OFS: Olšákův fontový systém

OFS je makro, které umožní přehledně v TEXu vybírat rodiny fontů, jejichž názvy jsou v souladu s názvy z písmového katalogu. Toto makro je z uživatelského pohledu stejné v plainu i LaTEXu. Výklad použití OFS bude tedy společný pro obě skupiny uživatelů. Pokud některý text platí jen pro csplain (a podobné formáty), bude uveden slovem PLAIN: a pokud je text určen pro uživatele LaTEXu2e, je uveden slovem LATEX:.

Pod skupinu PLAIN: patří nejen csplain, plain a odvozené formáty, ale též dnes už nepoužívaný LaTEX2.09 bez NFSS. Pod skupinu LATEX: bychom mohli zařadit všechny formáty, které pracují s NFSS 2, především tedy LaTEX2e.

Po zavedení příslušných maker v záhlaví dokumentu (viz níže) se na terminálu mimo jiné objeví:

```
PLAIN: OFS (Olsak's Font System) based on plain initialized. <verze>
LATEX: OFS (Olsak's Font System) based on NFSS initialized. <verze>
```

Výhody OFS:

- Sjednocuje rozhraní pro uživatele LaTEXu i plainu.
- Pomocí příkazu `\fontusage` dostaneme na terminál a do logu stručnou informaci o použití maker z OFS.
- Umožňuje pracovat se skutečnými názvy písmových rodin tak, jak jsou uvedeny v písmovém katalogu a tím nenutí uživatele si kromě těchto názvů pamatovat ještě interní zkratky použité v NFSS nebo v názvech metrik fontů.
- Umožní pracovat s fontem rozloženým do dvou metrik (základní a rozšiřující) jakoby se jednalo o jediný font.
- Umožní na začátku dokumentu vybrat oblíbené kódování základní metriky, jsou-li metriky připraveny v různých kódováních (typicky kódování IL2 a T1).
- Definuje rozhraní pro vytváření pomocných souborů, které obsahují konverzní informace mezi „dlouhými názvy“ fontů z katalogu a PLAIN: názvy metrik nebo LATEX: zkratkami používanými v NFSS.
- PLAIN: Umožňuje podobnou nezávislost výběru jednotlivých parametů fontů, jako NFSS v LaTEXu.
- PLAIN: Definuje deklarační příkazy pro podchycení kódování fontů.
- PLAIN: Je možné pro různé velikosti registrovat různé metriky, což využijeme zejména u rodiny Computer Modern.
- PLAIN: Obsahuje nástroje na použití PostScriptových fontů i v matematické sazbě.
- Interaktivní makro `ofstest.tex` umožní tisknout vzorky odstavců ve zvolených písmových rodinách, tisknout tabulky fontů, katalogy fontů, vzorky matematické sazby a seznamy znaků ve fontech včetně jejich TEXových sekvencí. Stačí napsat na příkazový řádek `tex ofstest [allfonts]` nebo `csplain ofstest [allfonts]` a řídit se pokyny na terminálu.

1. Členění písmových rodin

\rm \bf \it \bi

Makro OFS vychází z toho, že většina dnes dodávaných písmových rodin obsahuje čtyři řezy: základní (`\rm`), tučný (`\bf`), kurzívu (`\it`), a tučnou kurzívu (`\bi`). Těmto řezům budeme říkat „standardní varianty". Po zavedení rodiny příkazem `\setfonts` (viz níže) lze pak přepínat pomocí `\rm`, `\bf`, `\it` a `\bi` mezi variantami v dané rodině. První tři uvedené přepínače jsou známé z plainu (v LaTeXu přetrvávají z dob zašlé slávy LaTeXu2.09) a čtvrtý přepíná do varianty BoldItalic.

Některé rodiny mohou deklarovat přepínače dalších „nadstandardních variant" a u některých rodin naopak může chybět i některá ze standardních variant.

Varianty v některých rodinách se mírně liší od zde uvedených názvů pro standardní varianty. Místo tučného písma může být třeba přítomno jen polotučné a místo kurzívy třeba skloněné písmo. Přepínače `\rm`, `\bf`, `\it` a `\bi` zůstávají pokud možno zachovány, takže by nás nemělo vyvést z rovnováhy, že v některé rodině třeba `\it` přepíná do skloněného písma a ne do kurzívy.

Pokud chceme současně pracovat s dalšími přepínači fontů jiné rodiny (například `\tt` pro strojopis), je možné takové přepínače deklarovat pomocí příkazu `\fontdef`.

Členění fontů do rodin je deklarováno v souborech, které mají v plainu podobný význam, jako `fd` (Font Definition) soubory z LaTeXu. Doporučená přípona souboru je pro plain `tex` a pro LaTeX `sty`. Těmto souborům budeme říkat „deklarační soubory". Podle názvu deklaračních souborů by mělo být uživateli zřejmé, které rodiny jsou v něm deklarovány. Příklady:

```
PLAIN:          LATEX:
sjannon.tex,    sjannon.sty   ... souhrnná rodina Jannon ze Střešovic
a35.tex,        a35.sty       ... základní skupina 35 fontů od Adobe
```

Vidíme, že deklarační soubory (na rozdíl od `fd` souborů z LaTeXu) obsahují obvykle definice více souvisejících rodin. Mezi těmito soubory uživatel vybírá ty, které bude potřebovat a uvede jejich jména při zavedení makra OFS v záhlaví svého dokumentu. Aby měl uživatel zjednodušenou práci, zakládáme obvykle též souhrné soubory, které obsahují `\input` nebo `\RequirePackage` na jednotlivé deklarační soubory:

```
skatalog.tex, skatalog.sty ... všechny fonty ze Střešovic
allfonts.tex, allfonts.sty ... všechny fonty na TeXové instalaci
```

2. Uživatelské příkazy

2.1. Zavedení OFS

Pro příklad předpokládejme, že budeme chtít pracovat s fonty ze souhrnné rodiny Jannon a DynaGrotesk. Protože se jedná o rodiny ze Střešovické písmolijny od pana Štorma, najdeme odpovídající deklarační soubory v adresáři `storm`. Na Internetu je získáme společně s makrem OFS například na `www.cstug.cz/stormtype`. Souhrnné rodiny Jannon a DynaGrotesk jsou deklarovány v těchto souborech: PLAIN: `sjannon.tex`, `sdynamo.tex`; LATEX: `sjannon.sty`, `sdynamo.sty`. Písmeno `s` na začátku názvů znamená, že fonty pocházejí ze Střešovické písmolijny. Jména těchto souborů bez přípony je nutné uvést v hranaté závorce při zavedení makra OFS takto:

PLAIN: `\input ofs [sjannon, sdynamo] % mezera před "[" je nutná!`

LATEX: \usepackage [sjannon, sdynamo] {ofs}

\showfonts

Dále můžeme pracovat s rodinami, které jsou obsaženy v souhrnných rodinách Jannon a DynaGrotesk. Napíšeme-li do dokumentu příkaz `\showfonts`, zobrazí se na terminál (a zapíše do logu) seznam názvů jednotlivých rodin, které dále můžeme používat. Po zavedení `sjannon` a `sdynamo`, jak bylo uvedeno výše, zobrazí příkaz `\showfonts` tento seznam:

```
OFS (1.1): The list of known font families:
defaults:
   [CMRoman/]           \rm, \bf, \it, \bi, \sl
   [CMSans/]            \rm, \bf, \it,  -
   [CMTypewriter/]      \rm,  - , \it,  - , \sl
   [Times/]             \rm, \bf, \it, \bi
   [Helvetica/]         \rm, \bf, \it, \bi, \nrm, \nbf, \nit, \nbi
   [Courier/]           \rm, \bf, \it, \bi
sjannon.tex:
   [JannonAntikva/]     \rm, \bf, \it, \bi, \mr, \mi
   [JannonText/]        \rm, \bf, \it, \bi, \mr, \mi
   [JannonCaps/]        \rm, \bf, \it, \bi
sdynamo.tex:
   [DynaGroteskDXE/]    \rm, \bf, \it, \bi
   [DynaGroteskRXE/]    \rm, \bf, \it, \bi
   [DynaGroteskLXE/]    \rm, \bf, \it, \bi
   ... ⟨a dalších 15 rodin DynaGrotesk⟩ ...
```

Prvních 6 rodin (označené jako defaults) je definováno v OFS. Teprve další rodiny jsou deklarovány v použitých souborech.

Vedle názvů rodin jsou uvedeny přepínače variant, které pro danou rodinu můžeme použít. První čtyři přepínače jsou pro standardní varianty a pokud v některé rodině taková varianta není přístupná, je zde na místě odpovídajícího přepínače pomlčka. Pátý a případně další přepínače se vztahují k případným nadstandardním variantám. Zde například rodina Helvetica má navíc přepínače pro varianty „Narrow“ a rodiny Jannon mají přepínače pro varianty „Medium“.

\fontusage

Napíšeme-li do dokumentu příkaz `\fontusage`, zobrazí se na terminál a do logu stručná informace o použití příkazů z balíku OFS.

Kromě výše uvedeného způsobu zavedení:

PLAIN: `\input ofs [`⟨*soubor*⟩`, `⟨*soubor*⟩`, ...]`
LATEX: `\usepackage [`⟨*soubor*⟩`, `⟨*soubor*⟩`, ...] {ofs}`

existuje ještě možnost přímo zavést požadované soubory. Pak není nutné v dokumentu explicitně zavádět soubor `ofs.tex` resp. `ofs.sty`:

PLAIN: `\input` ⟨*soubor*⟩ `\input` ⟨*soubor*⟩ `...`
LATEX: `\usepackage {`⟨*soubor*⟩`} \usepackage {`⟨*soubor*⟩`} ...`

Příklad:

PLAIN: `\input sjannon \input sdynamo`
LATEX: `\usepackage {sjannon} \usepackage {sdynamo}`

Nedoporučuje se (zvláště v LaTeXu) oba tyto způsoby zavedení makra OFS míchat.

2.2. Příkaz `\setfonts`

`\setfonts`

V dalších příkladech předpokládáme, že jsme zavedli pouze potřebné rodiny, například ze souborů `sjannon` a `sdynamo`. Nyní třeba po použití příkazu

```
\setfonts [JannonText/12pt]
```

lze přepínat do jednotlivých variant této rodiny JannonText ve velikosti 12pt. Je možné použít přepínače `\rm`, `\bf`, `\it`, a `\bi` a speciálně pro tuto rodinu ještě `\mr` a `\mi`.

Příkaz `\setfonts [`⟨*JménoRodiny*⟩`/`⟨*velikost*⟩`]` přepíná do nové rodiny a ctí naposledy zapnutou variantu. Pokud před tímto příkazem byla zapnutá třeba varianta BoldItalic, pak po tomto příkazu zůstává zapnutá tato varianta i v nové rodině. Pokud v nové rodině naposledy zapnutá varianta neexistuje, přepíná se do `\rm`. Tato varianta musí být deklarována v každé rodině.

Příkaz `\setfonts` vymezuje všechny změny pouze lokálně. Po ukončení skupiny se sazba vrací k rodině a variantě, která byla aktuální při zahájení skupiny.

Parametry příkazu `\setfonts` mohou být prázdné: `\setfonts [`⟨*JménoRodiny*⟩`/]` přepne do nové rodiny a ctí naposledy nastavenou velikost a `\setfonts [/`⟨*velikost*⟩`]` nastaví novou velikost a ponechá aktuálně vybranou rodinu. Po inicializaci OFS je (pro případ použití prázdného parametru) defaultně nastavena rodina `CMRoman` a velikost 10 pt. Příkaz `\setfonts [/]` je syntakticky korektní, ale neudělá kromě zápisu do logu vůbec nic.

Parametr ⟨*JménoRodiny*⟩, je-li uveden, musí přesně odpovídat jménu rodiny podle seznamu známých rodin (viz nahoře příkaz `\showfonts`). Je třeba dodržovat malá a velká písmena a celé jméno psát bez mezer. Pokud se tento parametr neshoduje s žádnou známou rodinou, vypíše TEX varování a připojí seznam všech známých rodin. Takže například `\setfonts [?/]` lze použít se stejným efektem, jako příkaz `\showfonts`.

LATEX: ⟨*JménoRodiny*⟩ může být nejen jméno podle seznamu známých rodin (dlouhý název), ale je dovoleno použít i zkratku z NFSS. Takže například `\setfonts [Times/]` a `\setfonts [ptm/]` je v LaTEXu totéž.

PLAIN+LATEX: Parametr ⟨*velikost*⟩ má tyto možnosti:

- ⟨*číslo*⟩ .... např. `12`, `17.4`,
- ⟨*číslo*⟩⟨*jednotka*⟩ .... např. `12pt`, `17.4pt`, `10dd`,
- `at`⟨*číslo*⟩⟨*jednotka*⟩ .... např. `at12pt`, `at17.4pt`, `at10dd`,
- `scaled`⟨*celé číslo*⟩ .... např. `scaled1200`, `scaled\magstep3`,
- `mag`⟨*desetinné číslo*⟩ .... např. `mag1.2`, `mag.7`, `mag2.0`.

První tři možnosti se významově shodují. Klíčové slovo `at` je nepovinné a není-li uvedeno ani `at` ani jednotka, doplní se automaticky `at`⟨*číslo*⟩`pt`. Font se zavede přesně v požadované velikosti. Samozřejmě při globálním `\magnification` celého dokumentu se jedná o relativní a nikoli absolutní rozměry, pokud nepoužijeme jednotku uvozenou slovem `true` (např. `17truept`). Jestliže píšeme klíčové slovo `at`, nesmíme vynechat jednotku. Nesprávně: `at12`, správně: `at12pt` nebo jenom `12`.

Čtvrtá možnost (`scaled`) je shodná s použitím slova `scaled` při zavádění fontu pomocí primitivu `\font`. Např. `scaled1200` je font, jehož základní velikost je pronásobena koeficientem 1,2. Je-li základní velikost fontu 10 pt (což je obvyklé), pak je `scaled1200` shodné s `at12pt`.

Poslední možnost (`mag`) pronásobí aktuální velikost fontů daným koeficientem. Na rozdíl od `scaled` se tedy vztahuje k aktuální velikosti a nikoli k základní velikosti fontu.

Je-li nejprve použito třeba `\setfonts[/12]` a v zápětí `\setfonts[/mag2.]`, rodina fontů bude nyní sázena ve velikosti 24 pt. Příklad:

```
\def\small{\setfonts[/mag.7]}
Text {\small je menší \small a menší \small a ještě menší} a tady je
zase v normální velikosti.
```

Upozornění: Změna velikosti fontů pomocí `\setfonts` nijak nemění vzdálenost účaří (`\baselineskip`). To si musí uživatel nastavit sám.

Pomocí `\setfonts` můžeme nastavit nejen celou rodinu (to ovlivní přepínače variant `\rm`, `\bf`, `\it`, `\bi` a případně další), ale také jen zvolenou variantu v dané rodině. V takovém případě `\setfonts` nemění význam přepínačů variant ani aktuální velikost ostatních fontů. Při specifikaci varianty je nutno v parametru `\setfonts` za název rodiny napsat znak mínus následovaný zkratkou pro variantu. Příklady:

```
\setfonts [JannonText-it/12] ..... nastaví kurzívu dané rodiny ve
                                   velikosti 12pt
\setfonts [JannonText-rm/] ....... nastaví základní řez dané rodiny
                                   v aktuální velikosti.
\setfonts [CMTypeWriter-sl/] ..... nastaví základní variantu \sl
                                   v aktuální velikosti.
```

I při specifikaci varianty je možné v parametru vynechat název rodiny, pak se použije aktuální rodina. Například

```
\setfonts [JannonText/12]
\setfonts [-bf/17]     ... varinata Bold JannonText, velikost 17pt.
                           Přepínače \rm, \bf, \it a \bi zůstanou
                           nezměněny, v tomto příkladě přepínají
                           JannonText ve velikosti 12pt.
                           Následné použití třeba \setfonts [Times/]
                           nastaví rodinu Times ve velikosti 12pt.
ALE:
\setfonts [/17]\bf    ...  Varinata Bold aktuální rodiny, velikost 17pt.
                           Přepínače \rm, \bf, \it a \bi nyní
                           přepínají ve velikosti 17pt. Aktuální
                           velikost fontů je nyní 17pt.
```

LATEX: Co zde bylo řečeno o ponechání významu přepínačů variant při specifikování varianty v LaTeXu není pravda, protože by to narušilo logiku NFSS. V LaTeXu tedy je `\setfonts [-bf/17]` zcela shodné s příkazy `\setfonts [/17]\bf`.

2.3. Příkazy `\fontdef` a `\addcmd`

`\fontdef` Příkaz `\fontdef` umožňuje deklarovat nové fontové přepínače.

`\fontdef \`⟨*přepínač*⟩ `[`⟨*JménoRodiny*⟩`/`⟨*velikost*⟩`]`

Tato deklarace se zhruba shoduje s

`\gdef \`⟨*přepínač*⟩ `{\setfonts [`⟨*JménoRodiny*⟩`/`⟨*velikost*⟩`]}`

PLAIN: Pokud je specifikována rodina včetně varianty a parametr ⟨*velikost*⟩ není prázdný ani není zadán pomocí klíčového slova `mag`, pak `\`⟨*přepínač*⟩ není implementován jako makro obsahující `\setfonts`, ale jedná se o nativní přepínač fontu implementovaný pomocí `\global\font\`⟨*přepínač*⟩ (tzv. fixed font). Tak může uživatel deklarovat vlastní nativní přepínač fontu bez znalosti názvu metriky.

LATEX: Deklarovaný přepínač je ve všech případech implementován jako makro obsahující `\setfonts`. Přístup k nativním přepínačům je před uživatelem v NFSS skryt.

PLAIN+LATEX: Místo jména rodiny může být uveden vykřičník. Pak se doplní jméno rodiny aktuální v místě příkazu `\fontdef`. Na druhé straně, pokud je jméno rodiny prázdné, pak se jméno rodiny doplní podle aktuální rodiny v místě použití deklarovaného přepínače. Analogická vlastnost platí pro parametr ⟨*velikost*⟩. Příklady:

```
\setfonts [JannonAntikva/]
\fontdef \small  [/7]               % \small = \setfonts [/7pt]
\fontdef \sffam  [DynaGroteskR/]    % \sffam = \setfonts [DynagroteskR/]
\fontdef \velky  [Times/17]         % \velky = \setfonts [Times/17pt]
\fontdef \ttfam  [Courier/]         % \ttfam = \setfonts [Courier/]
\fontdef \mylogo [Times-rm/mag.8]   % \mylogo = \setfonts [Times-rm/mag.8]
                                    % velikost fontu bude vždy rovna
                                    % 0.8 násobku aktuální velikosti.
\fontdef \timbf  [Times-bf/12]      % \timbf = fixed-font přepínač, jako:
                                    % \global\font\timbf=ptmb8z at12pt
\fontdef \jansmall [!/7]            % \jansmall=\setfonts[JannonAntikva/7]
\fontdef \janbi [!-bi/17]           % \janbi = fixed-font přepínač, jako:
                                    % \global\font\janbi=sjnbi8z at17pt
\fontdef \tt [Courier-rm/!]         % \tt = fixed-font přepínač, jako
                                    % \global\font\tt=pcrr8u at10pt
```

Příkaz `\fontdef` deklaruje přepínač globálně, ovšem tento přepínač je sám o sobě lokální. Jinými slovy `\global\fontdef\jmeno` je totéž jako `\fontdef\jmeno`, ale přepínač `\jmeno` nastaví font (nebo celou rodinu) jen lokálně.

`\addcmd` Od verze OFS Oct. 2002 je podporován příkaz `\addcmd`, který umožní společně s příkazem `\fontdef` soustředit problematiku fontů do jediného místa v souboru maker. Příkaz má formát:

`\addcmd \`⟨*přepínač*⟩ `{`⟨*příkazy*⟩`}`

a chová se, jako `\def\`⟨*přepínač*⟩ `{`⟨*původní význam přepínače*⟩ ⟨*příkazy*⟩`}`. Pomocí `\addcmd` tedy můžeme „rozšířit“ obsah makra `\`⟨*přepínač*⟩ o další příkazy. Sekvence `\`⟨*přepínač*⟩ musí být (před použitím `\addcmd`) definována jako makro bez parametrů nebo musí být ve významu neexpandované kontrolní sekvence deklarované například pomocí `\font`, `\chardef` apod. Po použití `\addcmd` je sekvence `\`⟨*přepínač*⟩ definována vždy jako makro bez parametru. Opakované použití `\addcmd` na stejný `\`⟨*přepínač*⟩ je možné.

Příklad použití:

```
\setfonts [JannonText/]
\fontdef \footnotefont  [!/7]
\addcmd  \footnotefont  {\rm \baselineskip=9pt \relax}
\fontdef \sectionfont  [!/12]
\addcmd  \sectionfont  {\bf \let\it=\bi}
```

2.4. Test přítomnosti rodiny

PLAIN: Ve svých makrech můžeme testovat, zda je rodina fontů deklarovaná, tj. zda je načtena její deklarace z deklaračního souboru. Použijeme k tomu konstrukci `\knownfam` ⟨*JménoRodiny*⟩`? \iftrue`, která expanduje na `\iftrue`, pokud je rodina deklarovaná a na `\iffalse`, pokud deklarovaná není. Parametr ⟨*JménoRodiny*⟩ musíme uvést bez specifikace varianty.

`\ifknownfam` PLAIN+LATEX: Z důvodů zpětné kompatibility se starší verzí OFS dělá stejnou práci jako `\knownchar` makro `\ifknownfam [`⟨*JménoRodiny*⟩`]`. V OFS pro plain se ale od verze Feb. 2004 doporučuje používat `\knownfam`, protože to udržuje správně spárované primitivy `\if*`, `\else`, `\fi`. LaTeXový uživatel si může `\knownfam` snadno definovat.

2.5. LATEX: Propojení OSF s NFSS

`\OFSfamily` Tato sekce je určena pouze pro LaTeXisty. Příkaz

`\OFSfamily [`⟨*JménoRodiny*⟩`]`

konvertuje dlouhý název rodiny podle katalogu na interní název rodiny zanesený do NFSS. Například

```
\OFSfamily [JannonText]
```

expanduje na `sjng`. Makro pracuje jen na úrovni expandprocesoru, proto nedostaneme při překlepu v názvu rodiny chybové hlášení. Při neexistující rodině makro expanduje na text `unknown`. Použijeme-li například `\OFSfamily` ve svých stylových souborech a zpozorujeme, že se NFSS snaží substituovat rodinu `unknown`, můžeme si být jisti, že máme překlep v názvu rodiny nebo nemáme správně použité `\usepackage`.

Příklad použití:

```
\usepackage [sjannon, sdynamo] {ofs}
\edef\rmdefault {\OFSfamily [JannonAntikva]}
\edef\sfdefault {\OFSfamily [DynaGroteskR]}
\edef\ttdefault {\OFSfamily [Courirer]}
```

Význam maker `\rmdefault`, `\sfdefault`, `\ttdefault` je popsán v dokumentaci NFSS.

`\OFSfamilydefault` Dále OFS definuje příkaz

`\OFSfamilydefault [`⟨*JménoRodiny*⟩`]`

který nastaví základní rodinu celého dokumentu. Tato rodina se v příslušných velikostech a variantách objeví nejen v textu, ale i v záhlaví kapitol, sekcí a podobně (jsou-li makra v použitém class souboru udělána rozumně). Příkaz interně provede

`\edef\familydefault {\OFSfamily [`⟨*JménoRodiny*⟩`]}`

s tím, že navíc ošetří případ neexistující rodiny chybovým hlášením s výpisem všech aktuálně dostupných rodin.

2.6. Kódování fontů

LATEX: Přepínání mezi kódováním fontů probíhá zcela v režii NFSS. OFS v této věci nepřidává nic nového.

\fotenc PLAIN (až do konce sekce): Implicitně se předpokládá kódování CSfontů. Pokud ale potřebujeme použít fonty například v kódování T1, pak použijeme `\def\fotenc{8t}` a OFS bude pracovat s fonty v tomto kódování.

Můžeme dokonce uvnitř dokumentu přepínat:

```
\def\fotenc{8z} \setfonts[/] ... fonty v kódování CSfontů
\def\fotenc{8t} \setfonts[/] ... fonty v kódování podle Corku.
```

Kromě toho OFS obsahuje nástroje, aby makra závislá na kódování fontů (například `\v`, `\'`, `\ae`) expandovala správně na znak ve zvoleném kódování. Implicitně je
\loadingenc v OFS nastaveno `\loadingenc=0`, což znamená, že změnou kódování fontů ani příkazem `\setfonts` se nemění makra typu `\v`, `\ae`. Tato makra si ponechají svůj originální význam z plainu. To uvítají ortodoxní plainisté, kteří nemají rádi, když má použitý balíček zbytečně velkou inteligenci.

Je-li ale na začátku dokumentu uvedeno `\loadingenc=1`, například:

```
\input ofs [a35,sjannon]  \loadingenc=1
```

pak při každém příkazu `\setfonts` si TEX zkontroluje, zda má načteny všechny potřebné soubory se jménem `ofs-`⟨*kódování*⟩`.tex`. Tyto soubory obsahují předefinování maker závislých na nastaveném kódování fontů. Pokud tyto soubory načteny nejsou, TEX je během `\setfonts` načte. Podrobněji viz sekce 3.3 až 3.5.

Balíček OFS obsahuje tři soubory s deklaracemi maker závislými na kódování: `ofs-8z.tex`, `ofs-8t.tex` a `ofs-8c.tex`. Při použití jiného kódování fontů si můžete vytvořit další soubor analogického názvu. Příkazy `\accentdef` a `\characterdef` používané v těchto souborech jsou podrobně vysvětleny v sekci 3.4.

2.7. Fonty v matematice

LATEX: OFS pro LaTEX vůbec neřeší otázku fontů pro matematiku. Je potřeba využít nabídku možností NFSS.

PLAIN (až do konce sekce): Příkaz `\setfonts` a přepínače deklarované pomocí `\fontdef` přepínají jen písmo v textovém režimu. Pokud nepoužijeme příkaz `\setmath`, zůstává vše, co je ve vstupním souboru mezi dolary, vytištěno v Computer Modern ve velikostech 10 pt/7 pt/5 pt (velikost: základní/indexová/indexindexová). To nemusí být vždy žádoucí.

\setmath Příkaz `\setmath` má tento tvar:

`\setmath [`⟨*základní velikost*⟩`/`⟨*indexová velikost*⟩`/`⟨*indexindexová velikost*⟩`]`

V parametrech příkazu dáváme najevo, v jakých velikostech chceme matematické fonty nastavit. Je možné použít libovolný zápis velikosti, jako u příkazu `\setfonts`. Je-li použita možnost `mag`⟨*číslo*⟩, je velikost vypočítána jako ⟨*číslo*⟩ krát velikost aktuálního textového fontu. Je-li parametr prázdný, doplní se takto:

- základní velikost — `mag1.0`
- indexová velikost — `mag0.7`
- indexindexová velikost — `mag0.5`

Takže `\setmath [//]` udělá totéž jako `\setmath [mag1./mag.7/mag.5]`. OFS de-
\setsimplemath finuje makro `\setsimplemath` jako `\setmath [//]`.

Příkaz `\setmath` nastaví matematické fonty podle aktuální situace. Jeho činnost závisí na obsahu maker `\fomenc` a `\mathversion`.

\fomenc Matematické kódování je určeno hodnotou makra `\fomenc`. Existují tyto možnosti:

- Implicitní hodnota je `\def\fomenc{PS}` (PostScript-Symbol). Příkaz `\setmath` pak převezme kurzívu aktuální rodiny a použije ji jako matematickou kurzívu. Dále převezme základní variantu aktuální rodiny a použije ji pro číslice a další symboly, které jsou psány i v matematice jako `\rm`. Standardní přepínače variant `\rm`, `\it`, `\bf` a `\bi` začnou přepínat do odpovídajících fontů i v matematické sazbě. Při `\def\fomenc{PS}` se navíc chybějící matematické znaky, které obvykle nejsou přítomny v PostScriptových fontech (například písmena řecké abecedy), nahradí z PostScriptového fontu Symbol. Tento font se totiž ke většině PostScriptovým písmům hodí daleko lépe než Computer Modern. Ostatní znaky (například zvětšující závorky a velké operátory) zůstávájí v Computer Modern.
- Je-li `\def\fomenc{CM}` (Computer Modern), pak `\setmath` ponechá matematickým vzorečkům Computer Modern fonty, jen se případně přizpůsobí požadované velikosti.
- Po zavedení souboru `amsfn.tex` je možné použít `\def\fomenc{AMS}`. Matematika se pak chová stejně jako při `CM`, ale navíc jsou k dispozici veškeré symboly z AMSTEXu.
- Po zavedení souboru `txfn.tex` můžeme použít dvě nová kódování: `\def\fomenc{TX}` nebo `\def\fomenc{PX}`. V obou případech se pro matematiku použijí volně dostupné TXfonty, které vycházejí z řezů rodin Times a Helvetica. Při hodnotě `TX` se v matematice použijí výhradně TXfonty, zatímco při hodnotě `PX` budou TXfonty kombinovány s kurzívou a základním řezem aktuální textové rodiny fontů (podobně jako při kódování `PS`). OFS podporuje všechny kontrolní sekvence na matematické symboly podle manuálu k TXfontům. Jsou zde vešekré symboly z AMSTEXu a mnoho dalších. Symbolů z TXfontů je několik stovek.
- Po zavedení souboru `mtfn.tex` je možné použít `\def\fomenc{MT}`. Matematika bude obsahovat kurzívu a základní řez aktuální rodiny fontů kombinovanou se znaky komerční verze matematických fontů MathTimes.

Podobně jako v NFSS jsou podporovány dvě verze vzorečků: normal a bold. Požadovaná verze se nastavuje v makru `\mathversion` pomocí `\def\mathversion{normal}` nebo `\def\mathversion{bold}`. Makro `\setmath` pak přizpůsobí zavedení matematických fontů požadované verzi. Implicitně je nastaveno `\def\mathversion{normal}`. Nastavení `\mathversion` je potřeba provést před vyvoláním příkazu `\setmath`. Příklady:

\mathversion

```
\setmath [//] $vzoreček$      % vzoreček je ve verzi "normal"
$\def\mathversion{bold}\setmath[//] vzoreček$ % vzoreček ve verzi "bold"
```

3. Jak je to uděláno, aneb pohled do hloubky

LATEX: Všechna makra, která jsou jen pomocná, jsou ve stylu `ofs.sty` definována se jménem `\ofs@jménomakra`, aby nedošlo ke zmatení s jinými styly. Makra, která se používají v deklaračních souborech, mají název `\OFSjménomakra`. Dále styl definuje uživatelské příkazy `\setfonts`, `\fontdef`, `\showfonts`, `\fontusage`, a předefinovává přepínače `\rm`, `\bf`, `\it`, `\bi`.

PLAIN: Všechna makra definovaná v `ofs.tex` jsou uvedena v rejstříku na konci tohoto dokumentu. Při návrhu názvů maker jsem nevyužil konvenci se znakem `@`, protože tento znak v názvech maker osobně nenávidím.

3.1. Výpisy do logu

LATEX: Pro trasování makra NFSS se použije balíček `tracefnt`.

PLAIN: Pokud někoho ruší při častém přepínání rodin fontů příkazem `\setfonts` výpisy v logu, může použít příkaz `\nofontmessages`. Pokud naopak chceme tyto výpisy vidět i na terminálu, pišme `\displayfontmessages`. Implicitně je zapnut výpis pouze do logu příkazem `\logfontmessages`. Pokud chceme přesně vidět, které metriky fontů se zavádějí primitivním příkazem `\font`, použijeme `\detailfontmessages`.

\nofontmessages
\displayfontmessages
\logfontmessages
\detailfontmessages

Varování o nepřístupných znacích nebo kódováních se vždy vypisují na terminál i do logu. Chceme-li je mít jen v logu, můžeme napsat `\let\displaymessage=\wlog`, protože tato sekvence je v OFS používána na zobrazení zpráv na terminál.

\displaymessage

3.2. Robustní a křehké příkazy

LATEX: LaTEX2e má zavedené postupy na definování robustních příkazů. Význam těchto pojmů je vysvětlen v dokumentaci k LaTEXu. Příkaz `\setfonts` a přepínače deklarované pomocí `\fontdef` jsou samozřejmě definovány jako robustní, takže se mohou vyskytovat v textech pro obsahy, rejsříky apod. bez toho, aby způsobily obtíže.

PLAIN (až do konce sekce): Makro plainu problematiku křehkých příkazů neřeší a uživatelé plainu, pokud narazí na tento problém, používají různá svá vlastní řešení, která nejsou standardizována. Jedno takové řešení je i v OFS.

Nejprve stručně naznačíme, v čem spočívá problém s křehkými příkazy. Občas potřebujeme (pro generování obsahu, rejstříku apod.) poslat nějaký text do pomocného souboru (primitivem `\write`), aby byl tento soubor v zápětí (při dalším zpracování TEXem) znovu načten. Problém je, že primitiv `\write` zapíše do souboru text po expanzi maker, což nemusí vždy vyhovovat. Pokud je například fontový přepínač implementován jako složité makro a je použit v textu zpracovaném primitivem `\write`, v souboru se objeví celé makro po expanzi. Při opětovném načtení takového souboru dojde většinou k havárii; říkáme, že byl použit křehký (fragile) příkaz v textu posílaném do souboru a v souboru se nám tento příkaz „rozsypal“.

Pokud se dostane (potenciálně) křehký příkaz z maker OFS do souboru, pak se při novém načtení takového souboru automaticky při havárii zobrazí na terminálu a v logu návod jak postupovat, aby se uživatel tohoto problému zbavil. Text návodu po překladu do češtiny vypadá takto:

```
CHYBA !! křehký příkaz v toc/ind/aux nebo podobném souboru.
Tento problém můžete řešit následujícími kroky:
1. Vymažte pomocný soubor s tímto křehkým příkazem.
2. Vložte následující kód do záhlaví svého dokumentu:
      \let\orishipout=\shipout
      \def\shipout#1#2{\setbox0=#1{#2}\bgroup
           \let\expandaction=\noexpand \orishipout\box0 \egroup}
3. Znovu TeXujte svůj dokument nejméně dvakrát za sebou.
Další informace najdete v dokumentaci k OFS.
```

Vysvětlíme si, jak je to uděláno. Makro, které může v parametru primitivu `\write` působit potíže, je v OFS definováno zhruba takto:

```
\def\makro {%
   \ifx\expandaction\noexpand
      \noexpand\makro
```

```
    \else
       \csname fragilecommand!\endcsname
       ⟨vlastní kód makra⟩
    \fi
}
```

\expandaction \fragilecommand! Pokud se nenastaví `\expandaction` na hodnotu `\noexpand`, pak se provede část kódu za `\else`. Protože příkaz `\csname fragilecommand!\endcsname` není definován, realizuje se v hlavním procesoru jako `\relax` [TBN, strana 349]. Při expanzi primitivem `\write` se do souboru zapíše `\fragilecommand!`. Při dalším čtení soboru se spustí příkaz \fragilecommand `\fragilecommand`, který vypíše na terminál již zmíněnou nápovědu.

Jestliže uživatel použije kód z nápovědy v záhlaví svého dokumentu, pak v době činnosti primitivu `\shipout` (tj. v době, kdy expandují argumenty primitivů `\write`) je nastaveno `\let\expandaction=\noexpand`. V makru `\makro` se provede první část před `\else`, takže do souboru se zapíše jen `\makro`. To je přesně to, co potřebujeme.

V OFS není kód, který předefinovává `\shipout`, přímo zahrnut. Místo toho pouze doporučíme uživateli, aby to udělal sám. Uživatel plainu totiž potřebuje mít věci pod vlastní kontrolou a neměl by asi důvěru k makru, které bez jeho vědomí předefinuje tak zásadní věc, jako je primitiv `\shipout`. Také je možné, že uživatel plainu použije na různé části textu ve svých makrech `\edef` a potřebuje vědět, že je v takovém případě vhodné nejpve nastavit `\let\expandaction=\noexpand`.

Podobně jako zde v ukázce definice příkazu `\makro` jsou v OFS definovány příkazy `\setfonts` (sekce 2.2), `\setmath` (sekce 2.7), `\setextrafont`, `\printcharacter`, `\printaccent` (sekce 3.4), `\accentabove`, `\accentbelow` a `\ofshexbox` (sekce 3.6). Po použití kódu z nápovědy se tato makra stávají „robustními“ v LaTeXovém smyslu tohoto slova.

3.3. Deklarační soubory

LATEX: Deklarační soubory v LaTeXu jsou běžné stylové soubory, které shrnují několik rodin fontů ve smyslu NFSS. Tyto rodiny jsou deklarovány běžným způsobem ve `fd` souborech. V deklaračních souborech je možno použít následující příkazy:

\OFSprocessoptions
- `\OFSprocessoptions` — má většinou hodnotu `\undefined`. Pokud ale je stylový soubor načítán ze souboru `ofs.sty` jako parametr, má `\OFSprocessoptions` hodnotu `\relax`. Test na tuto sekvenci používáme v záhlaví stylového souboru, abychom vynachali LaTeXovské `\RequirePackage{ofs}`, pokud to není potřeba (a pouze by to zlobilo).

\OFSextraencoding
- `\OFSextraencoding {`⟨*rozšiřující kódování*⟩`}` — Toto makro si poznamená do paměti ⟨*rozšiřující kódování*⟩ a provede `\input {`⟨*rozšiřující kódování*⟩`ini.def}`. Předpokládá se, že tam jsou odpovídající definice pro ⟨*rozšiřující kódování*⟩, viz například soubor `se1ini.def`, který obsahuje deklarace pro rozšiřující kódování SE1 fontů ze Střešovické písmolijny. Pokud už byl soubor ⟨*rozšiřující kódování*⟩`ini.def` načten, makro jej nenačítá znovu. Pozor: ⟨*rozšiřující kódování*⟩ je nutno psát velkými písmeny, ačkoli v názvu souboru musí být naopak tato písmena malá.

\OFSputfamlist
- `\OFSputfamlist {`⟨*text*⟩`}` — vloží ⟨*text*⟩ do seznamu rodin, který se vypisuje při `\showfonts` nebo při neznámé rodině.

\OFSdeclarefamily
- `\OFSdeclarefamily [`⟨*JménoRodiny*⟩`] {`⟨*NFSS-jméno*⟩`}` — poznamená si do paměti, že ⟨*JménoRodiny*⟩ je ve skutečnosti ⟨*NFSS-jméno*⟩. Tato paměť se využije

například v příkazu `\OFSfamily`. Navíc zanese ⟨*JménoRodiny*⟩ do seznamu rodin vypisovaném při `\showfonts`.

`\OFSnormalvariants`
- `\OFSnormalvariants` — do textu, který se vypisuje při `\showfonts` nebo při neznámé rodině vloží, seznam standardních přepínačů, tj. `\rm, \bf, \it, \bi`.

PLAIN (až do konce sekce): Deklarační soubory mají příponu `tex` a předpokládá se o nich, že mají pojistku proti opakovanému načtení. Pokud není ještě načten soubor `ofs.tex`, je nutno jej před další činností načíst. Rovněž je vhodné zde načíst soubor definic znaků podle použitých kódování (viz sekce 3.4).

V deklaračním souboru je nutno vytvořit vazbu mezi názvy rodin a použitými metrikami. K tomu se používají následující příkazy:

`\protectreading`
- `\protectreading` ⟨*soubor*⟩⟨*mezera*⟩ — poznamená si do paměti, že je přečten ⟨*soubor*⟩. Pokud je příkaz se stejným parametrem čten podruhé, provede `\endinput`, takže následující deklarace jsou chráněny proti vícenásobnému čtení.

`\ofsputfamlist`
- `\ofsputfamlist {`⟨*text*⟩`}` — vloží ⟨*text*⟩ do seznamu rodin, který se vypisuje při `\showfonts` nebo při neznámé rodině.

`\ofsdeclarefamily`
- `\ofsdeclarefamily [`⟨*JménoRodiny*⟩`] {`⟨*příkazy*⟩`}` — deklarace nové rodiny se jménem ⟨*JménoRodiny*⟩. Toto jméno se zanese do seznamu známých rodin vypisovaných pomocí `\showfonts`. Pokud se tato rodina použije při `\setfonts`, pak se provedou ⟨*příkazy*⟩. Mezi ⟨*příkazy*⟩ se použije právě jeden příkaz `\loadtextfam`, který zavede fonty dané rodiny do paměti TeXu.

`\loadtextfam`
- `\loadtextfam` — Zavede čtyři fonty s danými metrikami. Tento příkaz vyžaduje podrobnější vysvětlení, které je včetně parametrů příkazu uvedeno níže.

`\newvariant`
- `\newvariant`⟨*číslice*⟩ `\`⟨*přepínač*⟩ `(`⟨*Varianta*⟩`)` ⟨*mezera*⟩ ⟨*metrika*⟩`;`⟨*extra-enc*⟩`;` — Deklaruje nový přepínač varianty. Podrobnější vysvětlení je uvedeno níže.

`\modifyenc`
- `\modifyenc` ⟨*kódování*⟩`:`⟨*identifikátor*⟩`;` — přidání výjimek vzhledem k základnímu kódování, viz sekce 3.5.

`\fosize`
- `\fosize` — makro, které uchovává informaci o aktuální velikosti fontů v naposledy vybrané rodině. Obsah `\fosize` může mít dvě podoby: `at`⟨*dimen*⟩ nebo `scaled`⟨*number*⟩ podle toho, jaký tvar parametru ⟨*velikost*⟩ byl použit v příkazu `\setfonts`.

`\fotenc`
- `\fotenc` — makro, které uchovává informaci o aktuálním kódování. Nejčastější možnosti jsou `\def\fotenc{8z}` (kódování podle CS-fontů) nebo `\def\fotenc{8t}` (kódování podle Corku). Je-li při načtení souboru `ofs.tex` makro `\fotenc` už známé, ponechá se beze změny, jinak se definuje jako `8z`.

`\extraenc`
- `\extranec` — makro, které uchovává informaci o rozšiřujícím kódování. Tuto informaci tam kopíruje z parametru ⟨*extra-enc*⟩ makro `\laodatextfam`.

`\defaultextraenc`
- `\defaultextraenc` — makro, jehož předefinováním můžeme změnit rozšiřující kódování základních rodin a rodin z `a35.tex`. Výchozí hodnota makra `8c`.

`\setfontshook`
- `\setfontshook` — makro, které se spustí vždy při činnosti příkazu `\setfonts` těsně před tím, než jsou vykonány ⟨*příkazy*⟩ z `\ofsdeclarefamily`.

`\registertfm`
- `\registertfm` ⟨*jméno*⟩ ⟨*od*⟩`-`⟨*do*⟩ ⟨*skutečná metrika*⟩ — umožňuje pracovat při různých velikostech fontů s různými metrikami. Podrobnosti viz sekce 3.8.

`\registerenc`
- `\registerenc` ⟨*JménoRodiny*⟩`:` ⟨*kódování*⟩ ⟨*mezera*⟩ — umožňuje omezit použití rodiny fontů jen pro vymezená kódování. Podrobnosti viz sekce 3.9.

Příkaz `\loadtextfam` používaný v deklaračních souborech mezi ⟨*příkazy*⟩ makra `\ofsdeclarefamily` má parametry:

`\loadtextfam (`⟨*Varianta-rm*⟩`)` ⟨*mezera*⟩ ⟨*metrika-rm*⟩`;%`

```
(⟨Varianta-bf⟩) ⟨mezera⟩ ⟨metrika-bf⟩;%
(⟨Varianta-it⟩) ⟨mezera⟩ ⟨metrika-it⟩;%
(⟨Varianta-bi⟩) ⟨mezera⟩ ⟨metrika-bi⟩;⟨extra-enc⟩;%
```

Procenta na koncích řádků v této ukázce označují, že za středníky nemá být mezera. Údaje (⟨*Varianta-XX*⟩) slouží jako komentáře o variantách zapisované do logu a je možno je kompletně včetně závorek a ⟨*mezery*⟩ vynechat (třeba jen některé). Ve verzích OFS do Sep 2002 nebyl tento parametr vůbec podporován. Při vynechání tohoto parametru se předpokládají následující implicitní hodnoty: `rm: ()`, `bf: (Bold)`, `it: (Italic)`, `bi: (BoldItalic)`.

Údaje ⟨*metrika-XX*⟩ odpovídají názvům metrik, které se pro uvedené varianty fontů makrem `\loadtextfam` zavedou. Makro zhruba provede toto:

```
\font\tenrm=⟨metrika-rm⟩ \fosize
\font\tenbf=⟨metrika-bf⟩ \fosize
\font\tenit=⟨metrika-it⟩ \fosize
\font\tenbi=⟨metrika-bi⟩ \fosize
```

Předpokládá se, že všechny ⟨*metriky-XX*⟩ jsou v deklaračním souboru napsány tak, že obsahují makro `\fotenc`. Tím je zaručena změna názvu metriky při přechodu na jiné kódování (tj. při předefinování makra `\fotenc`).

Konečně údaj ⟨*extra-enc*⟩ udává název extra kódování. Pokud je tento parametr neprázdný, pak `\loadtextfam` provede na přechodnou dobu `\def\fotenc{`⟨*extra-enc*⟩`}` a znovu expanduje parametry ⟨*metrika-rm*⟩, ⟨*metrika-bf*⟩, ⟨*metrika-it*⟩ a ⟨*metrika-bi*⟩. Výsledky těchto expanzí si poznamená do paměti. Jedná se o rozšiřující metriky k základním metrikám. Je-li parametr ⟨*extra-enc*⟩ prázdný, poznámka o rozšiřujících metrikách se nekoná. Když pak nějaké makro pro přístup ke speciálnímu znaku tuto metriku potřebuje, objeví se na terminálu varování, že aktuálně zvolená metrika nemá rozšiřující variantu.

Některé metriky v příkazu `\loadtextfam` s výjimkou ⟨*metrika-rm*⟩ mohou být vynechány. Je-li například prázdný parametr ⟨*metrika-XX*⟩, pak se zhruba provede

```
\def\tenXX{\message{varování o neexistující variantě na terminálu}}
```

takže při přechodu na takovou variantu se objeví na terminálu a v logu hlášení o neexistující variantě a font se nezmění.

`\setfonts` Příkaz `\setfonts` nemění význam maker `\rm`, `\bf`, `\it` a `\bi`, ale mění význam přepínačů fontů `\tenrm`, `\tenbf`, `\tenit` a `\tenbi`. Dělá to navíc zprostředkovaně tím, že spustí ⟨*příkazy*⟩ deklarované pro požadovanou rodinu v `\ofsdeclarefamily` a mezi těmito ⟨*příkazy*⟩ se vyskytuje příkaz `\loadtextfam`.

`\tenrm \tenbf \tenit \tenbi \currentvariant` Makra `\rm`, `\bf`, `\it`, resp. `\bi` používají fontové přepínače `\tenrm`, `\tenbf`, `\tenit`, resp. `\tenbi`. První tři jsou známy z plainu a čtvrtý je zaveden nově. Navíc makra `\rm`, `\bf`, `\it` a `\bi` ukládají do kontrolní sekvence `\currentvariant` informaci o naposledy aktivované variantě. Tato informace má podobu písmene (`M`, `F`, `T`, resp. `I`), protože jedině konstrukce `\let\currentvariant=`⟨*písmeno*⟩ je odolná vůči expanzi. Šlo o to, aby makra přepínačů variant byla robustní i bez předefinování `\shipout`.

Všimneme si, že `\loadtextfam` nastavuje přepínače `\tenrm`, `\tenbf`, `\tenit` a `\tenbi` na fonty v libovolné velikosti. Činí tak podle aktuální hodnoty makra `\fosize`, které mění makro `\setfonts` podle toho, jak je zadán parametr ⟨*velikost*⟩. Slovo „`ten`" v názvech fontových přepínačů tedy nemusí znamenat, že se jedná o font ve velikosti

10 pt. Považujme tyto názvy za historické a z hlediska OFS možná trochu kuriózní. Není přitom nutné pro přepínače textových fontů vymýšlet názvy nové.

Někdo může namítnout, že opakované používání makra `\setfonts` vždy znovu a znovu spouští čtveřici příkazů `\font`, což může způsobit větší nároky na paměť a čas. Není to pravda, protože TEX si eviduje interní tabulku už zavedených metrik a při opakovaném použití primitivu `\font` na stejnou metriku se už neobtěžuje metriku vůbec otevírat. Jen inicializuje příslušný fontový přepínač asi stejně rychle, jako by se to provedlo pomocí `\let`.

`\loadingenc` Příkaz `\loadtextfam` ještě před zavedením fontů načte při `\loadingenc>0` soubor `ofs-\fotenc.tex`. Při neprázdném parametru ⟨*extra-enc*⟩ navíc načte soubor `ofs-`⟨*extra-enc*⟩`.tex`. Tyto soubory jsou čteny jen jednou a v době čtení jsou ignorovány prázdné řádky, konce řádků. Čtení probíhá uvnitř skupiny, kde jsou pro jistotu lokálně nastaveny kategorie znaků podle plainTEXu (a `` \catcode`@=11 ``) a je nastaveno `\globaldefs=1`, tj. všechna předefinování ze souboru `ofs-`⟨*kódování*⟩`.tex` se provedou globálně. To nám nevadí, protože nově načtené kódovací soubory nejsou v konfliktu s předchozími (viz vlastnosti příkazů `\characterdef` a `\accentdef` v sekci 3.4). Načtení většího množství těchto souborů než je v dané chvíli potřeba, rovněž neohrozí funkcionalitu maker závislých na kódování. Důvod globální definice je v tom, že není vhodné, aby při opakovaném `{...\setfonts...}` musel být kódovací soubor načítán neustále znovu a znovu.

Pokud uživateli vadí, že je předefinování maker globální (chce například svůj článek zařadit do sborníku, kde by předefinování mohlo vadit jiným článkům), pak ponechá `\loadingenc=0` a musí načíst soubory s deklaracemi maker manuálně pomocí `\input` na začátku článku. V takovém případě se provede deklarace jen lokálně.

V parametru ⟨*příkazy*⟩ makra `\ofsdeclarefamily` se může deklarovat nadstan-
`\newvariant` dardní varianta rodiny pomocí příkazu `\newvariant`. Tento příkaz provede:

```
\font\ten⟨přepínač⟩=⟨metrika⟩ \fosize
\def \⟨přepínač⟩ {\let\currentvariant=⟨číslice⟩ \ten⟨přepínač⟩}
```

Dále příkaz `\newvariant` zanese do paměti informaci o případné rozšiřující metrice podle ⟨*ext-enc*⟩.

Pokud OFS potřebuje dodržet při přechodu na novou rodinu naposledy zvolenou nadstandardní variantu, činí tak podle hodnoty `\currentvariant`. Pokud tedy nová rodina má deklarovánu nadstandardní variantu pod stejnnou ⟨*číslicí*⟩, bude použita tato varianta a OFS nebude přepínat do defaultní varianty `\rm`. Je proto vhodné deklarovat varianty různých rodin ale „stejného typu“ pod stejnou ⟨*číslicí*⟩. Z principu věci vidíme, že v OFS existuje omezení v počtu nadstandardních variant jedné rodiny na maximálně 10. To by nemělo vadit, protože kdykoli můžeme rozdělit rozsáhlou rodinu na více rodin (například Štormovy rodiny DynamoGrotesk).

Makro `\setfonts` může měnit význam maker `\loadtextfam` a `\newvariant`. Udělá to právě tehdy, když je v parametru ⟨*Rodina*⟩ specifikovaná varianta. V takovém případě stačí zavést jeden font a není potřeba zavádět celou rodinu. Při specifikaci standardní varianty `\newvariant` nebude dělat nic a `\loadtextfam` zavede jediný font. Při specifikaci nadstandardní varianty `\loadtextfam` nebude dělat nic a `\newvariant` zavede font jen tehdy, pokud se jedná o font specifikované varianty.

`\setfonts` Na závěr této sekce si podrobněji vysvětlíme činnost příkazu `\setfonts`. Tento příkaz „vypočítá“ a definuje podle parametru ⟨*velikost*⟩ hodnotu makra `\fosize`. Dále

`\currentfamily`

při neprázdném pamaretru ⟨*JménoRodiny*⟩ a při neuvedení varianty tento příkaz provede `\def\currentfamily{`⟨*JménoRodiny*⟩`}`, zatímco při prázdném parametru hodnotu z `\currentfamily` pouze použije. Je-li v parametrech uvedena varianta rodiny, pozmění `\setfonts` na přechodnou dobu makra `\loadtextfam` a `\newvariant` tak, aby místo celé rodiny fontů byl zaveden jen jeden font. Dále spustí `\setfontshook` a pak ⟨*příkazy*⟩ z odpovídajícího `\ofsdeclarefamily`. Potom vrátí hodnoty maker `\loadtextfam` a `\newvariant` do původního stavu, pokud byly tyto hodnoty měněny.

`\runmodifylist`

Nakonec spustí makro `\runmodifylist`, které nastaví za jistých okolností výjimky ze zvoleného kódování (podrobněji viz sekci 3.5). Konečně příkaz `\setfonts` spustí `\ignorespaces`, aby případná zapomenutá mezera za závorkou „`]`“nebyla v horizontálním módu vytištěna.

3.4. Kódování fontů a deklarace znaků

`\setextrafont`

PLAIN: Pro přepínání do rozšiřující metriky slouží makro `\setextrafont`. Toto makro se podívá, zda k aktuálnímu fontu existuje v paměti poznámka o rozšiřující metrice (viz příkaz `\loadtextfam` a `\newvariant` v předchozí sekci). Pokud taková poznámka existuje, příkaz `\setextrafont` provede zhruba toto:

`\extrafont`

```
\font\extrafont=⟨rozšiřující metrika ve velikosti základní metriky⟩ \extrafont
```

LATEX: Pro přepínání do rozšiřujícího kódování, které je deklarováno příkazem `\OFSextraencoding`, slouží (stejně jako v plainu) makro `\setextrafont`. Toto makro se podívá, zda bylo rozšiřující kódování deklarováno, a pokud ano, provede zhruba toto:

```
\fontencoding{⟨rozšiřující kódování⟩}\selectfont
```

`\extchar`

PLAIN+LATEX: Chceme-li vytisknout z rozšiřující metriky (rozšiřujícího kódování) nějaký znak s kódem ⟨*číslo*⟩, je možné použít makro `\extchar` ⟨*číslo*⟩.

`\characterdef`

PLAIN (až do konce sekce): Pro definice maker závislých na aktuálním kódování je možné v OFS použít dva deklarační příkazy: `\characterdef` a `\accentdef`. První z nich má tyto parametry:

```
\characterdef \⟨sekvence⟩  ⟨kódování⟩ ⟨mezera⟩ ⟨číslo⟩
% příklad:
\characterdef \promile  8z  141
% nebo
\characterdef \⟨sekvence⟩  ⟨kódování⟩ ⟨mezera⟩ {⟨příkazy⟩}
% příklad:
\characterdef \promile  8t  {\%\char24 }
\characterdef \promile  *   {\vrule height1ex width1ex\relax}
                            % in another encodings
```

V prvním případě se bude \⟨*sekvence*⟩ při aktuálním kódování fontů shodném s ⟨*kódováním*⟩ expandovat na token kategorie 12 s kódem rovným ⟨*číslu*⟩. V druhém případě \⟨*sekvence*⟩ expanduje při shodnosti ⟨*kódování*⟩ na ⟨*příkazy*⟩. Vše proběhne na úrovni expand procesoru. Pokud použijeme oba výše zmíněné příklady pro definici `\promile`, pak platí:

```
\def\fotenc{8z} \promile % expanduje na znak s kódem 141
\def\fotenc{8t} \promile % expanduje na příkazy \%\char24
```

Mimoto je možné deklarovat přístup k rozšířenému kódování, pokud je toto kódování specifikováno jako parametr ⟨*extra-enc*⟩ v příkazu `\loadtextfam`. Příklad:

```
\characterdef \euro  8z  134
\characterdef \euro  6s  37

\def\fotenc{8z} \euro % expanduje na znak s kódem 134
\def\fotenc{8t} \euro % expanduje na: {\setextrafont znak s kódem 37}
```

V tomto příkladě se při `\def\fotenc{8t}` může stát, že aktuální font nemá poznámku o rozšiřující metrice, nebo má poznámku o rozšiřující metrice jiné než `6s`. Pak `\euro` vypíše varování na terminál o nedostupnosti znaku `\euro`.

`\printcharacter`

Vysvětlíme nyní podrobněji, jak makra deklarovaná pomocí `\characterdef` fungují. Příkaz `\characterdef` definuje `\`⟨*sekvenci*⟩ jako `\printcharacter{`⟨*sekvence*⟩`}`, takže například `\promile` expanduje nejprve na `\printcharacter{promile}`. a `\euro` na `\printcharacter{euro}`. Pokud se `\characterdef` použije opakovaně na stejnou `\`⟨*sekvenci*⟩, pak se vždy znovu definuje stejně, takže nedochází ke ztrátě předchozí informace. Kromě toho ale `\characterdef` definuje další makro `\`⟨*sekvence*⟩`:-`⟨*kódování*⟩ tak, že toto makro expanduje na znak se specifikovaným kódem nebo na ⟨*příkazy*⟩. Další inteligence je schována v příkazu `\printcharacter`. Ten ověří, zda je definováno makro `\`⟨*sekvence*⟩`:-\fotenc`. Pokud ano, expanduje na obsah tohoto makra. Jinak ověří, zda existuje poznámka o rozšiřující metrice vztažená k aktuálnímu fontu. Pokud ano, zjistí kódování této rozšiřující metriky ⟨*extra-enc*⟩ a podívá se, zda je definováno makro `\`⟨*sekvence*⟩`:-`⟨*extra-enc*⟩. Pokud ano, expanduje na

`{\setextrafont \`⟨*sekvence*⟩`:-`⟨*extra-enc*⟩`}`

`\printcharacterwarn`

Pokud se stále nepodařilo realizovat požadovaný znak, pokusí se vytisknout defaultní znak nezávislý na kódování `\`⟨*sekvence*⟩`:-*`. Pokud ani to není možné, spustí se `\printcharacterwarn{`⟨*sekvence*⟩`}`. Implicitní hodnota tohoto makra vypíše varování o nedostupném znaku ⟨*sekvence*⟩ na terminál a do logu a do výstupu nevytiskne nic.

Chceme-li se vyhnout výpisu varování, můžeme `\printcharacterwarn` předefinovat, třeba takto:

```
\def\printcharacterwarn #1{?(#1)?}
```

Od verze OFS Mar. 2004 `\characterdef` odmítá předefinovat už definované kontrolní sekvence. Definuje pouze sekvence, které mají zatím význam `\undefined` nebo `\relax`. V jiném případě (a pokud není sekvence už pomocí `\characterdef` definována) vypíše varování o ignorované definici do log souboru. Důvod tohoto opatření je ten, že kódovací soubory deklarují obrovské množství nových kontrolních sekvencí pomocí `\characterdef`, a přitom programátor maker nemusí všechny znát. Může se tedy stát, že použije pro své vlastní makro stejné jméno. V takovém případě `\characterdef` respektuje makro definované programátorem a příslušný znak ponechá nepřístupný. Pokud potřebujeme pomocí `\characterdef` skutečně některé kontrolní sekvence předefinovat (u maker závislých na kódování z plainu to je nutné), pak musíme před použitím `\characterdef` napsat `\let\`⟨*sekvence*⟩`=\relax`.

`\safelet` `\safeletwarn`

Ze stejných důvodů bylo do OFS přidáno makro `\safelet`, které se chová jako `\let`, ale odmítá předefinovat už definované kontrolní sekvence. Místo toho pouze vypíše varování, definované v makru `\safeletwarn`.

`\accentdef`

Kromě deklarace znaků pomocí `\characterdef` je k dispozici možnost deklarace akcentových příkazů prostřednictvím `\accentdef`, který má tyto parametry:

`\accentdef \`⟨*sekvence*⟩ ⟨*znak*⟩ ⟨*nepovinná mezera*⟩ ⟨*kódování*⟩ ⟨*mezera*⟩ ⟨*číslo*⟩
`% Příklad:`

```
\accentdef \v E  8z  204         % Ecaron
% nebo
\accentdef \⟨sekvence⟩ ⟨znak⟩ ⟨nep. mezera⟩ ⟨kódování⟩ ⟨mezera⟩ {⟨příkazy⟩}
% Příklad:
\accentdef \v *  8z  {\accent20 }  % default caron in 8z
\accentdef \v *  *   {\blackbox }  % default caron
```

Pokud je aktuální kódování fontů totožné s ⟨*kódováním*⟩, pak `\`⟨*sekvence*⟩ následovaná znakem ⟨*znak*⟩ expanduje na token kategorie 12 s kódem ⟨*číslo*⟩, respektive expanduje na ⟨*příkazy*⟩. To vše se děje jen na úrovni expand procesoru. Pokud je při deklaraci pomocí `\accentdef` jako ⟨*znak*⟩ uvedena hvězdička, pak se deklarované ⟨*příkazy*⟩ nebo kód ⟨*číslo*⟩ použije vždy, když znak následující `\`⟨*sekvenci*⟩ není nikde prostřednictvím `\accentdef` deklarován. Jedná se tedy o defaultní realizaci akcentu, pokud ve fontu neexistuje přímo akcentovaný znak.

Kromě toho mají akcenty deklarované pomocí `\accentdef` stejnou inteligenci týkající se možnosti využití rozšiřující metriky, jako znaky deklarované v `\characterdef`.

Vysvětlíme si nyní podrobněji, jak akcenty deklarované pomocí `\accentdef` fungují. Příkaz `\accentdef` definuje `\`⟨*sekvenci*⟩ jako makro s jedním neseparovaným

`\printaccent` parametrem `#1`, které v první řadě expanduje na `\printaccent{`⟨*sekvence*⟩`}{#1}`, takže například `\v E` expanduje na `\printaccent{v}{E}`. Dále `\accentdef` definuje makro `\`⟨*sekvence*⟩`:`⟨*znak*⟩`:-`⟨*kódování*⟩ jako znak s daným kódem ⟨*číslo*⟩ nebo jako ⟨*příkazy*⟩. Další práci udělá příkaz `\printaccent`. Tento příkaz zjistí, zda je definováno makro `\`⟨*sekvence*⟩`:`⟨*znak*⟩`:-\fotenc`. Pokud ano, expanduje na toto makro. Pokud ne, tak si ověří, zda je k aktuálnímu fontu poznamenána rozšiřující metrika a zjistí její kódování ⟨*extra-enc*⟩. Jestliže je definováno makro `\`⟨*sekvence*⟩`:`⟨*znak*⟩`:-`⟨*extra-enc*⟩, pak se expanduje na `{\setextrafont \`⟨*sekvence*⟩`:`⟨*znak*⟩`:-`⟨*extra-enc*⟩`}`. Pokud zatím vše selhalo, zkusí `\printaccent` postupně zjistit, zda jsou definována makra `\`⟨*sekvence*⟩`:*:-\fotenc` a `\`⟨*sekvence*⟩`:*:-`⟨*extra-enc*⟩. Pokud je definováno první z nich, expanduje a připojí na konec expanze ⟨*znak*⟩. Pokud je definováno jen druhé z nich, expanduje na

`{\setextrafont \`⟨*sekvence*⟩`:*:-`⟨*extra-enc*⟩ ⟨*normalfont*⟩ ⟨*znak*⟩`}`

kde ⟨*normalfont*⟩ označuje přepínač do aktuálního fontu. Jestliže dosud vše selhalo, pokusí se ještě o realizaci akcentu nezávislého na kódování, tj. pokusí se tisknout `\`⟨*sekvence*⟩`:`⟨*znak*⟩`:-*` nebo konečně `\`⟨*sekvence*⟩`:*:-*` ⟨*znak*⟩. Teprve když i toto

`\printaccentwarn` selhalo, spustí se `\printaccentwarn{`⟨*sekvence*⟩`}{`⟨*znak*⟩`}`. Implicitní hodnota tohoto makra vypíše se na terminál a do logu varování o nedostupnosti akcentovaného znaku a do výstupu nevytiskne nic.

Je třeba si uvědomit, že přechod ke znaku v rozšiřující metrice uvnitř slova znamená ztrátu vyrovnávacího kernu kolem tohoto znaku a nemožnost použít na takové slovo vzory dělení slov. Proto doporučujeme použít základní metriku kódovanou tak, aby přechod na rozšiřující metriku byl jen výjimečný. Pro český a slovenský jazyk jsou v tomto smyslu vhodná kódování `8z` i `8t`, protože obě zahrnují kompletní českou a slovenskou abecedu.

`\characterdel`
`\accentdel`
Chceme-li odstranit už deklarovaný znak (viz též tzv. ⟨*výjimky*⟩ v následující sekci), můžeme použít příkazy `\characterdel` a `\accentdel`. Tyto příkazy mají stejné parametry jako `\characterdef` resp. `\accentdef` a odstraní definici příkazu `\`⟨*sekvence*⟩`:-`⟨*kódování*⟩ resp. `\`⟨*sekvence*⟩`:`⟨*znak*⟩`:-`⟨*kódování*⟩.

3.5. PLAIN: Soubory maker závislých na kódování a výjimky z kódování

Příkazy \characterdef a \accentdef popsané v předchozí sekci předefinovávají makra závislá na kódování (\v, \ae, atd.). V této sekci popíšeme koncepci, kam tato makra zapisovat, aby OFS při \loadingenc=1 udržovalo konzistentní definice těchto maker v závislosti na zvoleném kódování a na množině výjimek použité rodiny fontů.

\loadingenc

Deklarace maker pomocí \characterdef a \accentdef se zapisují do souborů s názvem ofs-⟨*kódování*⟩.tex (tzv. *kódovacích souborů*), protože příkaz \loadtextfam (volaný z makra \setfonts) tyto deklarace při nastaveném \loadingenc=1 odtud čte.

\modifydef

Každé kódování má svou základní sadu znaků a akcentovaných písmen. Tato sada je v kódovacím souboru zanesena pomocí příkazů \characterdef a \accentdef. Vzhledem k této základní sadě mohou jednotlivé rodiny fontů obsahovat nějaké znaky navíc nebo naopak nějaké znaky mohou chybět. Tyto výjimky se deklarují pomocí příkazu \modifydef takto:

\modifydef ⟨*kódování*⟩:⟨*identifikátor*⟩; {⟨*výjimky*⟩}

Přitom ⟨*výjimky*⟩ obsahují příkazy \characterdef, \accentdef, \characterdel a \accentdel. Příkazy *del musejí obsahovat v argumentu hodnotu znaku podle základní sady daného kódování. Pokud má být nějaký znak vzhledem k základní sadě předefinován, musejí příkazy *del a *def k tomuto znaku následovat těsně za sebou.

\modifyenc

V parametru příkazu \ofsdeclarefamily můžeme pomocí „odkazu na ⟨*výjimky*⟩" příkazem \modifyenc vyznačit, že daná rodina obsahuje vzhledem k základní sadě kódování odpovídající ⟨*výjimky*⟩. Příkaz má tvar:

\modifyenc ⟨*kódování*⟩:⟨*identifikátor*⟩;%

Takových příkazů může být pro každou rodinu fontů uvedeno více (a samozřejmě tyto příkazy mohou obsahovat různá ⟨*kódování*⟩). Odkazuje-li tento příkaz na ⟨*výjimky*⟩, které ještě nebyly deklarovány, neprovede se nic.

Příklad deklarace výjimek kódování 8z:csfonts najdete v souboru ofs-8z.tex a odkazy na ně pak najdete u rodin CM* v souboru ofsdef.tex.

\newmodifylist

Příkazy \modifyenc jsou spuštěny při každém \setfonts. Ve skutečnosti tyto příkazy pouze ukádají do tzv. *seznamu odkazů* (do makra \newmodifylist) své parametry. Při startu \setfonts se zakládá prázdný seznam odkazů. \modifyenc ukládá do tohoto seznamu svůj parametr jen tehdy, pokud ⟨*kódování*⟩ je rovno \fotenc nebo \extraenc. Vlastní práci (nastavení výjimek) provede až příkaz \runmodifylist, který je spuštěn na konci \setfonts a jehož činnost si nyní podrobně popíšeme.

\runmodifylist
\modifylist

Makro \runmodifylist porovná seznam odkazů na výjimky předchozí rodiny (\modifylist) se seznamem odkazů nově nastavené rodiny (\newmodifylist). Pokud jsou oba seznamy stejné nebo \modifylist má význam \relax, pak \runmodifylist ukončí činnost. Jinak se pustí do nastavení výjimek. Nejprve prohodí významy \characterdef↔\characterdel a \accentdef↔\accentdel a spustí \modifylist Jinými slovy vrátí stav maker vázaných na kódování do výchozího stavu. Při této činnosti je ignorováno mazání znaku, pokud těsně před tím byl znak deklarován (viz pravidlo o předefinování znaku na začátku této sekce). Pak příkaz \runmodifylist vrátí \characterdef a \accentdef do původního stavu a spustí \newmodifylist. Veškerá předefinování, která se při této činnosti provádějí, jsou jen lokální. Mechanismus dvou seznamů odkazů zaručuje, že například při

\setfonts [Rodina1/] ... \setfonts [Rodina2/] ...

budou správně nastaveny výjimky aktuálního kódování i pro Rodinu2, ačkoli Rodina1 má jinou množinu výjimek než Rodina2.

Kontrolní sekvence `\modifylist` má po inicializaci OFS význam prázdného makra. Programátor maker může použít `\let\modifylist=\relax` a tím potlačí veškeré nastavování výjimek.

Poznamenejme ještě, že deklarační příkazy `\modifydef` nejenže ukládají ⟨*výjimky*⟩ do paměti, ale projdou také jejich obsah a definují všechny zde deklarované sekvence jako odpovídající `\printcharacter` resp. `\printaccent`. Tím máme zaručeno, že každá kontrolní sekvence ze všech výjimek je definována (nedojde ke zprávě `undefined control sequence`) a navíc OFS má dokonalý přehled o tom, zda je v aktuální rodině tato kontrolní sekvence použitelná nebo nikoli. Tvůrce maker si pak může podle potřeby předefinovat makra `\print*warn`.

`\skipfirststep`

Příkaz `\modifydef` prochází ⟨*výjimky*⟩ tak, že na přechodnou dobu pozmění příkazy `\accentdef`, `\accentdel`, atd. a ⟨*výjimky*⟩ spustí. Pokud nechceme, aby se makra ve ⟨*výjimkách*⟩ spouštěla už v době činnosti `\modifydef`, uvedeme před tato makra sekvenci `\skipfirststep`. Ta se chová jako `\relax`, ale při spouštění ⟨*výjimek*⟩ příkazem `\modifydef` způsobí, že vše za touto sekvencí až do konce ⟨*výjimek*⟩ bude přeskočeno.

`\lccodes`
`\lccodesloop`

Identifikátory ⟨*kódování*⟩`:lccodes` a ⟨*kódování*⟩`:ienc` jsou rezervovány pro použití mimo OFS v dalších makrech. OFS například vůbec neřeší nastavení `\lccode`, `\uccode` znaků podle použitého kódování. Makro, které se o toto stará, může vhodně definovat příkazy `\lccodes` a `\lccodesloop` a spustí `\csname` ⟨*kódování*⟩`:lccodes\endcsname`. OFS uvedené příkazy vůbec nedefinuje. Deklarace ⟨*kódování*⟩`:lccodes` jsou uvedeny v souborech `ofs-8t.tex` a `ofs-8z.tex` ačkoli nejsou v OFS využity, protože to souvisí s kódováním textových fontů. Příklad použití ⟨*kódování*⟩`:lccodes` je v makru `lang.tex` a makro `inec.tex` používá ⟨*kódování*⟩`:ienc`. V dokumentaci k těmto makrům je řečeno více.

`\modifyread`

Deklarace nejčastěji používaných výjimek se zapisují přímo do kódovacích souborů. Deklarace méně obvyklých výjimek (vztahujících se jen na určité rodiny fontů) můžeme psát za `\endinput` deklaračních souborů. Mezi ⟨*příkazy*⟩ makra `\ofsdeclarefamily` pak použijeme příkaz

`\modifyread` ⟨*jméno souboru*⟩`;%`

`\modifytext`

Tento příkaz při kladném `\loadingenc` načte daný soubor od výskytu sekvence `\modifytext`. Tuto sekvenci je vhodné umístit za `\endinput`, aby `\modifyread` přečetl tu část souboru, která zatím nebyla čtena. Přiřazení je v době čtení globální a ignorují se konce řádků a prázdné řádky. Opakovaně se soubor nenačítá.

Díky tomuto příkazu můžeme soustředit deklarace rodin fontů a deklarace výjimek z kódování do společného souboru. Výjimky TeX přečte až v době, kdy to potřebuje. Šetří se tím paměť TeXu. Příklad použití je v souboru `slido.tex`.

`\knownchar`

OFS nabízí také testovací makro, zda kontrolní sekvence odpovídá znaku, který je ve fontu přístupný či nikoli:

```
\knownchar ⟨znak nebo akcent+znak⟩? \iftrue znak je přístupný
              \else   znak je nepřístupný nebo nedefinovaný \fi
% příklad:
\def\tryeuro{\knownchar \euro? \iftrue \euro \else Euro\fi}
```

3.6. Pomocná makra pro akcenty a znaky

\accentabove
\accentbelow

Pro pohodlné deklarování defaultních akcentů jsou v OFS připravena dvě makra: `\accentabove` a `\accentbelow`, která se používají takto:

```
\accentabove {⟨znak akcentu⟩}{⟨vertikální mezera⟩}{⟨základní znak⟩}
\accentbelow {⟨znak akcentu⟩}{⟨vertikální mezera⟩}{⟨základní znak⟩}
```

Příkaz `\accentabove` resp. `\accentbelow` přisadí ⟨*znak akcentu*⟩ nad resp. pod ⟨*základní znak*⟩, přičemž mezi těmito znaky bude ⟨*vertikální mezera*⟩. Oba znaky budou na společné vertikální ose a při skloněném písmu na společné skloněné ose. Šířka výsledného složeného znaku je odvozena od šířky základního znaku. Makra jsou implementována jako `\vbox` resp. `\vtop` společně s `\halign` a s výpočtem usazení na případně skloněné ose.

Znaky pro akcent nahoře jsou ve fontech vesměs kresleny ve výšce odpovídající základnímu znaku o výšce 1 ex. Proto usazení takového akcentu vyžaduje kompenzaci:

```
\accentabove {⟨znak akcentu⟩}{-1ex}{⟨základní znak⟩}
```

V tomto případě se akcent umístí naprosto stejně, jako při použití primitivu `\accent`. Na rozdíl od tohoto primitivu ale makra `\accentabove` a `\accentbelow` umožnují vzájemné skládání, čímž můžeme dosáhnout několikanásobného akcentu. Vyzkoušejte:

```
\it \accentabove {.}{.1ex}{\accentabove {,}{.1ex}{\v A}}
```

PLAIN: Bohužel, makra deklarovaná pomocí `\accentdef` nejsou uzpůsobena na více než dvojí akcent. Navíc dvojí akcent je možný jen v tom případě, že se vnitřní akcent se základním znakem realizuje jako jediný znak. Pokud potřebujeme vícenásobný akcent, můžeme vždy použít přímo v textu makro `\accentabove` nebo `\accentbelow`.

\ofshexbox
\ofshexboxdef

PLAIN: Od verze OFS Feb. 2004 je k dispozici makro `\ofshexbox`, které pracuje podobně jako plainové `\mathhexbox`, ale navíc dovede nastavit font podle aktuální varianty. Nejprve pomocí příkazu `\ofshexboxdef` nastavíme „rodinu“ čtyř metrik:

```
\ofshexboxdef ⟨rodina⟩{⟨metrika-rm⟩}{⟨metrika-bf⟩}{⟨metrika-it⟩}{⟨metrika-bi⟩}
\ofshexboxdef 2 {cmsy}{cmbsy10}{cmsy}{cmbsy10} % například takto
```

Pak příkaz `\ofshexbox` ⟨*rodina*⟩⟨*hexa-kód*⟩ vytiskne znak s ⟨*hexa-kódem*⟩ z jednoho ze čtyř deklarovaných fontů a v aktuální velikosti `\fosize` Volba fontu závisí na aktuální variantě. Pokud je varianta jiná než `\bf`, `\it`, `\bi`, použije se ⟨*metrika-rm*⟩. OFS implicitně deklaruje jen ⟨*rodinu*⟩ = 2, protože plain používá jen `\mathhexbox2`… Cílem tohoto makra bylo definovat pro CMfonty/CSfonty znaky `\S`, `\dag`, `\ddag`, `\P` způsobem, který je nezávislý na aktuálním nastavení matematických fontů, ale je závislý na aktuální velikosti a variantě. Viz soubor `ofs-8z.tex`.

Pomocí `\ofshexbox` můžeme snadno definovat například symbol `\euro` pro všechna kódování fontů, ve kterých je tento symbol nepřítupný:

```
\ofshexboxdef {TS1}{tcrm1000}{tcbx1000}{tcti1000}{tcbi1000}
\characterdef \euro * {\ofshexbox{TS1}BF}
```

3.7. PLAIN: Fonty v matematice podruhé

\setmath
\textfosize
\scriptfosize

Nastal čas podrobněji vysvětlit činnost příkazu `\setmath`. Ten převezme parametry a vypočítá z nich základní, indexovou a index-indexovou velikost fontů. Výsledky uloží do maker `\textfosize`, `\scriptfosize` a `\scriptscriptfosize`. V nich jsou

\scriptscriptfosize údaje o velikosti uloženy ve tvaru `at`⟨*dimen*⟩ nebo `scaled`⟨*číslo*⟩ podobně jako v makru
\mathfonts `\fosize`. Pak příkaz `\setmath` spustí makro `\mathfonts`. V něm se předpokládá zavedení a nastavení matematických fontů. Pokud je `\setmath` spuštěno poprvé nebo pokud
\mathchars došlo ke změně hodnoty `\fomenc`, tak `\setmath` ještě spustí makro `\mathchars`. Tam se obvykle nastavují kódy matematických znaků pomocí `\mathcode`, `\mathchardef` a příbuzných primitivů. Makra `\mathfonts` a `\mathchars` mají sice v OFS svou implicitní hodnotu, ale zkušený uživatel je může podle své potřeby měnit.

Pro pohodlné zavádění a nastavování matematických fontů je připraveno makro
\loadmathfam `\loadmathfam`, které se používá v definici makra `\mathfonts`. Makro `\loadmathfam` má tyto možnosti parametrů:

```
%%                                     % font je deklarován:
\loadmathfam ⟨rodina⟩[/⟨metrika⟩]    % metrikou
\loadmathfam ⟨rodina⟩[-⟨varianta⟩/] % variantou aktuální rodiny
\loadmathfam ⟨rodina⟩[⟨přepínač⟩/]  % přepínačem textového fontu
\loadmathfam ⟨rodina⟩[X⟨přepínač⟩/] % rozšiřující metrikou přepínače
```

Příklady:

```
\loadmathfam 0[tenrm/]% metrika podle přepínače \tenrm
\loadmathfam 5[-bi/]%   metrika z akuální textové rodiny, varianta bi
\newmathfam \symbfam
\loadmathfam \symbfam [/psyr]%    metrika psyr
\newmathfam \extitfam
\loadmathfam \extitfam [Xtenit/]% rozšiřující metrika k přepínači tenit
```

V příkladě jsou do rodiny 0 zavedeny fonty stejné, jako v aktuálním textovém fontu `\tenrm`. V rodině 5 jsou zavedeny fonty jako při `\setfonts [-bi/]`. Dále je deklarována nová rodina `\symbfam`, do které jsou zavedeny fonty z metriky `psyr`.

Existuje nepatrný rozdíl např. mezi použitím příkazu `\loadmathfam 5[tenbi/]` a `\loadmathfam 5[-bi/]`. V prvním případě OFS zjistí metriku k přepínači `\tenbi` a tuto metriku použije ve všech velikostech stejnou pouze modifikovanou klíčovým slovem `at`⟨*dimen*⟩. Ve druhém případě se může pro různé velikosti použít různá metrika, pokud je tato vlastnost pro danou variantu fontu deklarována (viz sekce 3.8).

Pro deklaraci nové rodiny se v příkladu použila alternativa k `\newfam` s názvem
\newmathfam `\newmathfam`, protože makro `\newfam` z plainu je definováno jako `\outer` a nelze je tedy použít uvnitř definice. Navíc makro `\newmathfam` pracuje lokálně a tudíž šetří více místa pro uživatelské rodiny než plainovské `\newfam`. V deklaracích základních matematických kódování se nové matematické rodiny definují přímo pomocí `\chardef`
\lastfam a je tam pomocí příkazu `\lastfam=`⟨*číslo*⟩ nastavena maximální použitá hodonta. To zaručí, že uživatel může později použít `\newmathfam` a začne alokovat další rodiny s čísly těsně většími než `\lastfam`.

Popíšeme nyní princip činnosti makra `\loadmathfam`. Toto makro zjistí metriku, která odpovídá zadanému parametru. Pak provede třikrát primitiv `\font` zhruba takto:

```
\font \⟨název⟩-Mt = ⟨metrika⟩ \textfosize
\font \⟨název⟩-Ms = ⟨metrika⟩ \scriptfosize
\font \⟨název⟩-Mss = ⟨metrika⟩ \scriptscriptfosize
\textfont ⟨rodina⟩ = \⟨název⟩-Mt
\scriptfont ⟨rodina⟩ = \⟨název⟩-Ms
\scriptscriptfont ⟨rodina⟩ = \⟨název⟩-Mss
```

Přitom ⟨*název*⟩ je text parametru `\loadmathfam`, který deklaruje metriku (přepínač, varianta nebo metrika).

`\noindexsize` Fonty matematické rodiny 3 jsou v plainTeXu obvykle zavedeny bez zmenšování pro indexy a indexy indexů. Pokud požadujeme při zavádění matematické rodiny nezmenšovat fonty pro indexy, uvedeme těsně před `\loadmathfam` prefix `\noindexsize`. Makro `\loadmathfam` pak zavede font jen ve velikosti `\textfosize` a použije ho i pro indexovou a indexindexovou velikost. Příklad:

```
\noindexsize\loadmathfam 3[tenex/]% Standard extra symbols from CM
```

OFS definuje v souboru `ofsdef.tex` pro zavádění matematických fontů čtyři různá makra. Ta se použijí podle toho, zda je `\fomenc` nastaveno na `CM` nebo `PS` a `\mathversion` na `normal` nebo `bold`. Dále OFS definuje dvě makra s deklaracemi matematických kódů, která se použijí podle aktuálně nastavené hodnoty `\fomenc`.

`\loadCMnormalmath`
`\loadCMboldmath`
`\loadPSnormalmath`
`\loadPSboldmath`
`\setCMmathchars`
`\setPSmathchars`

- `\loadCMnormalmath` — zavede CM fonty ve verzi „normal“.
- `\loadCMboldmath` — zavede CM fonty ve verzi „bold“.
- `\loadPSnormalmath` — zavede PostScriptové fonty ve verzi „normal“.
- `\loadPSboldmath` — zavede PostScriptové fonty ve verzi „bold“.
- `\setCMmathchars` — ponechá matematické kódy z plainu.
- `\setPSmathchars` — nastaví matematické kódy tak, aby se nahradily některé znaky z fontu Symbol.

Dále je v OFS implicitně nastaveno:

`\defaultmathfonts`
`\defaultmathchars`

```
\ifx \fomenc\undefined \def\fomenc{PS}\fi
\def\mathversion{normal}
\def\defaultmathfonts{\csname load\fomenc\mathversion math\endcsname}
\def\defaultmathchars{\csname set\fomenc mathchars\endcsname}
\def\mathfonts{\defaultmathfonts}
\def\mathchars{\defaultmathchars}
```

Je možné přidávat k implicitně nastaveným matematickým fontům další matematické rodiny (v terminologii NFSS matematické abecedy) pozměněním maker `\mathfonts` a případně `\mathchars`. Níže je příklad přidání fraktury z Eulerových fontů od AMS:

```
\input amsfn % tam je deklarována metrika eufm a eufb
\addcmd\mathfonts{\def\tmpa{bold}%
   \ifx\mathversion\tmpa \def\tmpa{b}\else\def\tmpa{b}\fi
   \newmathfam\frakfam \loadmathfam\frakfam [/euf\tmpa]}
\def\frak#1{{\fam\frakfam#1}}
```

Další příklady deklarací matematických rodin jsou uvedeny v souborech `amsfn.tex`, `txfn.tex` a `mtfn.tex`. Soubory definují výchozí skupiny matematických fontů pro kódování `AMS`, `TX`, `PX`, `MT`. Na konci těchto souborů jsou komentáře včetně ukázek, jaké doplňující rodiny fontů je možno pomocí `\addcmd` zavést.

Protože chci umožnit načtení všech fontových deklarací OFS v iniTeXu (viz projekt OkTeX), ale přitom šetřit pokud možno pamětí TeXu, rozhodl jsem se nenačítat rozsáhlé definice obsahující deklarace kódování matematických fontů pomocí `\mathchardef` atd. okamžitě, ale až v době, kdy to uživatel v dokumentu skutečně potřebuje. Proto jsem od verze OFS Apr. 2004 předefinoval například makro `\setPSmathchars` takto:

```
\def\setPSmathchars{\mathencread ofs-ps;}
```

\mathencread — Příkaz `\mathencread` ⟨*soubor*⟩`;` načte kódovací příkazy ze souboru ⟨*soubor.tex*⟩. Např. v souboru `ofs-ps.tex` jsou kódovací příkazy pro kódování `PS`, v `ofs-ams.tex` jsou kódovací příkazy pro kódování `AMS` atd.

\mathencdef — V souborech `ofs-ps.tex`, `ofs-tx.tex` atd. jsou kódovací příkazy „baleny" do skupin a definovány příkazem `\mathencdef`. Tento příkaz pracuje implicitně jako „definuj, spusť a zapomeň". Tento model šetří paměť, ale při častých změnách matematického kódování budou kódovací soubory čteny opakovaně znovu. To většinou nevadí, protože většinou bývá pro celý dokument nastaveno jediné matematické kódování. Pokud ale tento model někomu nevyhovuje, může si jako cvičení zkusit předefinovat `\mathencread` a `\mathencdef` tak, aby se soubory četly jen jednou a při opakovaném spuštění `\mathchars` se kódovací příkazy četly ze zapamatovaných maker.

Příkaz `\mathencread` ⟨*soubor*⟩`;` založí skupinu, ve které pro jistotu nastaví kategorie všech znaků podle plainTeXu (s výjimkou `` \catcode`@=11 ``) a přečte ⟨*soubor*⟩`.tex`. Při čtení je `\globaldefs=1` a ignorují se konce řádků a prázdné řádky. Rovněž je ignorováno makro `\protectreading`, takže soubor může být čten opakovaně. Příkazy `\mathencdef` spustí a zapomenou nadefinovaná makra až po uzavření skupiny, takže tato makra spouští příkazy `\mathchardef` atd. už jen s lokálními přiřazeními.

\mathcharsback — Vysvětlíme, jak je zajištěno obnovování výchozích hodnot při přepínání mezi matematickými kódováními. Příkaz `\setmath` spouští těsně před `\mathchars` makro `\mathcharsback`, aby uvedl matematické kódování do výchozího stavu podle plainTeXu. Toto makro má implicitně význam `\relax`, protože implicitně je matematické kódování podle plainTeXu nastaveno. Mění-li ale příkaz `\set`⟨*fomenc*⟩`mathchars` hodnoty nastavené v plainTeXu, měl by také definovat makro `\mathcharsback`, do kterého uloží postup, jak vrátit nastavení do původního stavu. V definici makra `\mathcharsback` můžeme mimo jiné použít příkaz `\mathencread ofs-cm;`, protože v souboru `ofs-cm.tex` jsou uloženy deklarace metamatického kódování podle plainTeXu

Pokud si budeme deklarovat vlastní matematická kódování (různá od připravených `PS`, `CM`, `AMS` atd.), pak je potřeba myslet na následující zásadu: ve všech verzích matematických fontů (normal/bold/atd.) by měla být čísla matematických rodin alokována stejně a ukončena stejným `\lastfam`, protože makro `\mathchars` se po přepnutí do jiné verze nespouští znovu. Zvláště nesmíme měnit čísla těch rodin, na které se odvoláváme v makru `\set`⟨*kódování*⟩`mathchars`.

\hex — Nyní popíšeme další makra ulehčující deklaraci matematického kódování. Makro `\hex` konvertuje číslo na jednociferné hexadecimální číslo. Využití tohoto makra najdeme v souborech `ofs-ps.tex`, `ofs-tx.tex` a dalších.

\safemathchardef — Pro definování sekvencí, které budou pracovat v textovém i matematickém módu současně, slouží makro `\safemathchardef`. Toto makro má stejný způsob použití, jako primitivní `\mathchardef`:

```
\safemathchardef \⟨sekvence⟩ ⟨číslo⟩
```

Pokud není `\`⟨*sekvence*⟩ v okamžiku použití `\safemathchardef` definována, příkaz pracuje zcela stejně jako `\mathchardef`. Pokud ale je `\`⟨*sekvence*⟩ už definována, uloží příkaz `\safemathchardef` její význam do `\T`⟨*sekvence*⟩, dále provede `\mathchardef \M`⟨*sekvence*⟩ `=` ⟨*číslo*⟩ a konečně `\`⟨*sekvenci*⟩ předefinuje následujícím způsobem:

```
\def \⟨sekvence⟩ {\ifmmode \expandafter \M⟨sekvence⟩
                  \else \expandafter \T⟨sekvence⟩ \fi}
```

Od této chvíle \⟨*sekvence*⟩ pracuje v textovém i matematickém módu. Pro případ opakovaného použití `\safemathchardef` na stejnou \⟨*sekvenci*⟩ makro kontroluje, zda už není definována `\M`⟨*sekvence*⟩. Pokud ano, pak `\safemathchardef` neprovede nic.

Je potřeba, aby byly soubory s příkazy `\characterdef` načteny před prvním použitím `\setmath`. Pak lze v rámci `\setmath` (přesněji v makru `\mathchars`) použít `\safemathchardef` na předefinování některých sekvencí, které byly dříve deklarovány pomocí `\characterdef`. Tyto sekvence pak budou pracovat v textovém i matematickém módu.

`\safemathaccentdef` `\mathaccentdef` Naprosto stejně, jako `\safemathchardef` pracuje příkaz `\safemathaccentdef`. Pouze místo primitivu `\mathchardef` použije makro `\mathaccentdef`, které přečte za \⟨*sekvencí*⟩ ⟨*číslo*⟩ separované mezerou a provede:

```
\def \⟨sekvence⟩ {\mathaccent⟨číslo⟩ }
```

Pokud nechceme deklarovat novou matematickou rodinu pro několik málo znaků (počet matematických rodin je totiž v TEXu omezen na 16), můžeme použít makro `\pickmathfont` `\pickmathfont`, které má tyto parametry:

```
\pickmathfont {⟨metrika⟩}{⟨text⟩}
% příklad použití:
\mathbin {\pickmathfont {psybo}{\char"C4}}
```

Makro `\pickmathfont` použije primitiv `\font` na specifikovanou ⟨*metriku*⟩ a v tomto fontu zapíše ⟨*text*⟩, který v matematickém seznamu tvoří atom typu Ord. Důležité je, že je použit font v odpovídající velikosti (základní, indexová, indexindexová), takže to vypadá, jako by byl použita nová matematická rodina fontů. Přitom to není pravda, protože `\pickmathfont` je implementován pomocí primitivu `\mathchoice`. Použití příkazu `\pickmathfont` najdeme v souboru `ofsdef.tex` v definici makra `\setPSmathchars`.

3.8. Registrování metrik pro různé velikosti

LATEX: Registrování různých metrik pro různé velikosti se dělá ve `fd` souborech a je to dokonale vyřešeno v NFSS.

PLAIN (až do konce sekce): Existují speciální rodiny fontů (například Computer Modern), kde se pro různé velikosti fontů používají různé metriky. I tato vlastnost je implementována v OFS. Dosud jsme zatajili, že v parametrech příkazů `\loadtextfam`, `\loadmathfam` a `\pickmathfont` je možné místo názvů metrik uvést smyšlené ⟨*jméno*⟩, `\registertfm` které je registrováno příkazem `\registertfm`. V takovém případě se skutečná metrika pro zavedení primitivem `\font` vypočítá z požadované velikosti fontu a z údajů, které byly zadány pomocí skupiny příkazů `\registertfm`. Upřesníme nyní parametry příkazu `\registertfm`:

```
\registertfm ⟨jméno⟩ ⟨mezera⟩ ⟨od⟩-⟨do⟩ ⟨mezera⟩ ⟨skutečná metrika⟩ ⟨mezera⟩
```

Parametry ⟨*od*⟩ a ⟨*do*⟩ musejí obsahovat jednotku a vymezují uzavřený interval, pro který platí: jestliže je požadovaná velikost fontu ve tvaru `at`⟨*dimen*⟩ a současně ⟨*dimen*⟩ leží v intervalu ⟨*od*⟩-⟨*do*⟩, pak se místo ⟨*jména*⟩ použije ⟨*skutečná metrika*⟩ `at`⟨*dimen*⟩. Příkaz `\registertfm` je potřeba pro vymezení různých intervalů použít opakovaně na stejné ⟨*jméno*⟩. Ačkoli jsou intervaly ⟨*od*⟩-⟨*do*⟩ uzavřené, později deklarovaný interval má přednost před dříve deklarovaným, takže nakonec je kladná reálná osa rozdělena na polouzavřené intervaly. Příklad použití je v souboru `ofsdef.tex`.

Jsou-li oba parametry ⟨*od*⟩ a ⟨*do*⟩ prázdné, je příkazem `\registertfm` deklarována metrika, která se použije při specifikaci velikosti pomocí `scaled`⟨*číslo*⟩ nebo v případě, kdy ⟨*dimen*⟩ neleží v žádném specifikovaném intervalu. Hvězdička místo parametru ⟨*do*⟩ značí „nekonečnou velikost“.

Příkaz `\registertfm` ⟨*jméno*⟩ ⟨*mezera*⟩ `-` ⟨*mezera*⟩ `-` ⟨*mezera*⟩ vymaže z paměti předchozí registrace pro dané ⟨*jméno*⟩ a navíc vyznačí ⟨*jméno*⟩ jako nepřístupný font. OFS se pak chová stejně, jakoby odpovídající varianta nebyla vůbec deklarována. Dává nám to možnost vyznačit neexistující varianty deklarované rodiny jen pro některá kódování (viz například `cmssbxti` v souboru `ofsdef.tex`).

Parametry ⟨*jméno*⟩ a ⟨*skutečná metrika*⟩ se expandují v okamžiku činnosti příkazu `\registertfm`. Není proto obvyklé v nich používat makro `\fotenc`. Nicméně toto makro můžeme v uvedených parametrech použít, pokud zahrneme potřebnou skupinu příkazů `\registertfm` do vlastního makra a toto makro spustíme opakovaně pro různé hodnoty `\fotenc`.

`\registerECfont` `\registerECTTfont` Makro `\registerECfont` je zkratkou za opakované použití příkazu `\registertfm` na všechny velikosti EC fontů od `0500` až po `3583`. Stejně makro `\registerECTTfont` je zkratkou pro velikosti EC fontů od `0800` do `3583` používané pro strojopis. Definice a použití těchto maker najdeme v souboru `ofsdef.tex`. Makra můžeme použít i na jiné fonty, které mají stejně odstupňované velikosti jako EC fonty (např. LH fonty s azbukou nebo fonty odvozené z CM-super).

3.9. Omezení použití rodiny fontů pro určitá kódování

LATEX: Použití rodiny v daném kódování je závislé na existenci `fd` souboru, tj. vše je v režii NFSS.

`\registerenc` PLAIN (až do konce sekce): OFS od verze Feb. 2004 zavádí příkaz `\registerenc`, kterým můžeme omezit použití deklarované rodiny fontů jen pro vymezená kódování. Pokud pak uživatel napíše `\setfonts` a vyžaduje rodinu v neregistrovaném kódování, OFS vypíše varování na obrazovku a vůbec do požadované rodiny nepřepne. Tím se vyhne pokusu zavést pravděpodobně neexistující metriky fontů. Příkaz `\registerenc` má dva parametry:

```
\registerenc ⟨JménoRodiny⟩: ⟨kódování⟩⟨mezera⟩
\registerenc Times: 8t   % například
\registerenc Times: 8z   % nyní má Times registrována dvě kódování
```

Pokud nemá rodina registrováno žádné kódování, pak OFS předpokládá, že je použitelná v jakémkoli kódování (například fonty značek).

V deklaračních souborech tedy můžeme pomocí `\registerenc` omezit použití rodiny jen na vymezená kódování. Uživatel ale může příkazem `\registerenc` přidat k rodině další povolené kódování. Nebo také může naznačit, že rodina má povoleno libovolné kódování příkazem `\registerenc` ⟨*JménoRodiny*⟩`: *` .

Abychom v deklaračních souborech nemuseli opisovat ⟨*JménoRodiny*⟩, může být tento parametr prázdný. V takovém případě `\registerenc` použije rodinu z posledního příkazu `\ofsdeclarefamily`.

`\registeredfam` Ve svých makrech se můžete zeptat, zda je rodina pro aktuální hodnotu `\fotenc` registrovaná:

```
\registeredfam ⟨JménoRodiny⟩? \iftrue
      Rodina má \fotenc registrováno nebo má povoleno jakékoli kódování
      nebo vůbec není deklarována.
\else Rodina má registována kódování, ale \fotenc mezi nimi není.
\fi
```

\setfontsOK Pokud spustíme \setfonts[⟨*JménoRodiny*⟩/] pro nedeklarovanou rodinu nebo pro rodinu s nedovoleným kódováním, pak OFS napíše varování na terminál a vrátí \setfontsOK jako \undefined. Po úspěšném nastavení fontu má sekvence \setfontsOK hodnotu \relax.

4. Licence

Balíček OFS může používat kdokoli bez licenčních poplatků. Kdokoli jej rovněž může distribuovat, pokud nezmění obsah souboru readme.ofs ofs.tex, ofsdef.tex, ofs.sty, ofs-8z.tex, ofs-8t.tex, a35.tex, a35.sty, ofsdoc.tex, ofsdoce-e.tex, ofsmtdef.tex a všechny tyto soubory budou v distribuci přítomny. Právo na změnu těchto souborů a tím změnu verzí zůstává výhradně autorovi. Pokud potřebujete změnit obsah některého z uvedených souborů, nazvěte jej jinak. Balíček je poskytován s přáním, aby byl užitečný, ale BEZ JAKÉKOLI ZÁRUKY.

V Praze dne 16. 8. 2001 autor: Petr Olšák

5. Historie

16. 8. 2001 — uvedena první verze
24. 10. 2002 — provedeny drobné úpravy respekutjící změny ve verzi OFS Oct. 2002. V této verzi byl přidán příkaz \addcmd a byly umožněny nepovinné parametry (⟨*Varianta*⟩) v příkazu \loadtextfam.
10. 2. 2004 — Všechny úpravy se týkají jen OFS pro plain:
+ Vylepšena manipulace s deklaracemi maker závislých na kódování, deklarace výjimek a registrování kódování pro zvolenou rodinu. Viz sekce 3.5 a sekce 3.9.
+ Přidána možnost předefinování rodiny (např. dodatečným načtením souboru deklaračního souboru, který modifikuje vlastnosti implicitních rodin).
+ Deklarace CMRoman, CMSans a CMTypewriter pro kódování 8t použitím EC fontů.
+ Přidána podpora rozšiřujícího kódování 8c.
+ Upravena deklarace matematického kódování PS, aby \int, \sum a \prod pracovaly v display módu s většími znaky.
+ Definováno makro \ofshexbox a \ofshexboxdef.
+ Nově zveřejněno interaktivní makro ofstest.tex.
+ Zaveden adresář examples/, kde průběžně budu dávat příklady použití OFS.
12. 3. 2004 — OFS pro plain:
+ Kódovací soubory jsou čteny přímo z \loadtextfam. Tím je umožněno dávat do kódovacích souborů mapování metrik příkazem \registerenc (využiju pro LANG).
+ \characterdef respektuje definované makro a nepředefinuje jej.
+ Reimplementováno \showfonts. Nyní šetří výrazně paměť i čas na rozsáhlých seznamech fontů. Pozor: pokud jste měli makra využívající interní makro \listfamilies, pak toto přestane fungovat. Např. nefunguje od této verze zastaralé ofscatal.tex. Místo něj je možno použít ofstest.tex
+ Modifikováno ofstest.tex, aby pracovalo s novou implementací seznamu rodin \ofslistfamilies.
2. 4. 2004 — přidáno \plaincatcodes před čtením souborů ofs-⟨*kódování*⟩.tex.
+ Zavedeno \safelet
+ Mezera za ⟨*znakem*⟩ v \accentdef je nepovinná.

+ Přidána možnost `\loadmathfam` ⟨*rodina*⟩`[-`⟨*varianta*⟩`/]` a rozšířena a přepracována deklarace pro matematická kódování `CM`, `PS`, `AMS`, `TX`, `PX`, `MT`.

6. Rejstřík všech maker definovaných v OFS

Rejstřík obsahuje odkazy pouze na místa v textu, kde je k makru vysvětlující komentář. V tom místě je též uveden pro snadnější orientaci název makra v levém okraji. V rejstříku nejsou odkazy na všechny zmínky o makru v textu. Rejstřík je správně vygenerován hned po prvním průchodu zpracování tohoto dokumentu příkazem `csplain ofsdoc`.

Některá makra definovaná v `ofs.tex` jsou interní nebo pomocná a není o nich zmínka v textu. Pak odkaz na stranu záměrně chybí a je připojena jen krátká poznámka o takovém makru. Rovněž jsou připojeny poznámky o tom, zda je makro použito ve verzi pro PLAIN nebo LATEX.

`\sgvariant` (PLAIN) interní, pro uložení informace o variantě
`\showfonts` (PLAIN+LATEX) 3
`\singlefont` (PLAIN)
`\singlefontname` (PLAIN) interní, odstraní `at`⟨*dimen*⟩ z názvu metriky
`\skipfirststep` (PLAIN) 19
`\slantcorrection` (PLAIN) interní, pro potřeby `\accentabove` a `\accentbelow`
`\storeofsvariant` (PLAIN) interní, separace nepovinného parametru u `\loadtextfam`
`\switchdeftodel` (PLAIN) interní, prohodí `\accent/characterdef` s `\*del`
`\tenbi` (PLAIN) 13
`\tenbf` (PLAIN) 13
`\tenit` (PLAIN) 13
`\tenrm` (PLAIN) 13
`\testOFSoptions` (PLAIN) interní, pro potřeby makra `\endOFSmacro`
`\testtfmsize` (PLAIN) interní, pro potřeby `\registertfm`
`\testtfmsizeat` (PLAIN) interní, přechodná hodnota `\testtfmsize`
`\testtfmsizescaled` (PLAIN) interní, přechodná hodnota `\testtfmsize`
`\textfosize` (PLAIN) 20
`\tryloadenc` (PLAIN) interní, načte kódovací soubory
`\tmpa` (PLAIN+LATEX) přechodné
`\tmpb` (PLAIN+LATEX) přechodné
`\tmpc` (PLAIN) přechodné
`\warnmissingfont` (PLAIN) interní, výpis o chybějící variantě
`\warnM`, `\warnF`, `\warnT`, `\warnI` (PLAIN) interní, výpis o chybějící standardní variantě
`\warnunregistered` (PLAIN) interní, výpis o nedovoleném kódování zvolené rodiny

7. Reference

[TBN] Petr Olšák *TEXbook naruby*, Konvoj 1. vyd. 1997 (ISBN 80-7302-007-6), 2. vyd. 2001 (ISBN 80-85615-64-9), Brno, 466 stran. PDF verze knihy je volně k dispozici na `http://math.feld.cvut.cz/olsak/tbn.html`.