11374260171)
  (FONTDSIZE R 10.000000))
(CHARACTER O 317
  (CHARWD R 0.763889)
  (CHARHT R 0.881250)
  (CHARDP R 0.000000)
  (MAP
    (PUSH)
    (MOVERIGHT R 0.131944)
    (MOVEDOWN R -0.252777)
    (SELECTFONT D 16)
    (SETCHAR O 24)
    (POP)
    (SETCHAR C D)))
...
```

Po převedení na VF a TFM, lze používat font obvyklým způsobem.

6.5.2. Bm2Font

Bm2Font sice není program, který by konvertoval mezi různými formáty fontů, ani není nějakou pomocnou „utilitkou“, která by v součinnosti s jinými *x* „utilitkami“ umožňovala používat neTEXovský font v TEXu, ale samostatným

programem na převod bitmap do TEXovských fontů. Program napsal Friedhelm Sowa. Bm2Font je dobře dokumentován např. v (RYBIČKA, 1997).

Program je schopný pracovat s mnoha rastrovými formáty, např. GIF, PCX, TIFF, BMP. Po zpracování dostaneme příslušný soubor PK, metriku TFM a soubor .tex, kde jsou definice maker pro zavedení obrázku do dokumentu.

Vidíme, že tento program pracuje s fonty pouze na jedné (výstupní) straně a proto pro něj asi těžko budeme hledat nasazení pro přípravu fontů TeXu. Může nám ale dobře posloužit, když potřebujeme do svého dokumentu pouze vložit nějaký nápis v exotickém písmu (např. jméno čínského hrdiny). Syntaxe programu:

```
bm2font soubor_obrázku parametry
```

Volby:

-f jméno výstupního souboru, není třeba udávat
-h -v horizontální a vertikální rozlišení (std 300 DPI)
-m -n šířka a výška výstupního obrázku v mm
-g příznak zda se v obrázku vyskytují odstíny šedé (varianty y n)
-w interpretace bílé jako světle šedé (varianty y n)
-l šířka řádku obrázku v bytech
-u-c velikost rastru; kolik bodů výstupního zařízení zobrazuje jeden bod obrázku
-t -z korekce světlosti; hodnota a rozsah v procentech (std 70 %)
-j ořízнutí bílého okraje (varianty y n)
-r opakování každého bodu (zvětšení výsledného obrázku)
-x počet odstínů šedé (std 256)
-s separace šedých bodů
-b redukce světlosti barevných obrázků (program dává nápovědu pro optimální hodnotu)
-d rozložení chyb (varianty y n)
-i inverzní obrázek (varianty y n)
-a zobrazení na obrazovce

Postup vložení obrázku do TEXu: Bitmapu si můžeme připravit v jakémkoli grafickém programu, který umí exportovat do formátů pro Bm2Font. Podle charakteru obrázku a požadovaného výstupu zvolíme vhodné parametry pro Bm2Font a provedeme konverzi. Překopírujeme výsledné soubory do adresářové struktury instalace TEXu a můžeme obrázek použít.

Bm2Font náš obrázek podle velikosti rozdělí na čtverce — písmena, které se při použití musí poskládat tak, aby byl složen původní obraz. Aby nebylo třeba zkoumat jak program obrázek rozdělil, vytvoří soubor *obraz.tex*, který obsahuje makro \setobraz pro vysázení tohoto obrázku. Problémem je, že toto makro při dalším použití obrázek nevysází. Lze to obejít tak, že si zavedeme do TEXu příslušný font a vysázením příslušných znaků (ty zjistíme v souboru *obraz.tex*) vysázíme obrázek:

```
\font\obr=obraza
\def\setobr{\obr\char0\char1\char2}
Poprvé \setobr a podruhé \setobr
```

Obrázek 28: *Použití Bm2font*

„Jakarimašikana wakarišika,“ zvolal Li-Wu a zemřel!
Pro umocnění dojmu z Li-Wuovi smrti můžeme použít:
„  ,“ zvolal Li-Wu a zemřel!

6.5.3. Pkbbox

Program Pkbbox je program pro získání použitelné metriky AFM pro Postscriptové fonty. Jeho činnost je zapojena do následující posloupnosti:

1. Vygenerování AFM souboru programem Pfm2Afm, který není úplný — chybí v něm údaje o BoundigBoxech znaků, ty obsahuje soubor PFB.

2. Programem PS2PK si vytvoříme PK soubor. PS2PK sice potřebuje AFM soubor pro svou práci, ale nezajímají ho BoundingBoxy, takže nevadí neúplnost našeho AFM souboru získaného Pfm2Afm.

3. Pkbbox, jak už napovídá jeho název, čte neúplný AFM a PK soubor a vytváří přiměřený AFM soubor s BoundigBoxy.

4. Nyní již lze použít Afm2tfm.

Program má dva nedostatky:

1. Podle autora nejsou BoudingBoxy stejné, které by obsahoval pravý AFM soubor.

2. Program nefunguje! Naprosto spolehlivě „zatuhne“ počítač. Nejen že zatuhne DOS, ale dokonce i celé Windows (ne pouze DOSovské okno)!

Bohužel v archívu je pouze zdrojový pascalovský soubor a chybí příslušné knihovny, takže nelze program opravit.

6.5.4. RUMgraph

Program RUMgraph převádí jednobarevné grafické soubory do TEXovského PXL souboru. Jsou podporovány dva formáty: PCX firmy ZSoftu a ADI od AutoDesku. Vstupem je jeden nebo více grafických souborů a metrické informace znaku. Výstupem je PXL soubor, který obsahuje kresbu znaku, TFM soubor a TEXovský soubor s příkazy vztahující se k danému fontu. Dalším výstupem je soubor .rgl s informacemi o průběhu transformace. Program napsal H. W. Kisker.

Spuštění programu:

rumgraph *options*

Options: parametry se uvozují znakem - nebo /; dále následuje písmeno volby, znak přiřazení (: nebo =) a hodnota parametru. Obsahuje-li hodnota parametru znak mezera je nutné uzavřít tuto hodnotu do uvozovek. Parametry jsou rozděleny do tří skupin:

1. Volby fontu

/P:*ident* volba specifikuje názvy souborů ident.PXL, ident.TFM, ident.TEX a ident.RGL.
/R:*resolution* udává rozlišení bitmap.

2. Volby znaků

/C:*character* tímto parametrem začíná množina voleb pro daný znak. Všechny následující volby druhé skupiny se vztahují k hodnotě tohoto parametru. Hodnota může být znak abecedy nebo číslo od 0 do 127.
/N:*name* jméno příkazu, pod kterým bude možné tento znak používat v LaTeXu.
/T:*type* specifikace grafického formátu vstupního souboru znaku (PCX nebo ADI).
/F:*filename* jméno a cesta grafického souboru, z kterého bude vytvořen znak.
/D:*depth* /H:*height* /W:*width* Tyto parametry budou použity při generování TFM souboru. Implicitní hodnoty jsou *depth* nula, *height* a *width* podle rozměrů obrázku.
/X:*Xoffset* /Y:*Yoffset* Hodnoty *x* a *y* určují referenční bod znaku v PXL souboru. Defaultní hodnoty jsou pro *x* nula a pro *y* vertikální rozměr bitmapy.
/S:*count* Rumgraph může rozdělit velké obrázky na několik znaků fontu. Hodnota parametru udává počet částí, na které bude obrázek rozdělen.
/I: Znak bude vyveden inverzně.

3. Makefile

/M:*makefile* udává jméno souboru makefile, který obsahuje další volby.

Tento program je spíše používán pro import černobílých obrázků do TeXu. Lze sice používat pro konverzi fontů, ale proces je poněkud zdlouhavý — tvorba bitmap a případná konverze do monochromatického PCX, sestavení souboru makefile, konverze na PXL formát, konverze na PK formát. Chybějící kerning a sladění účaří bitmap jsou jasnými argumenty pro nepoužívání při generování TeXovských fontů. Mohou se ovšem vyskytnout mezní situace, kdy tento nástroj dojde svého využití.

Příklad 26: *Ukázka souboru makefile*

```
/P:omni
/R:300
/C:65 /N:A /T:PCX /F:A.PCX   /X:20
/C:66 /N:B /T:PCX /F:B.PCX   /X:40  /Y-20
/C:67 /N:C /T:PCX /F:C.PCX   /X:20  /I
```

6.5.5. PXtoPK, PKtest, Pbm2PK a Pbm2TeX

PXtoPK je součástí instalace TeXu v balíku mfware. Slouží k převodu staršího formátu fontů do podoby PK. Používají ho také některé jiné programy jako například Rumgraph.

PKtest je součástí distribuce PStoPK, tato utilita převádí obrazy v textové podobě .bm na formát PK (je tam i utilita pro opačnou konverzi).

Pbm2PK a Pbm2TeX jsou programy, které převádejí soubory Portable Bitmap (PBM) do formátu použitelného v TeXu. Pbm2PK vytváří bitmapu PK a Pbm2TeX převede obraz do LaTeXovského prostředí picture.

7. Převody a úpravy postscriptových a TrueType fontů

V různých fázích konverzí a úprav fontů může nastat situace, kdy je potřeba nějaké utility k úpravě fontu nebo získání nějakého dalšího souboru. Utility popisované níže jsou ty, jejichž potřeba může být častá nebo jejich schopnosti nejsou nahraditelné jinými programy.

7.1. Type 1 utility

7.1.1. T1utils

T1utils je balík konverzních programů, které převádějí mezi různými formáty Type 1 fontů. Programy pracují s formáty PFB (binární formát pro MSDOS) a PFA (ASCII vyjádření binárního formátu). Při používání ASCII formátu je třeba mít na zřeteli správnou (systémově závislou) reprezentaci konce řádku. Autorem je Piet Tutelaers.

T1ascii převádí Type 1 font PFB v binární podobě na hexadecimální formát PFA; program pracuje se standardním vstupem a výstupem.

T1binary převádí Type 1 PFA font v ASCII hexadecimální podobě na binární formu; parametrem -l block-length lze nastavit maximální délku bloku.

T1disasm rozkládá Type 1 font v hexadecimální nebo binární podobě na čitelnou formu — rozkrývá eexec a charstring na čitelný postscriptový zápis. Při rozkrývání z PFA souboru je výsledný soubor o jeden byte větší než při rozkrytí z binárního (první znak procento je zdvojen).

T1asm sestavuje Type 1 font z rozkrytého PFA nebo PFB souboru — kóduje charstrings a šifruje eexec. Přepínač -b způsobí výstup v PFB binárním formátu; -l length nastaví maximální délku bloku výstupního souboru.

T1tidy cílem programu je zmenšení velikosti Type 1 fontů bez ztráty informací; toho je dosaženo vyhledáním stejných tahů a jejich zařazení do podprogramů, nahrazením rozepsaného kódového vektoru jeho symbolickým zápisem a vymazáním nepoužívaných částí postscriptového kódu. Program dále opravuje kód pro použití ATM, pokouší se generovat chybějící znaky (uvozovky), může vytvářet kompozitní znaky z jejich složek, pokud má informace jak postupovat (CC v AFM), maže prázdné popisy znaků, pokud je přítomná metrika AFM je vytvořená nová s provedenými transformacemi, všechny položky metrik jsou testovány, opravovány a doplňovány a může rozepsat podprogramy přímo do popisu znaku. Program implicitně provádí vše co umí, pokud chceme pouze vybrané akce musíme nechtěné vypnout pomocí přepínačů. T1tidy očekává rozkrytou verzi fontu.

Příklad 27: *Ukázka práce T1utils*

```
t1ascii <gbc.pfb >gbc.pfa % převod do hexadecimální podoby
t1binary <gbc.pfa >new_gbc.pfb % převod zpět
t1disasm gbc.pfb gbc.ps % rozkrytí pfb souboru
t1disasm gbc.pfa gbc.sp % rozkrytí pfa souboru (o 1 byte větší)
t1asm -b gbc.ps gbcb.pfb % sestavení binárního souboru
t1asm gbc.sp gbca.pfb % sestavení hexadecimálního souboru
t1tidy gbc.ps >gbc.pss % ztýdování fontu
```

7.1.2. Pfm2Afm a Pf2Afm

Pro konverzi z postscriptových metrik do TEXovských se používá soubor AFM. Problém ovšem je, kde tyto soubory získáme, protože často máme instalaci pouze soubory PFB a PFM. Nejjednodušší je vlastnit program Fontographer nebo FontMonger, popř. FontLab; tyto programy tuto metriku vytvoří. Jde ovšem často o komerční produkty, takže pokud je nechceme koupit, musíme se vydat jinou cestou.

Existuje program Pfm2Afm, který napsal Russell Lang. Problém je v tom, že ve výsledném AFM souboru chybí informace o boudingboxech. Tento nedostatek nevadí programu PS2PK, protože ten tyto informace čerpá z PFB souboru.

Další možnost je použití „chytré dávky“ puštěné na GhostScript. K dávce Pf2afm je třeba mít postscriptový soubor pf2afm.ps, který vlastně dělá všechnu práci. Syntaxe:

pf2afm.bat *soubor*

Máme-li nainstalovanou pouze verzi pro Windows 95 je nutné tuto dávku mírně pozměnit:

Příklad 28: *Změněná dávka pf2afm.bat pro Windows 95*

```
@echo off
gswin32 -q -sDEVICE=nullpage -- pf2afm.ps %1
```

Dalším zdrojem AFM souborů může být internet. Například archív CTAN nebo FTP server firmy Adobe Systems (ftp.adobe.com). Když si koupíme komerční fonty, měli bychom zároveň obdržet i soubory AFM, nebo alespoň na požádání.

7.1.3. Mmpfb a Mmafm

Mmpfb a Mmafm je dvojice programů, které vytvářejí interpolované Type 1 fonty z Multiple Master fontů. Oba programy jsou vytvořeny pro Unixové systémy Eddiem Kohlerem, verze pro DOS zatím nebyla upravena. Použití:

mmpfb [options] [font file]
mmafm [options] [filenames]

Volby:

--output=file, -o file výstup bude směrován do souboru file
--pfb, -b výstupní formát bude PFB, tato volba je implicitní
--pfa, -a výstpuní fomrát bude PFA
--amcp-info nevytvoří font, ale AMCP soubor pro Mmafm
--weight=N, -w N nastaví weight axis na hodnotu N
--width=N, -W N nastaví width axis na hodnotu N
--optical-size=N, -O N nastaví optical size axis na hodnotu N
--style=N nastaví style axis na hodnotu N
--1=N (--2=N, --3=N, --4=N) nastaví osy podle pořadí na hodnoty N

Příklad:

```
mmpfb --weight=400 --width=600 MyriadMM.pfb > MyriadMM_400_600_.pfb
mmafm --weight=400 --width=600 MyriadMM.amfm > MyriadMM_400_600_.afm
```

Vytvořené single masters jsou určené pro program PS2PK, ale přesto některé znaky nelze vyrastrovat: '373 PSFakePush: Stack full. Po rozkrytí programem T1Disasm můžeme pozorovat části kódu typické pro multiple masters:

```
...
/FullName (Tekton MM Oblique) readonly def
/FamilyName (Tekton MM) readonly def
/Weight (All) readonly def
/isFixedPitch false def
/ItalicAngle 0 def
/UnderlinePosition -100 def
/UnderlineThickness 50 def
/BlendDesignPositions [ [0 0] [1 0] [0 1] [1 1] ] def
/BlendDesignMap [[[100 0][620 1]][[250 0][1100 1]]] def
/BlendAxisTypes [/Weight /Width ] def
...
```

7.1.4. T1tools

T1tools je soubor programů a dávek pro manipulaci s foty Type 1. Stručný neúplný výpis:

makeafm.bat vytváří metriku AFM z rozkrytého fontu
getmetri.bat generuje metriku AFM volámím GhostScriptu.
pfbshow.bat ukázka fontu
pcs.bat vypíše znakovou množinu fontu v porovnání s jiným fontem
pfblist.bat vypíše seznam fontů a jejich názvů
afm-pfm.bat vytváří PFM a INF soubor z metriky AFM
afm-sort.bat řadí kerningová páry AFM
pfb-enc.bat zjistí a vypíše kódování fontu
compose.bat dávka počítá pozice akcentů a jiných částí kompozitních znaků

Dávky pouze volají jiné programy např. GhostScript nebo utility od Adobe. Některé programy volané dávkami v balíku nejsou obsaženy.

7.1.5. T1Accent

Petr Olšák napsal program T1Accent, který vytváří v Type 1 fontu nové kompozitní znaky podle návodu v konfiguračním souboru. K tomuto programu je vhodné znát základy postscriptového jazyka a specifikace Type 1 fontů. Výstupem je rozkrytá podoba fontu s novými akcentovanými znaky. Pracné je zejména ladění fontu, zvlášť v porovnání s programy FontMonger nebo TypeTool. Dobře může posloužit v přípravné fázi počeštění fontu — nemusíme mechanicky vytvářet akcentované znaky kopírováním ze schránky systému.

Program je dobře dokumentován a je obsažen v balíku spolu se vzorovými konfiguračními soubory.

7.1.6. Další utility

Psfr a Psfw

Firma Fontlab nabízí Adobe Type 1 font Compiler a Decompiler k volnému použití. Jde o obdobné programy jako T1asm a T1disasm, ale vzájemná přenositelnost nefunguje.

Refont

Refont je utilita, která umožňuje:

- konverzi Type 1 pro Mac do formátu Windows PFB
- konverzi TrueType pro Mac do formátu Windows TrueType
- konverzi AFM souborů do Windows PFM souborů
- generování INF souborů z AFM souboru

Zajímavá je zejména možnost generování PFM souborů z AFM. Program pracuje spolehlivě a po zpětných převodech nevykazuje žádné změny v AFM souboru — všechny bounding boxy a kerningové páry jsou zachovány. Tímto lze změněnou metriku AFM promítnout v binární podobě fontu.

```
refont /i gbc.afm        % vytvoří gbc.pfm a gbc.inf
```

DumpPfm

DumpPfm převádí binární PFM soubor na textový, který odpovídá struktuře AFM. Použití:

```
DUMPPFM [/Verbose] file[.PFM] [file.[AFM]]
```

Volba /Verbose způsobí úplný výpis informací obsažených v souboru PFM. V PFM souboru nejsou informace o velikostech boundingboxů, takže AFM soubor není úplný. Může být použit programy, které rozměry boundigboxů čerpají přímo z PFB souboru, např. PS2PK.

7.2. TrueType utility

7.2.1. Ttf2Pfa

Ttf2Pfa je konverzní program, který převádí Windows TrueType fonty do formátu postscriptových fontů Type 3, které lze použít v jakémkoli interpretu Postscriptu. Program je určen a pracuje pod Unixovými systémy a autor sám pochybuje, že by program mohl pracovat např. pod DOSem. Ttf2Pfa byl testován v systému Solaris. Použití:

```
ttf2pfa fontfile.ttf fontname
```

Soubor fontfile.ttf je Windowsovký TrueType font, fontname je základ výsledných souborů fontname.pfa a fontname.afm.

Jako pokus byl sharewareový font Medusa v systému Solaris převeden na font Type 3. Vytvořený soubor medusa.pfa a medusa.afm byl testován ve Windows 95 v interpretu GhostScript — byl vytvořen postscriptový soubor test.ps, přidán řádek do souboru fontmap v adresáři GhostScriptu: /Medusa (medusa.pfa) ; a soubory s fontem Medusa překopírovány do podadresáře FONTS — úspěšně. Teoreticky lze tento font použít v TEXu vytvořením metriky z AFM souboru běžným způsobem.

Příklad 29: *Jednoduché testování Medúzy v GhostScriptu (test.ps)*

```
/Medusa findfont 32 scalefont setfont
20 600 moveto
(Woody Alen & Mighty Aphrodite) show
showpage
```

Obrázek 29: *Medusa Type 3 v TEXu a GhostScriptu prostřednictvím Ttf2Pfa*

7.2.2. TTFDump

TTFDump je program, který převede TrueType soubor do textové podoby. Spouští se z příkazové řádky v DOSu, za parametr se uvádí název fontu. Použití:

ttfdump filename.ttf [options]

Volby:

-nNNNN NNNN je číslo znaku (glyph) podle GlyphID; ve výpisu se objeví pouze záhlaví fontu a výpis znaku.
-tCCCC CCCC určuje tabulku, která bude vypsána; implicně všechny.
-h nevypíše se hlavička fontu.
-zHHH HHH specifikuje časové pásmo (+, − od Grennwiche).
-cNNNN pouze pro TrueType Collections: NNNN určuje, který subfont bude vypsán.
-q potlačení číslování bajtů a hintovacích informací, volba je určená pro porovnání verzí fontů, které mají stejný kód, ale liší se v offsetech.

Příklad 30: *TTFDump — příklady použití*

```
ttfdump bodoni.ttf >bodoni.txt
ttfdump bodoni.ttf -tglyf >bodoni.glf
ttfdump bodoni.ttf -n203 -tglyf -h >bodoni.203

První příklad vylistuje font bodoni.ttf do souboru bodoni.txt.
Druhý vypíše tabulku glyf a třetí znak 203 z tabulky glyf bez
hlavičky fontu.
```

Program je vhodný, když se chceme dozvědět, co všechno font obsahuje. Můžeme zjistit, které tabulky font má, případně nejsou-li prázdné; může být použit i jako vstup pro další zpracování, např. vytvoření metriky i s kerningovými páry pro jiné systémy.

Příklad 31: *Kerningová tabulka fontu TimesNewRoman*

```
'kern' Table - Kerning Data
---------------------------
Size = 5220 bytes
 'kern' Version:        0
 Number of subtables:   1

  Subtable format    0
  Subtable version   0
  Bytes in subtable  5216
  Coverage bits      0x1
  Number of pairs    867
  Search Range       3072
  Entry Selector     9
  Range Shift        2130
```

```
Left Glyph   Right Glyph   Kern Move
----------   -----------   ---------
       3          36         -113
       3          55          -37
       3          57          -37
       3          58          -37
       3          60          -76
       3         497         -113
       3         505         -113
       3         507         -113
       3         513         -113
     ...         ...          ...
     646          15         -205
     646          17         -205
     646         170          +25
     648          15         -203
     648          17         -203
     648          29          -51
     648          30          -51
     648         169         -180
     648         170          -78
     648         178          -51
```

Rozkrytí fontu trvá delší dobu, zvlášť u fontů TrueType Open, které jsou velmi rozsáhlé. Program může pracovat na pomalejších strojích i několik minut. Pentium 100 zpracovávalo times.ttf z instalace Windows cca 2 minuty, soubor TTF měl 190 KB a výsledný textový 3,3 MB (Pentium 200 čtyři vteřiny).

7.2.3. TTOAsm a TTODasm

TrueType Open Assembler a TrueType Open Disassembler jsou dva programy podobné utilitám T1Asm a T1Disasm. Používají se k vytváření, modifikování a testování tabulek TrueType Open fontů. Programy bohužel převádějí pouze pět TTO tabulek, nepřevádějí celý TrueType font. Použití:

ttoasm file.cmd
ttodasm file.cmd

Soubor file.cmd obsahuje názvy vstupních a výstupních souborů, první tři soubory jsou povinné, některé mají význam pouze pro rozkódování nebo pouze pro zakódování:

-STextFile jméno souboru, který bude převeden do binární formy.
-FFormatFile formátový soubor popisující tabulku fontu, jsou v instalaci a mají příponu .fmt.
-BBinaryFile název binárního souboru výstupního souboru.
-LListFile názvy formátových souborů, které budou vytvořeny při vytváření binárního fontu.
-EErrorFile soubor pro chybová hlášení.
-TTableName název tabulky, která bude rozkódována.

7.2.4. Addtable

Addtable je utilita v DOSu pro připojování binárních tabulek k existujícímu TrueType fontu. Například mohou být použity TrueType Open tabulky generované TTO Assemblerem. Použití:

addtable <table name> <table file> <old TTF> <new TTF>

7.2.5. TtfMap

TtfMap je program, který mění kódování TrueType fontu.

ttfmap [options] [<source file name> [<destination file name>]]

Volby:

-k kódování KOI-8
-w kódování Windows (1251)
-d kódování DOS (866)
-t uživatelské kódování: -t [codepage file name]
-n nepřidá příponu souboru
-b přemapuje v obráceném směru
-a automaticky přidá druhý Unicode index do fontů Cyrillic
-? nápověda

7.3. Profesionální nástroje pro postscriptové a TrueType fonty

Profesionální tvůrci fontů většinou nepoužívají k realizaci svých záměrů matematický popis znaků, ale specializovaný software na vytváření fontů. Převod do jazyka určitého formátu fontu je úkol tohoto software a autor digitalizovaného písma může své úsilí plně zaměřit na vizuální kvality fontu bez studia a znalostí matematického vyjádření svého návrhu, organizace souboru fontu a zdlouhavého testování vyhovujících parametrů.

Ovládání těchto nástrojů je podobné jiným grafickým programům, ale vzhledem k cíli má svá specifika. Většinou se v okně objeví znaková sada s kódy nebo jmény jednotlivých znaků. Pracovat můžeme s vlastnostmi celého fontu nebo jeho částmi (znaky, kódovým vektorem, metrikou, kerningovými páry). Výběrem určitého znaku můžeme vytvářet a měnit jeho obrys nebo upravovat jeho parametry (název, boundingbox, ...).

Výhodou těchto programů je snadnost ovládání, názornost a menší nároky na znalost vnitřního uspořádání fontu. Nevýhodou bývá, že jde o komerční produkty, za které se platí. Existují však starší profesionální nástroje, které je možné používat zdarma nebo aplikace s omezenou dobou použití (trials).

V této části nebude popsán způsob jak s těmito aplikacemi pracovat nebo jak postupovat při návrhu písma, ale upozornit na produkty, které mohou pomoci v různých situacích: například konverze různých formátů, korekce názvů znaků, kerningu a metrických informací, vytvoření určitých znaků vlastních nebo chybějících (ligatury a akcentované znaky), generování chybějící metriky AFM atd.

7.3.1. FontMonger

FontMonger, původem aplikace firmy Adobe Systems, je nyní spravovaná společností Ares a je možné si tento program najít na internetu a volně používat. Velikost instalace má kolem 700 kB a už při instalaci můžeme tušit, že jde o starší produkt.

FontMonger umožňuje běžnou správu fontů, ale chybí mu komfort a některé funkce oproti jiným programům. Jeho velkou výhodou je množství formátů, které zná: Type 1, Type 3, TrueType, Nimbus a to samé i pro Mac; načíst umí ještě více formátů. Další výhodou je možnost konverze fontů dávkovým způsobem. Při Convert Batch vytvoříme seznam fontových souborů a určíme cílový formát, jména jsou vytvářena automaticky a nepříliš vhodně.

Při ukládání v novém formátu Build Font, lze nastavit automatické generování AFM a INF souborů. FontMonger také umožňuje vytváření kapitálkových, skloněných a indexových mutací znaků.

7.3.2. Fontographer

Fontographer je také starší nástroj podobný programu FontMonger, ale umožňuje větší komfort práce. Zná sice méně formátů (pouze TrueType a Type 1), ale editace znaků je pohodlná, změna názvu znaku (Ctrl-I), úprava metrik a kerningových párů (Ctrl-K a Ctrl-7) je snadná a názorná.

Program je velmi vhodný pro korekci chybných názvů znaků a vytváření kompozitních znaků s využitím schránky systému pro přípravu fontu pro další konverze do systému TEX.

7.3.3. Aplikace firmy FontLab

Firma FontLab vytvořila celý balík programů pracujících s fonty. Aplikace umožňují vytvářet Multiple Masters fonty i jiné formáty pro rozsáhlé znakové sady. Jde o moderní produkt, ale komerční. Všechny programy je možné vyzkoušet v jejich demoverzích, které neumožňují ukládat výsledky práce s vyjímkou programu TypeTool, kde je tato funkce pouze omezená počtem.

7.3.4. Další nástroje

Firma Corel se svými programy CorelTrace (vektorizace bitmap) a CorelDraw podporují výstupní formát Type 1 a TrueType. Lze tedy obrázky nakreslené v tomto programu uložit a používat jako fonty.

Existují další nástroje na prohlížení, tisk a správu fontů, např. FontNavigator nebo FontPro, většinou jde o shareware nebo freeware programy.

8. Hodnocení a návrh konverzní cesty do prostředí TEXu

Předchozí kapitoly popisovaly nejdůležitější formáty fontů a konverzní programy, které umožňují jejich transformace. Z popisu programů je zřejmé, které cesty jsou vhodné a které méně.

8.1. Hodnocení konverzních programů

Jako na začátku můžeme programy rozdělit na tři druhy a zvolit parametry, podle kterých budeme programy hodnotit. Volba úrovně konverze a způsob realizace by se neměly posuzovat pouze podle následujících bodů, ale vždy by se mělo přihlédnout k danému fontu (jeho vlastnostem) a testovat úspěšnost konverze až do podoby, která se nejvíce blíží zamýšlenému finálnímu výstupu.

Hodnocena je věrnost vzoru, z kterého je metrika získávána, dále možnost zapojení do dávkového systému z důvodu automatizace a odstínění problematiky od nezkušeného uživatele a hodnocení nové užitné hodnoty, která by neměla být nižší.

8.1.1. Konverze metrických informací

Konverze metrických informací není příliš problematická na úrovni, kde se pohybuje TEX. Problém by samozřejmě nastal při práci s OpenType a CID fonty — problém rozsahu a zvláštních ligačních a substitučních znaků nelatinkových fontů. Do hodnocení jsou zahrnuty i programy, které vlastní konverzi neprovádí, ale pouze metrické informace upravují nebo rozšiřují.

Program	věrnost vzoru	dávkový systém	užitná hodnota
Afm2tfm	+	+	+
Ttf2tfm	+	+	+
Af2Pl	+	+	–
Fontinst	+	–	+
Accents	+	+	+
A2Ac	+	+	+

Tabulka 8: *Hodnocení konverzí metrických informací*

Z tabulky je zřetelné, že získání kvalitních metrik není problémem pro konverze, ale spíše tvůrce fontů. Konverzí do prostředí TEXu zvyšujeme užitnou hodnotu tím, že můžeme užívat ligatury a počešťovat fonty. Lze říci, že **používáním typografického systému TEX zhodnocujeme úsilí tvůrců fontů a konverze jsou velmi věrné.**

8.1.2. Konverze obrysů na bitmapy

Rastrováním obrysových fontů vystupujeme z teze o nezávislosti na výstupním zařízení a vážeme bitmapový font na určitý rastr. Ve spolupráci s dvi-ovladači se mohou tyto programy jevit podobně jako interprety fontů zabudované v systémech. Problematická se může jevit úroveň kvality těchto interpretů a interpretů programovaných výrobci fontů. S vysokou pravděpodobností lze předpokládat, že kvalita těchto programů nepřesáhne kvalitu komerčních produktů. Vzhledem k tomu, že tyto programy vycházejí z licencí výrobců, můžeme říci,

že kvalita rastrování je dostatečná, ale užíváním interpretů OS spíše na kvalitě získáme.

Program	věrnost vzoru	dávkový systém	užitná hodnota
PS2PK	+	+	–
Ttf2PK	+	+	–
Gsf2PK	+	+	–
TTF2Gf	+	–	–
Rumgraph	–	+	–

Tabulka 9: *Hodnocení získávání bitmap*

Popis v tabulce odpovídá předcházejícímu odstavci. Jsme schopni získat odpovídající bitmapy, ale jejich kvalitu určitě nezvýšíme použitím konverzních programů. Z rozboru a testování vyplývá, že **lze do prostředí TEXu importovat potřebné bitmapové obrazy fontů v dostatečné kvalitě, ale možnost rastrování fontů v jejich přirozeném prostředí se jeví jako opodstatněná a žádoucí.**

8.1.3. Konverze do metafontových zdrojů

Nejnižší kvality dosahujeme při konverzi do metafontových zdrojů. Nelze vhodně transformovat hintovací informace, které jsou při nízkých rozlišení nebo malých velikostech často rozhodující. Z růstem těchto veličin se však stávají méně podstatné a věrnost původnímu vzoru také roste.

Program	věrnost vzoru	dávkový systém	užitná hodnota
PS4MF	+	+	–
Ttf2MF	+	–	–
Fog2MF	+	–	–
Ega2MF	–	+	–

Tabulka 10: *Hodnocení získávání MF zdrojů*

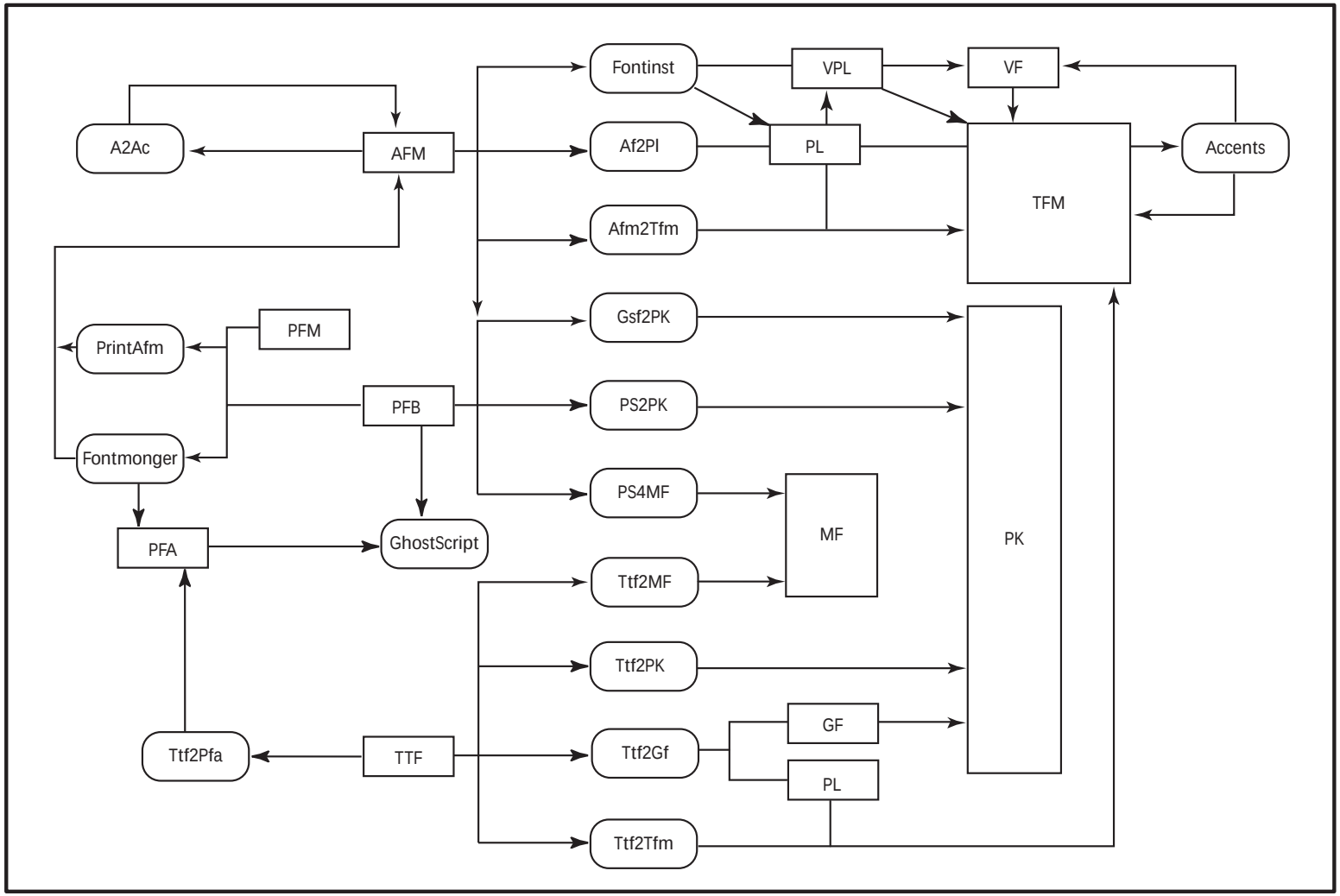
Konverze do METAFONTu patrně zůstane okrajovou variantou konverzí formátů fontů do prostředí TEXu. **Největším nedostatkem jsou nekonvertované hintovací informace, takže věrnost vzoru můžeme považovat za dostačující pouze při dostatečně hustém rastru.** Důvodem pro tyto konverze může být získání přechodných kontrolních bitmap nebo výchozí bod pro vytvoření nového metafontového písma.

8.2. Grafický model konverzí

Na následující straně je mapa možných konverzí fontů do prostředí TEXu. Nejsou zahrnuty všechny programy, vynechány jsou pouze ty, které nelze považovat z hlediska věrnosti původnímu vzoru za dostatečně kvalitní.

Výchozí soubory fontů jsou PFB, PFA, PFM, AFM; za cílové jsou považovány formáty TFM, VF, PK a MF — jsou značeny obdélníky. V oválech jsou jsou konverzní programy. Do mapy jsou zahrnuty i programy, které se vztahují k počešťování fontů. Mapa může být vhodnou názornou pomůckou pro nezkušené uživatele konverzních programů.

Mapa konverzí



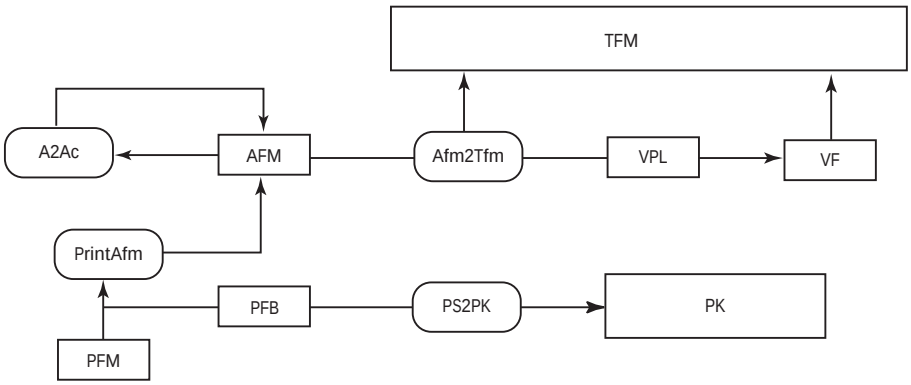
8.3. Návrh optimální konverzní cesty

Ne všechny diskutované programy splňují požadavky na kvalitu a zařazení do automatického konverzního systému. Důležitou vlastností programu je běh a parametrizace z příkazové řádky — umožní částečné odstínění problematiky a spolupráci různých programů a utilit. Při dobrém řešení by bylo možné zajistit propojení s dvi-prohlížeči (na unixových instalacích tomu tak je) a problém uživatele by spočíval pouze ve správném umístění fontu v adresářové struktuře. Bohužel instalace EmTEXu, respektive její ovladače, toto neumožňují, ale i tak lze vytvořit systém, který usnadní přípravu fontu pro prostředí TEXu. Z kapitoly o hodnocení je nasnadě provázanost i s jinými grafickými formáty.

8.3.1. Pro postscriptové fonty

Program Afm2tfm je univerzální z hlediska způsobu získání bitmap i cílového grafického formátu. Spolu s programem DVIPS lze realizovat rastrování i tisk na postscriptovém zařízení, spolu s PS2PK generovat potřebné bitmapy pro dvi-ovladače. PS2PK má s Afm2tfm provázané i přepínače na tvorbu skloněného a rozšířeného písma. Kapitálky je nutno generovat jiným způsobem.

Obrázek 30: *Automatický konverzní systém pro postscriptové fonty*



Výchozím bodem pro český font je česká metrika fontu. Soubor AFM považujeme za nejvýhodnější a výchozí bod pro počeštění. Pokud nechceme dělat úpravy některých znaků nebo kompozitů, vystačíme s pomocí virtuálních fontů. V opačném případě je vhodné použít T1Accent a ve Fontographeru ručně nabodeníčka upravit. Nesmíme zapomenout dogenerovat kerningové informace pro nové znaky! V každém případě použitím A2Ac zvýšíme užitnost fontu.

Virtuální popis fontu bude vždy použit k překódování metriky fontu. V případě nečeského postscriptového fontu použijeme virtuální popis i k vytvoření kompozitních znaků.

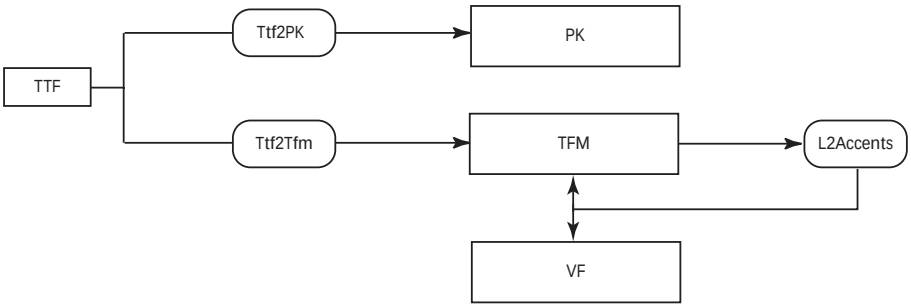
8.3.2. Pro TrueType fonty

Pro automatickou konverzi TrueType fontů je vhodná dvojice Ttf2tfm a Ttf2PK — zajistí metriku i potřebné bitmapy pro dvi-ovladače.

Předpokladem pro český font v prostředí TEXu je již český TrueType font. Druhou možností je konverze do podoby sedmibitových CM fontů a počeštit přes virtuální font programem L2Accents. Existuje možnost získat TFM soubor v kódování Adobe Standard Encoding a postupovat opět přes L2Accents.

Budeme-li používat implementaci pdfTEX, použijeme program Ttf2Afm. S metrikou AFM naložíme obvyklým způsobem — počeštění a konverze do prostředí TEXu. PdfTeX obsahuje mechanismus začlenění TrueType fontů do PDF souboru.

Obrázek 31: *Automatický konverzní systém pro TrueType fonty*



Použití virtuálních fontů se různí podle výchozího TrueType fontu. Pokud TrueType font obsahuje české znaky i se správnými postscriptovými názvy znaků, bude virtuální font použit pouze k získání metriky s kerningovými informacemi. Do prostředí TEXu bude zařazena jen tato metrika. V případě, že zároveň s konverzí do prostředí TEXu bude probíhat i počeštění fontu, musí být zařazena raw metrika, virtuální metrika i virtuální popis fontu.

9. Návrh konverzního systému pro prostředí TEXu

Cílem této části je navrhnout samostatné moduly, které umožní snadnou přípravu neTEXovských fontů pro pro prostředí TEXu. Půjde o jednoduché dávky pro OS DOS. Celý problém opět rozložíme na logické celky.

Pro postscriptové fonty:

1. získání metriky AFM
2. počeštění metriky AFM
3. získání metriky TFM a VF
4. zanesení vazeb pro DVIPS
5. vytvoření bitmap PK

Pro TrueType fonty:

1. získání metriky TFM
2. získání bitmap PK
3. počeštění metriky VF

Automatické konverze do formátu METAFONTu jsou obtížné. Pro TrueType fonty neexistuje program, který lze zapojit do dávkového režimu a postscriptové fonty příliš často vyžadují „individuální přístup“. Pro plně český font bez kompozitních znaků může posloužit příklad k MF4PS a připravený konfigurační soubor s kódováním českých fontů xl2.cfg.

9.1. Postup při konverzích fontů

Vyřešení absolutně automatického konverzního systému, který by nevyžadoval znalosti a aktivní účast uživatele, a zároveň dovedl rozhodnout o nejlepším způsobu transformace, by vyžadovalo nadměrné úsilí. Poměrně snadno lze navrhnout jednoduché dávky, které může případný zájemce upravit pro svůj operační systém a instalaci TEXu. Předpokladem pro úspěšné zvládnutí konverzí tedy bude určitá znalost problematiky a pomocných nástrojů.

9.1.1. Postup u postscriptových fontů

Výchozím bodem je kvalitní font, musí obsahovat všechny potřebné znaky nebo kompozity, názvy znaků musí odpovídat kresbám. Takovou kontrolu můžeme provést Fontographerem nebo testem např. v GhostScriptu. Je-li font v pořádku můžeme přikročit k získání metriky AFM. Můžeme využít Fontographeru nebo požádat o vygenerování GhostScript. Obsahuje-li font kompozitní znaky, je dobré se přesvědčit, zda jsou v AFM tyto definice, například Fontmonger, TypeTool a GhostScript je negenerují.

Není-li font český ani počeštěný, použijeme A2Ac k doplnění akcentovaných znaků a kerningových párů. To můžeme provést i s českým fontem, protože dojde k vytvoření specifických slovenských a polských znaků. K vytvoření metriky je jediným vhodným nástrojem Afm2tfm. Bez ohledu na zamýšlený způsob získání bitmap je dobré zapsat nová data do psfonts.map. Rastrování fontů provede program PS2PK; před tím se spustí vygenerování databáze fontů mkpsres.

9.1.2. Postup u TrueType fontů

Konverze TrueType fontů se rozpadá na dva směry podobně jako u postscriptových fontů, ale musí se bohužel řešit odděleně. Máme-li český font, postupujeme přes virtuální font, kterým pouze vytvoříme metriku — dále používáme pouze PK a TFM. Pokud font není český, ale obsahuje kompozity, generujeme sedmibitový font kódovaný OT1 a L2Accents počeštíme.

9.2. Dávky a jejich použití

Pro konverze byly vytvořeny tři dávky — pro postscriptové, české a nečeské TrueType fonty. Uživatel musí sám rozhodnout, kterou dávku zvolí. Pro každou variantu se předpokládá, že font splňuje všechny náležitosti a případné nedostatky nebo chyby nejsou analyzovány.

Dávky mají stejné spuštění, parametry i průběh. Každá dávka se jmenuje !make.bat a jako parametry se zadávají názvy fontů určené pro převod; bez přípony. U postscriptových fontů je vyžadována shoda jmen souborů. Průběh dávek lze rozdělit do pěti částí:

1. **Inicializační** testuje zda byl zadán parametr a inicializuje soubory pro přesměrování výpisů — *.map soubory slouží pro úschovu parametrů pro další zpracování a @*.log jsou přesměrované výstupy konverzních programů, které mohou sloužit pro hledání příčin neúspěchu nebo analýze průběhu konverze. V této fázi je také vytvořena databáze PS fontů.
2. **Konverzní** vytváří metriky a jeden PK soubor pro testování. Počešťované metriky mají prefix c, nepočešťovaným jsou ponechány původní názvy. PK soubor je generovaný pro rozlišení 300 dpi a velikost 12dd.
3. **Kopírovací** přemístí důležité vytvořené soubory do adresářové struktury TEXu. Zde mohou být problémem různé instalace TEXu (nutnost vlastní úpravy). Také jsou doplněny informace do souborů psfonts.map a ttfonts.map v konfiguračním adresáři DVIPS, kam byl umístěn i parametrický soubor pro TrueType fonty.
4. **Úklidová,** která není aktivní (pro zapojení vymazat rem); tato akce může být samostatně vyvolána dávkou !uklid.bat.
5. **Chybová,** ta obsahuje návod a příklad použití

Nové soubory a adresáře:

Konverzní balík je připraven ve vhodné adresářové struktuře pro instalaci EmTEX. Dávky jsou v novém adresáři CONVERTE. Pro nové metriky TFM slouží podadresáře TFM\PSFONTS a TFM\TTFONTS. Dávky pracují s vlastními kódovacími soubory. Soubor xl2.enc byl upraven o definice ligatur fi a fl (vis poslední dva řádky), cm.enc je kódový soubor pro sedmibitové kódování CM fontů. Pro přípravu generování PK souborů jsou určeny utility mkttf a mkpsf, které vytvoří dávku s požadovanými parametry. Tyto programy jsou volány dávkami pro kovertování obrysů na PK bitmapy; jsou označeny !example.bat, protože slouží jako příklad práce s programy mk??f.exe. Na programy navazuje utilita cpf, která provádí kopírování PK souborů do adresářové struktury instalace TEXu.

Obrázek 32: *Dávka pro konverzi postscriptových fontů (zkráceno)*

```
@echo off
if "%1"=="" goto noafm
echo. >psfonts.map
echo. >@vptovf.log
echo. >@a2ac.log
if exist psres.dpr del *.dpr
mkpsres >nul

:smycka
if "%1"=="" goto copying
a2ac %1.afm ..\data\cscorr.tab c%1.afm >>@a2ac.log
afm2tfm c%1 -t ..\data\xl2.enc -v c%1 r%1 >>psfonts.map
vptovf c%1 >>@vptovf.log
del *.dpr >nul
ps2pk -X300 -Y300 -P12.84 -a%1.afm %1.pfb r%1.pk
shift
goto smycka

:copying
copy *.pk %emtexdir%\fonts\pixel.lj\385dpi
copy *.vf %emtexdir%\fonts\vf\a2ac
copy *.tfm %emtexdir%\tfm\psfonts
type psfonts.map >> %texconfig%\psfonts.map

rem For deleting generated files automaticly unrem following line
rem call !uklid.bat
goto end
:noafm
echo.
echo Error: No AFM file specified.
echo.
:end
```

Dávka byla uvedena téměř celá. Následující dávky jsou zkráceny natolik, že obsahují pouze podstatnou konverzní část uvozenou návěštím :smyčka. Celé dávky je možné si prohlédnout v příloze.

Obrázek 33: *Dávka pro konverzi NonCZ TrueType fontů (zkráceno)*

```
.... inicializační část
:smycka
if "%1"=="" goto copying
ttf2tfm %1 -P 3 -E 1 -n -u -q -T ..\data\cm.enc -v c%1 %1 >>ttfonts.map
vptovf c%1 %1 %1 >>@vptovf.log
ttf2pk -q -n %1 385
l2accent %1.tfm c%1.vf c%1.tfm >>@l2accent.log
del %1.vf >nul
shift
goto smycka
.... kopírovací část
.... úklidová část (neaktivní)
.... chybová část
:end
```

Obrázek 34: *Dávka pro konverzi CZ TrueType fontů (zkráceno)*

```
.... inicializační část
:smycka
if "%1"=="" goto copying
ttf2tfm %1 -P 3 -E 1 -n -u -q -T ..\data\xl2.enc -v c%1 %1 >>ttfonts.map
vptovf c%1 %1 %1 >>@vptovf.log
del %1.vf >nul
ttf2pk -q -n %1 385
shift
goto smycka
.... kopírovací část
.... úklidová část (neaktivní)
.... chybová část
:end
```

Postup při konverzích: překopírování souborů do správného adresáře pro konverze, provedení konverze, kontrola a doplnění údajů do prostředí instalace TEXu. Pro průběh konverze je nutné nastavení systémových proměnných. Tohoto efektu lze dosáhnout spuštěním shellu z prostředí MNU.

Program Mkpsf, Mkttf a Cpf

První dva programy vytvářejí dávkové soubory, kde jsou obsaženy příkazové řádky pro vytvoření PK bitmap. Pro TrueType fonty je tento program užitečný i z hlediska toto, že nelze pro Ttf2PK specifikovat velikost bitmap, ale pouze rozlišení. Cpf je utilita, která kopíruje PK soubory do adresářové struktury TEXu. Syntaxe a příklad:

mkttf Fontname [-d] [-r Target resolution] [-s Sizes]
mkpsf PFBname PKname [-d] [-r Target resolution] [-s Sizes]
cpf [-q] PKdir

Fontname jméno fontu v ttfonts.map
PFBname název postscriptového fontu, shodný PFB a AFM
PKname název generované bitmapy PK, často odpovídá psfonts.map
PKdir adresář bitmap daného zařízení, %emtexdir%\fonts\pixel.lj
-d velikost bitmap v didotových bodech
-r rozlišení PK souborů (300)
-s požadované velkosti (10), lze udat více
-q potlačí informace o průběhu programu

```
mkttf arial -d -r 300 -s 10 12 14 17
mkpsf times rtimes -d -r 300 -s 10 12 14 17
call arial.bat
call rtimes.bat
cpf %emtexdir%\fonts\pixel.lj
call !copy.bat
call !smaz.bat
```

Pro vlastní generování jsou volány příslušné programy. Dávky neodkazují na parametrické soubory instalace EmTEXu, předpokládají vše v aktuálním adresáři — fonty, ttfonts.map a psres.dpr. Do balíku jsou zahrnuty i zdrojové kódy programů v Pascalu, které lze upravit pro vlastní potřebu.

10. Přílohy

Přílohy jsou umístěny na disketě. Disketa obsahuje důležité příklady řešené v diplomové práci, konverzní dávky a část adresářové struktury pro běh konverzních dávek — před testováním je nezbytné překopírovat do instalace CSTEXu. Pro běh a nastavení systémových proměnných je vhodné pracovat v shellu prostředí MNU.

10.1. Příklady popisované v diplomové práci (disketa)

Důležité příklady nebo výsledky příkladů jsou v adresáři Samples. Názvy podadresářů jsou voleny podle názvu programu nebo utility řešené příkladem. Každý adresář (příklad) obsahuje dávku !make, která provede všechna volání jiných programů; může se objevit i dávka !smaz, která maže soubory, které nesmí být kopírovány dávkou !copy do adresářové struktury instalace TEXu. Ta také předpokládá již vytvořené adresáře a musí být volána po úspěšném provedení dávky !make. Není-li dávka !copy přítomna, musí být kopírování a změny v parametrických souborech udělány ručně. Dávka !uklid maže všechny vytvořené soubory v aktuálním adresáři, který se tím vrací do původního stavu.

Výstupy programů popisující prováděné akce jsou přesměrovány do souborů @*.log; mohou sloužit pro zjištění všech provedených akcí nebo hledání případných chyb (* zastupuje program jehož výstup je přesměrován). Adresáře obsahují všechny potřebné vstupní soubory. Programy jsou volány z adresáře EMTEX\BIN a musí být v cestě %path%.

Jako poslední soubor je v adresáři zt-*.tex, kde hvězdička znamená kódování souboru — to musí být shodné se vstupním kódování TEXu. Vyskytují se tři druhy zt-win.tex, zt-lat.tex a zt-kam.tex pro kódování Windows1250, Latin2 a Kamenických. Pokud není přítomný soubor v potřebném kódování použijte CSTOCS z nabídky MNU. Jde o testovací soubor, který po bezchybném provedení dávek !make a !copy bude zobrazovat výsledky konverze. Pokud jsou generovány bitmapy PK, jsou v rozlišení 300 dpi pro tiskárny Laser300 a velikosti 12dd.

A2Ac	PS4MF
Afm2tfm	Rumgraph
Bm2Font	Ttf2MF
GsftoPK	Ttf2Pfa
L2Accents	Ttf2PK
Pf2Afm	Ttf2Tfm1
PS2PK	Ttf2Tfm2

Tabulka 11: *Seznam příkladů (adresářů)*

10.2. Dávky a programy pro automatizaci konverzí (disketa)

Tyto dávky jsou v adresáři CONVERTE. Popis dávek a jejich běh je popsán v předešlé kapitole. Vše co platí o příkladech platí i o konverzních dávkách. Ty navíc využívají proměnných prostředí a pracují s vlastními adresáři a parametrickými soubory. Adresáře obsahují pouze dávky !make nebo !example a !uklid. Dávka !example ilustruje použití programů mkpsf a mkttf.

Kopírování do adresářové struktury EmTEXu probíhá u dávek !make automaticky pomocí programu cpf. Cpf čte soubory v aktuálním adresáři, které mají za příponu číslo — toto číslo znamená rozlišení pro dané zařízení, které je také parametrem tohoto programu. Program vytváří dvě dávky: !copy a !smaz; první obsahuje příkazy pro vytvoření neexistujících adresářů a kopírování souborů a druhá maže v aktuálním adresáři soubory, které byly kopírovány.

Dávka !general ilustuje možnost obecnějšího zápisu dávky !example. Vhodnými úpravami pro určitý operační systém a implementaci TEXu, lze dosáhnout větší integrace s prostředím TEXu.

Adresář	**funkce**
DATA	soubory kódování a změnové soubory
PS2TEX	konverze metrik PS fontů a jednoho PK 12dd
TT2TEX	konverze metrik TT fontů nečeských a PK 12dd
PS2TEX.CZ	konverze metrik TT fontů českých a PK 12dd
PS2PK	konverze PS obrysů na PK ve zvolených velikostech
TT2PK	konverze TT obrysů na PK ve zvolených velikostech

Tabulka 12: *Adresáře konverzního systému*

Do adresářové struktury instalace CSTEXu je nutné přidat nové adresáře a soubory. Pro běh konverzního systému jsou zkopírovány do podadresáře BIN tyto soubory:

Program	**činnost**
a2ac	počeštění metrik AFM
afm2tfm	konverze AFM metrik do formátu TFM a VPL
cpf	umístění PK bitmap do adresářové struktury instalace TEXu
l2accent	počeštění TFM metrik v kódování OT1 a Adobe Standard
mkpsf	příprava dávky pro generování PK bitmap z PS fontů
mkpsres	vytvoření databáze PS fontů pro ps2pk
mkttf	příprava dávky pro generování PK bitmap z TT fontů
ps2pk	generování PK bitmap z PS fontů
ttf2pk	generování PK bitmap z TT fontů
ttf2tfm	konverze TrueType metrik do formátu TFM a VPL

Tabulka 13: *Programy konverzního systému*

Dávky konverzního systému i příkladů vypisují návod, pokud jsou spuštěny bez parametrů. Další informace jsou v souboru dávky, v řádcích s příkazem rem. Dávky i nové programy jsou psány tak obecně, že přenos do jiných operačních systémů by neměl být problémem.

11. Závěr

Znakové sady v typografických systémech mají jeden cíl: kvalitní obraz písma v tištěném dokumentu. Liší se v matematickém zápisu křivek, interpretaci typografických vlastností písma, zápisem metrických informací i průběhem získávání obrazu znaků. Přes všechny tyto rozdíly mají společné vlastnosti. Těchto společných vlastností lze s úspěchem využít k vzájemným převodům formátů. Důvody pro konverze jsou opodstatněné, ale vždy jim musí předcházet úvaha o jejich nutnosti a míře.

Problematika konverzí formátů fontů není příliš zmapovaná. Dodnes neexistuje literatura, která by popisovala nejdůležitější formáty, zabývala se důvody pro konverze a funkcemi konverzních programů. Přitom tato práce dokazuje na typografickém systému TEX užitečnost a prospěšnost takových převodů.

Po prostudování formátů fontů a testování náležitých programů lze konstatovat, že konverze mezi formáty ve spojení s určitými typografickými systémy mohou zvyšovat hodnotu výsledné tiskoviny a kvality sazby. Existuje nebezpečí chybně zvolené transformace a nový formát může znehodnocovat původní vzor. Při vhodné konverzi můžeme využívat vlastností fontů ve zdrojovém formátu nevyužitelné nebo vytvářet nové, které budou zvyšovat hodnotu písma.

Převod písma není většinou jednorázovou akcí, často je třeba se vracet a upravovat různé parametry. Tento proces odladění fontu je sice zdlouhavý, ale pečlivá analýza písma před konverzí a zvolení vhodného postupu, může tuto dobu výrazně zkrátit.

Zjištěné poznatky a vytvořené programy mohou být cenným zdrojem informací o transformacích fontů pro každého, kdo se chce o této problematice něco dozvědět nebo si vědomosti rozšířit. Ve spojení na odkazovanou literaturu bude jistě velmi málo neřešitelných situací, které se při formátových konverzí fontů mohou vyskytnout.

12. Použitá literatura

ADOBE DEVELOPER SUPPORT *Adobe Font Metrics Specificaton — Version 4.1.* ftp.adobe.com/supportservice/devrelations/PDFS/TN/5004.AFM_Spec.pdf.

ADOBE DEVELOPER SUPPORT *Designing Multiple Master Typefaces — Technical Note #5091.* ftp.adobe.com/supportservice/devrelations/PDFS/TN/5091.Design_MM_Fonts.pdf.

ADOBE DEVELOPER SUPPORT *Type 1 Font Format Supplement — Technical Specification #5015.* ftp.adobe.com/supportservice/devrelations/PDFS/TN/5015_Type 1_Supp.pdf.

ADOBE SYSTEMS INCORPORATED *Adobe Type 1 Font Format — Version 1.1.* Massachusetts: Addison-Wesley,1990. ISBN 0-201-57044-0.

GOOSENS, M., RAHTZ, S., MITTELBACH, F. *The LaTeX Graphics Companion.* Massachusetts: Addison Wesley,1994. ISBN 0-201-85469-4.

HEROUT P. *Postscriptové fonty pro ty, co o nich moc nevědí.* Zpravodaj CSTUGu. 1996, roč. 6, č. 1. ISSN 1211-661.

HORÁK K. *Můj zápas s Metafontem aneb Pérovky a jiná zvěrstva (s ukázkami.)* Zpravodaj CSTUGu. 1992, roč. 1, č. 3. ISSN 1211-661.

KOLEKTIV AUTORŮ *Jak publikovat na počítači.* Science: Veletiny,1996. ISBN 80-901475-77.

KNUTH D. E. *The Metafontbook.* Massachusetts: Addison-Wesley,1994. ISBN 0-201-13444-6.

KNUTH D. E. *The TEXbook.* Massachusetts: Addison-Wesley,1994. ISBN 0-201-52983-1.

KNUTH D. E. *Virtual fonts, a more fun for grand wizards.* TUG-boat. 1990, roč. 11, č. 1.

MICROSOFT CORPORATION *OpenType specification v. 1.1.* http://www.microsoft.com/typography/otspec/otsp11.zip.

MICROSOFT CORPORATION *TrueType 1.0 Font Files Technical Specification Revision 1.66.* http://www.microsoft.com/typography/tt/ttf_spec/ttspec.zip.

MICROSOFT CORPORATION *TrueType Open Font Specification v. 1.0.* http://www.microsoft.com/typography/tt/tt_open/msdn/ttospec.zip.

MRÁZ T. *Postscriptová písma v TEXu.* Zpravodaj CSTUGu. 1994, roč. 4, č. 2 ISSN 1211-661.

OLŠÁK P. *EncTEX — změny konverzních tabulek TEXu.* Zpravodaj CSTUGu. 1997, roč. 7, č. 3. ISSN 1211-661.

OLŠÁK P. *TEXbook naruby.* Brno: Konvoj,1997. ISBN 80-85615-64-9.

OLŠÁK P. *Program A2Ac — manipulace s fontem na úrovni Postscriptu.* Zpravodaj CSTUGu. 1996, roč. 6, č. 1. ISSN 1211-661.

OLŠÁK P. *Putování písmene ř z klávesy na papír.* Zpravodaj CSTUGu. 1997, roč. 7, č. 3. ISSN 1211-661.

OLŠÁK P. *Typografický systém TEX.* Praha: 1995. ISBN 80-901950-0-8.

OLŠÁK P. *Úvaha o fontech v CSTEXu.* Zpravodaj CSTUGu. 1993, roč. 6, č. 1. ISSN 1211-661.

RYBIČKA J. *LATEX pro začátečníky.* Brno: Konvoj,1995. ISBN 80-85615-42-8.

RYBIČKA J. *LATEX pro začátečníky (2. vydání).* Brno: Konvoj,1997. ISBN 80-85615-77-0.

SLEZÁK M. A KOL. *Písmo ve výtvarné výchově.* Praha: SPN,1985.

SOJKA P. *Virtuální fonty, accents a přátelé.* Zpravodaj CSTUGu. 1994, roč. 4, č. 2. ISSN 1211-661.

ŠEDIVÝ A KOL. *Kreslíme Metafontem.* Zpravodaj CSTUGu. 1998, roč. 8, č. 1. ISSN 1211-661.

WAGNER Z. *Instalace českých Postscriptových fontů.* ftp.cstug.cz.

Tato práce byla zpracována typografickým systém TEX a může sloužit jako příklad využití navrženého konverzního systému pro převod postscriptových fontů do prostředí TEXu.

Pro původní postscriptové fonty Type 1 Esprit-Book, Esprit-Italic, Esprit-Bold a Myriad-Roman byly získány AFM metriky programem Fontographer, kde byly vytvořeny akcentované české znaky, doplnění kerningových údajů do nové metriky pomocí utility A2Ac.

Vygenerovaná metrika byla konvertována programem Afm2Tfm, pro ladění v prostředí TEXu byly také vytvořeny PK bitmapy pomocí PS2PK. Pro zobrazení písma strojopisu, které slouží k výpisům zdrojových kódů a jiných přímých příkazů počítači, bylo použito bezpatkové proporcionální písmo Myriad-Roman. Při tomto převodu byl použit kódový vektor XT2.

Pro maximální přenostitelnost jsou obrázky obsažené v práci ve formátu EPS (generované programem Illustrator 7.0 a PhotoShop 3.05). Protože tisk musí probíhat za asistence postscriptového interpretu, byl dvi soubor převeden pomocí DVIPS do Postscriptu. Písma byla přibalena do tohoto souboru a rastrování provádí postscriptový interpret.

Protože postscriptový soubor má velikost cca 8 MB, byla vytvořena programem Distiller PDF verze této práce. Při této příležitosti byl postscriptový soubor obohacen o krátký kód, který využívá možností operátoru pdfmark *(pohyb dokumentem pomocí bookmarks).*

Všechny výše uvedené formáty dokumentu budou spolu s obsahem příloh dostupné v síti internet.