Zpravodaj Československého sdružení uživatelů TeXu

Petr Olšák

CSplain z roku 2012

Zpravodaj Československého sdružení uživatelů TeXu, Vol. 22 (2012), No. 1, 42–58

Persistent URL: http://dml.cz/dmlcz/150201

$\mathcal{CS}$plain z roku 2012

Petr Olšák

$\mathcal{CS}$plain existuje od roku 1994 a je to *jemné* rozšíření plainTEXu tak, aby v něm bylo možno pracovat v češtině a slovenštině. Toto platilo až do října 2012, kdy byla provedena významná revize a doplnění $\mathcal{CS}$plainu. Výchozí změna vyplynula z rozhodnutí implicitně nastavit vstupní kódování $\mathcal{CS}$plainu na UTF8. Kromě toho získal $\mathcal{CS}$plain mnoho dalších nových vlastností: zavádění všech dostupných vzorů dělení všemožných jazyků, schopnost spolupráce s 16bitovými TEXovými mašinami (LuaTEX, XeTEX), efektivní práce s fonty včetně matematických, snadné přepínání interního kódování včetně Unicode, uživatelsky přívětivá sada maker OPmac. V implicitní konfiguraci zůstává $\mathcal{CS}$plain stále jemným rozšířením plainTEXu zpětně kompatibilní s předchozími verzemi. Nové možnosti jsou snadno dostupné pomocí `\input` a při jejich použití už zdaleka není možné hovořit jen o rozšíření *jemném*. Naopak, je to silný konkurent všech možných poněkud přerostlých makro-nadstaveb TEXu, přitom vítězí v jednoduchosti, účelnosti řešení, přímočarosti a snadnosti použití. Nový $\mathcal{CS}$plain je přístupný na `http://petr.olsak.net/csplain.html`.

1. Úvod

V říjnu roku 2012 proběhla na diskusním listu `cstex` debata o konfiguraci vstupního kódování $\mathcal{CS}$plainu, ze které vyplynulo, že už mnoho let je $\mathcal{CS}$plain v distribucích pro MS Windows implicitně konfigurován špatně: s kódováním ISO-8859-2, které je v tomto operačním nesystému cizí. Zhrozil jsem se. Zjistil jsem posléze, že například studenti MFF, kteří pořádají korespondenční seminář, kvůli této chybě používají jakési své makro, které napravuje tuto chybu pomocí aktivních znaků. Zhrozil jsem se podruhé.

V diskusi jsem se snažil vysvětlit naše dávné rozhodnutí, že vstupní kódování $\mathcal{CS}$plainu musí být nastaveno v závislosti na použitém systému. Podstatné je, že když v editoru ve zdrojovém textu správně vidím české a slovenské znaky, $\mathcal{CS}$plain je správně vytiskne. Když je vidím blbě, nesmím se divit, že i z $\mathcal{CS}$plainu vylezou porouchané. Bohužel tato naše idea byla platná před deseti lety. Dnes jsou textové editory vybaveny inteligencí, která se snaží o autodetekci kódování a pokusí se cokoli zobrazit správně. V takovém prostředí výše uvedené pravidlo pozbývá smyslu. Na druhé straně tyto moderní editory zvládají kódování UTF8, takže jsme se rozhodli, že toto bude implicitně nastavené vstupní kódování $\mathcal{CS}$plainu pro všechny systémy. Ušetříme si tím starostí s různým nastavováním pro různé systémy. Samozřejmě je možné vygenerovat $\mathcal{CS}$plain postaru v nějakém zvoleném

 doi: 10.5300/2012-1/42

8bitovém vstupním kódování. Ale po instalaci TEXové distribuce bez dodatečného vrtání se v konfiguraci bude CSplain číst vstup v UTF8.

Konverze mezi UTF8 vstupními kódy a interním kódováním TEXu musí být přímočará na úrovni input procesoru, jako tomu bylo dosud. Proto je pro UTF8 kódovaný csplain v 8bitových TEXových mašinách (TEX, pdfTEX) použito rozšíření encTEX. Zda je toto rozšíření do TEXové mašiny zabudováno, poznáte pomocí přepínače `-enc`, kterému encTEXem poznamenané mašiny rozumějí. Není-li toto rozšíření k dispozici, není možné pro 8bitovou mašinu zprovoznit UTF8 kódovaný CSplain. Všechny obvyklé TEXové distribuce mají v 8bitových mašinách encTEX zabudován.

V době, kdy píšu tento článek (začátek prosince 2012) udržuji čilou komunikaci s Karl Berrym o zařazení nového CSplainu do TEXlive. Karl si už v listopadu sám od sebe všiml, že se něco děje, a oslovil mě. Je to tím, že mé FTP adresáře se automaticky zrcadlí do CTANu. Dá se tedy očekávat, že v nejbližší době bude nový CSplain součástí TEXlive. Až bude ukončeno jeho zařazení do TEXlive, oslovím Christiana Schenka a projednám s ním zařazení do MikTEXu.

2. Generování formátu

Formát CSplainu s UTF8 vstupem vygenerujeme následujícím způsobem:

```
pdftex -ini -enc "\let\enc=u \input csplain.ini"
pdftex -jobname pdfcsplain -ini -enc "\let\enc=u \input csplain.ini"
```

První řádek vygeneruje `csplain.fmt` s výstupem do DVI (místo mašiny pdfTEX lze použít i TEX). Druhý řádek vygeneruje `pdfcsplain.fmt` s výstupem do PDF. Makra při inicializaci CSplainu poznají podle názvu generovaného formátu (`csplain/pdfcsplain`), zda mají nastavit výstup do DVI nebo PDF. Přepínačem `-enc` je probuzen k životu encTEX a pokynem `\let\enc=u` je řečeno, že má encTEX nastavit vstupní kódování UTF8. Formát bude mít zavedeny anglické vzory dělení stejné jako v plainTEXu a dále české a slovenské vzory dělení, obě připraveny ve dvou kódováních: podle CSfontů a v T1 kódování (alias Cork). O možnosti zavést vzory dělení slov dalších jazyků pojednává sekce 10.

V konfiguračních souborech TEXových distribucí pro generování formátu se obvykle používá soubor `csplain-utf8.ini`, který obsahuje uvedený pokyn `\let\enc=u` a zavolá `csplain.ini`. Také je tam obvykle přidán přepínač `-etex`, který probudí k životu rozšíření eTEX. Makra CSplainu nikdy toto rozšíření nepoužívají, takže to není nutné rozšíření. Ovšem, pokud uživatel má zájem to použít, má i možnost. Takže v konfiguračních souborech pro generování formátu najdete obvykle analogii těchto příkazů:

```
pdftex -jobname csplain -ini -etex -enc csplain-utf8.ini
pdftex -jobname pdfcsplain -ini -etex -enc csplain-utf8.ini
```

Můžete samozřejmě také vygenerovat formát $\mathcal{CS}$plainu postaru s 8bitovým vstupem:

```
pdftex -ini -enc csplain.ini
pdftex -ini -enc "\let\enc=w \input csplain.ini"
pdftex -ini -enc "\let\enc=p \input csplain.ini"
```

Všechny tyto řádky generují `csplain.fmt` s rozšířením encTEX. První řádek vytvoří formát se vstupním kódováním ISO-8859-2, druhý s kódováním CP1250 (Windows) a třetí s kódováním CP852. Všechny varianty budou vystupovat do DVI. Chcete-li vytvářet PDF, přidejte `-jobname pdfcsplain`.

Při volbě 8bitového vstupu není nutné použít encTEX. Formát můžete generovat třeba takto:

```
pdftex -ini -translate-file=cp227.tcx csplain.ini
pdftex -ini -translate-file=cp1250cs.tcx csplain.ini
pdftex -ini -translate-file=cp852-cs.tcx csplain.ini
```

Jako prve: první řádek vytvoří formát se vstupním kódováním ISO-8859-2, druhý s kódováním CP1250 (Windows) a třetí s kódováním CP852. Jednotlivé možnosti nastavení (encTEX nebo TCX tabulka) nelze míchat dohromady.

Formát $\mathcal{CS}$plainu je možné vygenerovat i pro použití v 16bitových TEXových mašinách (LuaTEX a XETEX) pomocí následujících příkazů:

```
xetex -jobname pdfcsplain -etex -ini csplain.ini
luatex -ini csplain.ini
luatex -jobname pdfcsplain -ini csplain.ini
```

Vidíme, že na rozdíl od 8bitových mašin není nutné (a ani to není možné) použít rozšíření encTEX, protože UTF8 vstup je zde implicitní. Formát v tomto případě zavede vzory dělení češtiny a slovenštiny nejen podle kódování $\mathcal{CS}$fontů a Corku, ale také v Unicode. Program XETEX vystupuje pouze do PDF, takže můžete sice vygenerovat i `csplain.fmt` (bez předpony `pdf`), ale i ten bude vystupovat do PDF. Program LuaTEX se chová shodně jako pdfTEX a má tedy možnost přepnout mezi DVI a PDF výstupem. Implicitně nastavený výstup je tedy rozlišen podle názvu (`csplain/pdfcsplain`). Protože 16bitový výstup do DVI není dále snadno zpracovatelný, preferujeme výstup do PDF.

3. $\mathcal{CS}$plain – opakování

Nejprve stručně shrnu vlastnosti, které $\mathcal{CS}$plain odjakživa měl. V dalších sekcích jsou pak popsány novinky.

Formát csplain je inicializován tak, aby se implicitně choval jako plainTEX. To znamená, že je nastaveno anglické dělení slov a sekvence typu `\v`, `\'` expandují

na primitiv `\accent`. Rovněž je nastaveno `\nonfrenchspacing`. Rozdíl mezi plainTEXem a $\mathcal{CS}$plainem je jen ve velikosti implicitního rozměru zrcadla sazby, které je v $\mathcal{CS}$plainu nastaveno pro palcové okraje při formátu A4, zatímco v plainTEXu je nastaveno pro palcové okraje formátu Letter.

Volbu vzorů dělení slov a nastavení expanze sekvencí `\v`, `\'`, `\^`, `\‘`, `\"`, `\r` na přirozené znaky lze provést následující příkazy:

```
\chyph      % inicializuje české dělení slov a \frenschpacing
\shyph      % inicializuje slovenské dělení slov a \frenschspacing
\csaccents  % způsobí jiné chování \', \v, \^, \‘, \" a \r, které
            % od této chvíle expandují přímo na znaky podle CSfontů
```

Doporučení: První řádek dokumentu by měl být například:

```
\chyph  % use format csplain
```

Když bude uživatel takový dokument zpracovávat jiným formátem, který nemá definován příkaz `\chyph`, výše uvedený řádek se objeví v chybovém hlášení včetně komentáře, takže uživatel vidí, čím má dokument zpracovat.

Návrat k původnímu nastavení:

```
\ehyph      % americké dělení a \nonfrenchspacing
\cmaccents  % \', \v a spol.. expandují na primitiv \accent
```

Další řídicí sekvence jsou jen zkratkami k některým znakům v $\mathcal{CS}$fontech:

```
\clqq       % levá dvojitá česká uvozovka
\crqq       % pravá dvojitá česká uvozovka
\flqq       % levá dvojitá francouzská uvozovka
\frqq       % pravá dvojitá francouzská uvozovka
\promile    % znak pro promile
\uv         % text uvozený českými uvozovkami: \uv{text}
\ogonek a   % polské „a s ocáskem" (sestaveno z komponent)
```

4. UTF8 vstup při použití encTEXu

UTF8 kódovaný $\mathcal{CS}$plain s encTEXem napíše při každém spuštění do logu a na terminál toto hlášení:

```
The format: csplain <Nov. 2012>.
The cs-fonts are preloaded and A4 size implicitly defined.
The utf8->iso8859-2 re-encoding of Czech+Slovak alphabet
    activated by encTeX
```

Na vstupu se může vyskytnout desetitisíce různých UTF8 kódů. Nečekejte ovšem, že jsou všechny správně $\mathcal{CS}$plainem zpracovány bez starosti navíc. Vygenerovaný formát zaručí správné zpracování ASCII znaků. Dále je zaručeno správné zpracování znaků české a slovenské abecedy, tedy znaků:

Á á Ä ä Č č Ď ď É é Ě ě Í í Ĺ Í Ľ ľ Ň ň Ó ó Ö ö Ô ô Ŕ ŕ Ř ř Š š Ť ť Ú ú Ů ů Ü ü Ý ý Ž ž

Tyto znaky jsou z UTF8 kódu mapovány na interní jeden byte implicitně podle kódování $\mathcal{CS}$fontů, což je pro českou a slovenskou abecedu totéž jako kódování ISO-8859-2. Dále jsou mapovány UTF8 kódy všech znaků, které mají svou řídicí sekvenci definovanou v plainu nebo v $\mathcal{CS}$plainu. Jsou to tyto znaky:

```
plain: \ss ß  \l ł  \L Ł  \ae æ  \oe œ  \AE Æ  \OE Œ
       \o ø  \O Ø  \i ı  \j ȷ  \aa å  \AA Å
       \S §  \P ¶  \copyright ©  \dots …  \dag †  \ddag ‡
csplain: \clqq „  \crqq “  \flqq «  \frqq »  \promile ‰
```

Pokud je na vstupu jakýkoli jiný UTF8 kód, implicitně jej $\mathcal{CS}$plain nemá mapován, takže se vypíše na terminál a do logu následující varování:

```
WARNING: unknown UTF-8 code: `X = ^^xx^^xx' (line: ??)
```

a do tiskového výstupu se vloží na dané místo černý čtvereček. Uživatel si musí takový znak dodefinovat. Například po hlášení:

```
WARNING: unknown UTF-8 code: `Ñ = ^^c3^^91' (line: 42)
```

si uživatel do záhlaví dokumentu například doplní definici:

```
\mubyte\Ntilde ^^c3^^91\endmubyte % UTF8 kód mapován na \Ntilde
\def\Ntilde{\~N}                  % sekvence \Ntilde definována
```

Po této úpravě se při zpracování dokumentu už varování neobjeví a vstupní kód se zpracuje jako řídicí sekvence `\Ntilde`, která expanduje na `\~N`, takže v tiskovém výstupu se objeví Ñ.

Pokud se řídicí sekvence mapovaná pomocí `\mubyte` objeví v argumentu `\write` souboru, nebude expandovat, ale promění se při zápisu do souboru zpětně do odpovídajícího UTF8 kódu.

Během `\write` a při zápisu do logu se také zpět na UTF8 kódy proměňují interní byte, které byly mapovány. Implicitně tedy jsou to znaky české a slovenské abecedy vyjmenované výše. Ostatní nemapované interní byte s kódem větším jak 127 se přepisují podle dvouzobákové konvence např. takto: `^^ad`. Jestliže do deklarace dokumentu napíšete `\xprncodes=1`, budou se ostatní nemapované byty vypisovat přímo.

Příkazy `\mubyte` a `\endmubyte` jsou součástí encTEXu a jsou popsány v jeho dokumentaci. Stručně řečeno `\mubyte token string\endmubyte` zařadí do kódovací tabulky další údaj: `string` bude na vstupu převeden na interní `token` v TEXu a při činnosti `\write` bude `token` zpětně převeden na `string`.

Shrnuto: UTF8 kódy znaků české a slovenské abecedy jsou mapovány na interní byte a ostatní UTF8 kódy je třeba mapovat na kontrolní sekvence. Těch máme víceméně neomezeně mnoho, takže můžeme i v 8bitovém TEXu pracovat s rozsáhlým počtem různých vstupních UTF8 kódů.

Při použití jiných fontů pomocí `\input ctimes` (atd., viz sekci 7) je zavolán soubor `chars-8z.tex`, který mapuje další UTF8 kódy na řídicí sekvence, jež jsou pomocí `\chardef` propojeny se znakem fontu. Jsou to tyto znaky:

```
\euro €   \trademark ™   \registered ®   \ellipsis …
\textbullet •   \sterling £   \currency ¤
```

Soubor `chars-8z.tex` navíc předefinovává makra plainu `\P`, `\S`, `\dag`, `\ddag`, `\copyright`, `\lslash`, `\Lslash` pomocí `\chardef`, aby tyto řídicí sekvence vedly přímo na znak ve fontu.

V balíčku `enctex.tar.gz` (a podruhé i v `csplain.tar.gz`) jsou připraveny soubory, které mapují UTF8 kódy celých bloků UNICODE tabulky na řídicí sekvence a definují jejich výchozí chování. Můžete použít:

```
\input utf8lat1  % Latin-1 Supplement U+0080-U+00FF
\input utf8lata  % Latin Extended-A U+0100-U+017F
```

a možná v budoucnu i další. Po zavedení těchto souborů se nově deklarované UTF8 kódy mapují na řídicí sekvenci, např. ë se mapuje na `\edieresis` a tato sekvence expanduje na příslušné sestavení akcentu, v tomto příkladě na `\"e`. Na interní byte zůstávají mapovány jen znaky české a slovenské abecedy. Pro nově mapované řídicí sekvence jsou použity definice ze souborů `utf8lat*.tex` jen tehdy, pokud tyto řídicí sekvence nejsou definovány dříve.

Pokud předložíte $\mathcal{CS}$plainu (generovaném pro UTF8 kódování vstupu) dokument kódovaný jinak, než v UTF8, skoro jistě se dočkáte chybového hlášení:

```
! UTF-8 INPUT IS CORRUPTED !
  May be you are using another input encoding.
```

Je ovšem možné snadno přejít do módu, při němž UTF8 kódovaný $\mathcal{CS}$plain pracuje stejně jako $\mathcal{CS}$plain s kódováním ISO-8859-2. Stačí na začátek dokumentu napsat:

```
\mubytein=0  \mubyteout=0  \mubytelog=0  \xprncodes=1
```

Tuto práci vykoná také soubor `utf8off.tex`, tj. stačí napsat `\input utf8off`. Tento soubor navíc definuje makro `\clearmubyte`, které vymaže data deklarovaná pomocí `\mubyte...\endmubyte`.

Je dokonce možné mít na vstupu při sazbě jediného dokumentu různé soubory různě kódované. Není nutné při přechodu z jednoho kódování přepínat na druhé. Stačí na začátek dokumentu napsat:

```
\input mixcodes
```

a od této chvíle je možné na vstupu střídat texty kódované dle UTF8, ISO-8859-2 nebo CP1250 dle libosti. Všechny výstupy pomocí `\write` jsou kódovány podle UTF8 a jsou tedy opět připraveny k načtení.

V předchozím odstavci není zcela přesná formulace „střídat kódování dle libosti“. Platí to jen pro český text. Slovákům nefunguje »ľ« v CP1250. Pokud jim to vadí, napíšou `\shyph` před voláním `\input mixcodes`. Pak ale zase nefunguje »ž« v ISO-8859-2. Tyto dva znaky jsou v uvedených dvou kódováních poněkud v konfliktu. UTF8 vstup ale funguje bez problémů.

5. Interní kódování

Interní kódování TEXu je v $\mathcal{CS}$plainu implicitně nastaveno na kódování $\mathcal{CS}$fontů, které má znaky české a slovenské abecedy kódovány podle ISO-8859-2. Na interní kódování musejí správně navázat vzory dělení slov a použité fonty. Konverzi ze vstupního kódování do interního dělá input procesor TEXu, typicky encTEX.

$\mathcal{CS}$plain podporuje kromě kódování $\mathcal{CS}$fontů ještě další interní kódování: T1 (alias Cork) a Unicode. Posledně jmenované kódování je možné jen v 16bitových TEXových mašinách. Chcete-li přejít do kódování podle Corku nebo do Unicode, napište na začátek dokumentu:

```
\input t1code  % nastaveno kódování T1
\input ucode   % nastaveno kódování Unicode
```

Používáte-li 8bitovou TEXovou mašinu s encTEXem, pak `\input t1code` zařídí nejen správné nastavení `\lccode` atd., ale také přenastaví encTEXovou konverzní tabulku tak, že nyní se budou všechny UTF8 kódy, které odpovídají znakům v T1 kódování, převádět na interní byte v T1 kódování.

Používáte-li 16bitovou TEXovou mašinu, pak je bezvýhradně nutné na začátek dokumentu napsat `\input ucode`, protože tyto mašiny implicitně převádějí UTF8 kódy do Unicode a není jednoduché je přimět k jinému chování.

Po změně kódování pomocí `\input t1code` nebo `\input ucode` se automaticky změní chování příkazů `\chyph` a `\shyph` tak, že tyto příkazy zapnou vzory dělení slov ve správném kódování. Je ovšem nutné tyto příkazy psát *až poté*, co je změněno kódování.

Po změně kódování pomocí `\input t1code` nebo `\input ucode` se změní také vnitřní chování makra `\csaccents`: sekvence `\v`, `\'` atd. budou expandovat na znaky podle zvoleného kódování.

Změna kódování pomocí `\input t1code` nebo `\input ucode` neřeší zavedení tomu odpovídajících fontů. To znamená, že pokud neuděláte nic dalšího, zůstanou v $\mathcal{CS}$plainu zavedeny $\mathcal{CS}$fonty, jejichž kódování na `t1code` ani `ucode` nenavazuje. Takže ve výsledku uvidíte pomršenou češtinu nebo slovenštinu. Ovšem řešení je snadné: po změně kódování zavést třeba Latin Modern fonty příkazem `\input lmfonts`. Tento balíček maker má v sobě zabudovaný rozcestník podle zvoleného kódování, takže zavede ve všech třech případech fonty správně kódované:

```
\input lmfonts                  % Latin Modern v kodování CSfontů
\input t1code  \input lmfonts   % Latin Modern fonty v T1 kódování
\input ucode   \input lmfonts   % Latin Modern fonty v Unicode
\chyph      % aktivace vzorů dělení ve vybraném kódování
```

Podobnou vlastnost jako `lmfonts.tex` mají další „fontové soubory“, o kterých pojednává sekce 7.

Doporučuje se dodržovat pravidlo nejprve nastavit vnitřní kódování TEXu a pak řešit cokoli jiného. Například v sekci 4 byly zmíněny mapovací soubory `utf8lat1.tex` a `utf8lata.tex`. Pokud je zavedete dřív než `\input t1code`, budou znaky è, ã atd. mapovány na `\egrave`, `\atilde` atd. a tyto kontrolní sekvence budou definovány jako `\‘e`, `\~a` atd. Bude to sice fungovat, ale v T1 mají tyto znaky svůj vlastní kód. Možná by bylo tedy přirozenější mapovat znaky è, ã atd. v takovém případě přímo na jejich kódy. K tomu stačí, aby `\input t1code` *předcházel* před `\input utf8lat1`. Pak totiž makra v souboru `utf8lat1` budou vědět, že používáte T1 kódování, a vyřeší mapování přímočarým způsobem na interní byty. Dokonce tím vyřešíte znaky `\Eth`, `\Thorn` atd., které jsou implicitně v `utf8lat1.tex` definovány tak, že vypíší varování o nedostupnosti znaku, zatímco v T1 kódování jsou jednoduše i tyto znaky mapovány na své kódy.

Možná nyní někoho napadne otázka, co udělá `\input utf8lat1`, pokud předchází `\input ucode`. Nic. Unicode je možné použít jen v 16bitových mašinách, ale tyto mašiny nemají encTEX, takže v nich nefunguje mapování pomocí encTEXu. Ani není potřeba nic mapovat, protože v 16bitových mašinách asi zavedete rovnou Unicodový font.

6. Změna velikosti fontů

Nový *CS*plain nabízí elementárním způsobem možnost zvětšování nebo zmenšování fontů do požadované velikosti. Pomocí makra `\regfont` se registruje fontový přepínač, který má podléhat změně velikosti. Implicitně jsou registrovány přepínače `\tenrm` (Regular), `\tembf` (**Bold**), `\tenit` (*Italic*), `\tenbi` (***BoldItalic***), `\tentt` (`Mono`). Pak je potřeba definovat makro `\sizespec` jako „scaled1200“ nebo „at17pt“ a poté spustit makro `\resizeall`. To zavede pro každý registrovaný přepínač jeho původní font v nově požadované velikosti. Individuálně je možné zavést jedinému přepínači font v nové požadované velikosti (podle `\sizespec`) makrem `\resizefont`. Příklad:

```
\def\sizespec{scaled1200}
\resizeall % fonty \tenrm, \tenbf, \tenit, \tenbi, \tentt
           % nyní budou scaled 1200.
\def\sizespec{at8pt}
\resizeall % fonty \tenrm, \tenbf, \tenit, \tenbi, \tentt
           % nyní budou at 8pt.
```

```
\font\tenss=csss10 % nově zavedený font sans serif
\regfont\tenss     % registrujeme jej pro změnu velikostí
\def\sizespec{at13pt}
\resizeall % fonty \tenrm, \tenbf, \tenit, \tenbi, \tentt
           % a \tenss nyní budou at 13pt.
\def\sizespec{scaled700}
\resizefont\tenbi  % font \tenbi bude nyní scaled 700.
```

Veškerá nastavení z `\resizefont` a `\resizeall` jsou lokální v rámci skupiny. Můžete si definovat třeba makro `\small`, které přepne na 8bodové písmo takto:

```
\def\small{\def\sizespec{at8pt}\resizeall \tenrm}
Tady je normální text {\small a tady je zmenšený \it i v kurzívě.}
A tu je zase normálně veliké písmo.
```

Makro OPmac společně s makrem `ams-math.tex` rozvíjí tuto jednoduchou myšlenku změny velikostí fontů k téměř dokonalosti. Jednak nabízí uživateli pohodlnou změnu velikostí a nastavení třeba i relativně vzhledem k aktuální velikosti písma (což nemusí být 10bodů) a velikost správně nastavuje i všem matematickým rodinám včetně indexů a subindexů. Také interně pracuje s tabulkami designovaných velikostí metrik (např. `csr10 at12pt` je něco jiného než `csr12`) a zvětšuje/zmenšuje za použití metriky s vhodnou designovanou velikostí. Nabízí tedy srovnatelné možnosti jako NFSS, ale jednodušeji implementované a s pohodlnějším uživatelským rozhraním. Počet řádků kódu tohoto řešení je 5 (ve formátu $\mathcal{CS}$plain) a 40 v `ams-math.tex`. V jednoduchosti je síla.

Názvy fontových přepínačů `\tenrm`, `\tenbf` atd. zůstávají tedy shodné, jako v plainTEXu, ale jejich význam je typicky jiný. Nejsou to vždy fonty výhradně ve velikosti `\ten*`. Je třeba ten název chápat jako takovou trošičku pozůstalost z původního plainTEXu. Aspoň nám to signalizuje, že je to kontrolní sekvence, která je přepínačem nějakého fontu.

7. Fontové soubory

Fontový soubor je soubor, který zavádí fonty v jedné rodině, typicky v jednom kvartetu Regular, **Bold**, *Italic*, ***BoldItalic***. K přepínání do těchto variant jsou připravena makra `\rm`, `\bf`, `\it`, `\bi`. Tato makra volají fontový přepínač `\tenrm`, `\tenbf`, `\tenit` a `\tenbi`. Fontové soubory zavedou pro tyto fontové přepínače požadované fonty. Například soubor `ctimes.tex` nastaví rodinu fontů Times tak, že zavede font `\tenrm` jako ptmr8z (při zvoleném kódování $\mathcal{CS}$fontů), `\tenbf` jako font ptmb8z atd. Přitom `ptmr8z.tfm`, `ptmb8z.tfm` jsou metriky odpovídajících fontů rodiny Times. Kromě zmíněného kvartetu typicky fontové soubory zavádějí aspoň `\tentt` pro strojopis (makro `\tt`). Fontový soubor `ctimes.tex` pro tento účel použije Courier.

Zavedené fonty je možné dodatečně zvětšovat a zmenšovat způsobem popsaným v předchozí sekci.

„Klasické“ fontové soubory, které jsou součástí $\mathcal{CS}$plainu už mnoho let, jsou:

```
\input ctimes    % TimesRoman
\input chelvet   % Helvetica
\input cavantga  % AvantGarde
\input cncent    % NewCentury
\input cpalatin  % Palatino
```

Všechny tyto klasické fontové soubory zavádějí fonty v kódování $\mathcal{CS}$fontů[1] i v T1[2]. Unicode není v klasických fontových souborech podporován.

Některé fontové soubory připraví více než běžný fontový kvartet+tt. V každém případě se o tom zmíní na terminálu a v logu. Například po zavedení `chelvet.tex` se lze dočíst:

```
FONT: Helvetica - \rm, \it, \bf, \bi, \tt,
                  \cond\rm, ..., \cond\bi, \narrow
```

Je tedy zřejmé, že v případě Helveticy máme kromě běžných přepínačů `\rm`, až `\tt` ještě zúžené verze, do kterých se přepíná pomocí `\cond\rm`, `\cond\bf` atd. nebo je možné do nich jednorázově přepnout pomocí `\narrow`.

Vzhledem k tomu, že jedno písmeno `c` na začátku názvu nepůsobí příliš přehledně a může vzniknout soubor, který už v `texmf/` stromu je pro jiné účely, rozhodl jsem se nové fontové soubory pojmenovávat s předponou `cs-`. Jaké všechny fontové soubory jsou k dispozici zjistíte pomocí příkazu `tex cs-all` na příkazovém řádku. V tuto chvíli jsou připraveny k použití následující soubory:

```
\input cs-termes   % TeXgyre Termes (Times)
\input cs-heros    % TeXgyre Heros (Helvetica)
\input cs-cursor   % TeXgyre Cursor (Courier)
\input cs-adventor % TeXgyre Adventor (AvantGarde)
\input cs-bonum    % TeXgyre Bonum (Bookman)
\input cs-pagella  % TeXgyre Pagella (Palatino)
\input cs-schola   % TeXgyre Schola (NewCentury)
```

[1] Metriky všech těchto fontů jsem v říjnu 2012 kompletně přegeneroval. Původní verze řešila akcenty pomocí virtuálních fontů a kompozitů, což znamenalo, že ve výsledném PDF souboru nešlo vyhledávat česká slova a nefungovalo v PDF prohlížeči nabírání textu do clipboardu kvůli přenosu textu do jiné aplikace. Nové verze odkazují na znaky ve fontu přímo, a proto výsledné PDF uvedenými neduhy netrpí. V rámci přegenerování metrik jsem do nich přidal znaky Euro, Registered a Trademark.

[2] Metriky všech těchto fontů jsou už cca 15 let součástí TEXových distribucí a jsou vygenerovány programem fontinst. Fonty takto generované trpí chybou, která způsobuje bohužel jejich nepoužitelnost v češtině a slovenštině. Za znaky ď, ť a ľ se rozprostírá hrozivá mezera, která je dobře patrná zejména uprostřed slova. Tyto metriky asi přegenerovat nepůjde, protože jsou používány mezinárodní TEXovou komunitou. Karl Berry, se kterým jsem toto řešil, soudí, že pokud to českým a slovenským uživatelům 15 let nevadilo, není třeba spěchat s řešením.

```
\input lmfonts     % Latin Modern fonts
```

Všechny tyto uvedené fontové soubory zavedou fonty v kódování $\mathcal{CS}$fontů nebo T1 nebo Unicode (podle nastaveného vnitřního kódování). Tyto fontové soubory zavádějí kromě kvartetu+tt ještě celý kvartet ve verzi Caps and Small caps. Do těchto variant se přepíná pomocí `\caps\rm`, `\caps\it` atd.

Dále jsou k dispozici následující fontové soubory:

```
\input cs-antt    % Antykwa Torunska
\input cs-polta   % Antykwa Poltawskiego
\input cs-bera    % Bera
\input cs-arev    % ArevSans
\input cs-charter % Charter
```

a předpokládám, že si uživatelé budou schopni jednoduše dopsat další.

8. Matematika

Matematické fonty jsou řešeny v makru `ams-math.tex` nebo `tx-math.tex`. Soubor `ams-math.tex` zavádí Computer Modern fonty (nebo Latin Modern) pro základní matematické rodiny a $\mathcal{AMS}$fonty pro rozšiřující matematické rodiny. Nabízí kompletní sadu všech znaků z $\mathcal{AMS}$TEXu. Fonty jsou vizuálně kompatibilní s Computer Modern fonty. Proto je toto makro zavedeno fontovým souborem `lmfonts.tex`.

Soubor `tx-math.tex` zavádí TX fonty, které nabízejí nadmnožinu znaků známých z $\mathcal{AMS}$TEXu. Fonty jsou vizuálně kompatibilní s Times, ale snesou se i s dalšími rodinami fontů odvozených z dynamické antikvy. Proto je makro `tx-math.tex` použito pro matematiku ve vesměs všech ostatních fontových souborech. Následující text popisuje možnosti matematické sazby po zavedení souboru `ams-math.tex` nebo `tx-math.tex` ať už explicitně nebo prostřednictvím fontového souboru.

Je k dispozici příkaz `\setmathsizes[`⟨*text*⟩`/`⟨*script*⟩`/`⟨*scriptscript*⟩`]`, kterým se nastavují velikosti základní sazby, indexů a subindexů. Uvedené parametry jsou údaje v bodech (`pt`), ale tyto jednotky se tam nepíší. Po nastavení velikostí je možné použít `\normalmath` pro zavedení všech matematických rodin fontů v normálním duktu nebo `\boldmath` pro zavedení matematiky v tučném duktu. Například:

```
\setmathsizes[12/8.4/6]\normalmath % základní velikost 12pt
                                   % indexy 8.4pt a subindexy 6pt
```

Nastavení `\normalmath`, `\boldmath` je lokální v rámci skupiny. Následuje příklad pro zavedení matematiky pro případ, že uživatel napíše dolary s matematickou sazbou v nadpisu:

```
\def\titulfont{\def{at14pt}\resizefont\tenbf \tenbf
               \setmathsizes[14/9.8/7]\boldmath}
\def\titul#1\par{\centerline{\titulfont #1}}

\titul Kam konverguje $\int_x^\infty f(t){\rm d}t$?
```

Při použití makra OPmac je řešení nadpisů ještě snazší. V takovém případě ani tvůrce maker nemusí explicitně psát velikosti indexů a subindexů. Jednoduše použije příkaz `\typosize` nebo `\typoscale`. Indexy pak budou mít 70 % základní velikosti a subindexy 50 %.

Po `\normalmath` nebo `\boldmath` je zavedeno 12 matematických rodin (v případě `ams-math.tex`) resp. 14 matematických rodin (v případě `tx-math.tex`). Jedna matematická rodina sdružuje stejný font ve třech velikostech: základní, indexovou a subindexovou. Matematická rodina se zavádí buď proto, aby byla k dispozici další „matematická abeceda“ nebo proto, aby se daly použít další symboly typu $\int$, ∂ atd.

Jakmile mezi dolary začnete psát písmena bez použití matematického přepínače, je použita implicitní matematická abeceda `\mit`. Celkově jsou k dispozici následující abecedy:

`\mit`	`% matematická kurzíva`	$abc{-}xyz, ABC{-}XYZ$
`\it`	`% textová kurzíva`	$abc{-}xyz, ABC{-}XYZ$
`\rm`	`% textová antikva`	abc−xyz, ABC−XYZ
`\cal`	`% jednoduché cal. znaky`	$\mathcal{ABC}{-}\mathcal{XYZ}$
`\script`	`% kroucenější cal. znaky`	$\mathscr{ABC}{-}\mathscr{XYZ}$
`\frak`	`% fraktura`	$\mathfrak{abc}{-}\mathfrak{xyz}, \mathfrak{ABC}{-}\mathfrak{XYZ}$
`\bbchar`	`% zdvojené tahy`	$\mathbb{ABC}{-}\mathbb{XYZ}$
`\bf`	`% sans serif bold`	$\mathbf{abc{-}xyz, ABC{-}XYZ}$
`\bi`	`% sans serif bold slanted`	$\boldsymbol{abc{-}xyz, ABC{-}XYZ}$

Přepínač typu `\bf`, `\bi` může fungovat jinak v textovém a jinak v matematickém fontu. Takže skutečnost, že `\bi` přepíná v matematickém módu do sans serif bold slanted (vhodné pro sazbu vektorů nebo matic), zatímco v textovém módu do čtvrtého prvku fontového kvartetu (BoldItalic) není chyba, ale záměr.

V matematické sazbě jsou k dispozici stovky symbolů dostupných pomocí $\backslash\langle\mathit{něco}\rangle$, například `\alpha` α, `\geq` $\geq$, `\sum` $\sum$, `\sphericalangle` $\sphericalangle$, `\bumpeq`, $\bumpeq$. Seznam všech těchto symbolů najdete v dokumentaci k $\mathcal{AMS}$TEXu, která se typicky jmenuje `amsguide.ps`. Nebo v dokumentaci TX fontů `txfontsdoc.pdf`.

Makro `ams-math.tex` nastavuje jako implicitní abecedu matematickou kurzívu, která je mírně jinak kreslena než kurzíva textová. Naopak `tx-math.tex` nastavuje jako implicitní abecedu textovou kurzívu převzatou z „okolního“ textového fontu. Ta se ve vzorečcích spolu s textovým fontem bude esteticky snášet daleko lépe. Chcete-li toto implicitní chování změnit, je možné použít následující globální přepínače:

```
\itvariables    % implicitní abeceda bude textová kurzíva,
\mitvariables   % implicitní abeceda bude matematická kurzíva.
```

Chce-li uživatel zavést vlastní matematickou abecedu nebo nové symboly, má volné už jen dvě rodiny s čísly 14 a 15. Ostatní rodiny jsou obsazeny makrem `tx-math.tex`. Uživatel si vybere jedno z těchto dvou čísel (v následujícím příkladě si vybral 15) a napíše:

```
\input opmac  % zavede ams-math.tex a definuje makro \addto
\addto\normalmath {\loadmathfamily 15 ocrb10 } % OCRB metrika
\addto\boldmath   {\loadmathfamily 15 ocrb10 } % bold bohužel není
\def\ocrb{\fam15 } % definice matematického přepínače
\normalmath        % zavedení všech mat. fontů včetně nové rodiny
Zkouška: $a \cdot \ocrb x_{y_z}$
```

Možná se někomu může zdát tento příklad příliš komplikovaný. Ocení to asi pouze ti TEXisté, kteří se někdy pokusili zavést úplnou matematickou rodinu fontů v plainTEXu. A když to v potu tváře na mnoha řádcích nakonec vytvořili, nebylo to zvětšovací (pro nadpisy) ani zmenšovací (pro poznámky pod čarou). V tomto příkladě je nová rodina deklarována na dvou řádcích a je zvětšovací i zmenšovací.

9. Unicodové fonty

16bitové mašiny pracují interně v Unicode a je potřeba na to navázat Unicodovými fonty. XETEX disponuje rozšiřujícími možnostmi primitivu `\font`, který dovede rovnou načíst font formátu OpenType (přípona `.otf`). Takový font je Unicodový. Načtení fontu probíhá takto:

`\font\prepinac="[`⟨*filename*⟩`]:`⟨*fontfeatures*⟩`"` ⟨*sizespec*⟩

kde ⟨*filename*⟩ je název souboru (bez přípony `.otf`) a ⟨*sizespec*⟩ je obvyklé `at`⟨*dimen*⟩, `scaled`⟨*factor*⟩ nebo nic. Konečně ⟨*fontfeatures*⟩ jsou modifikátory fontu odděleny od sebe středníkem. Je potřeba vědět, jakými modifikátory font disponuje a kterým modifikátorům rozumí fontový loader. XETEXový fontový loader nabízí modifikátor `mapping=tex-text`, který způsobí aktivování TEXovských ligatur typu `---` → —. Obyčejné ligatury (fi, atd.) jsou aktivovány automaticky. Dále fonty disponují typickým modifikátorem `+smcp`, který způsobí, že font se chová jako Caps and small caps. Pokud napíšete neznámý nebo chybný modifikátor, nic se nestane, žádné chybové hlášení se nezjeví.

XETEX má svůj fontový loader slinkovaný s knihovnami operačního systému a nic dalšího neřeší, zatímco LuaTEX si vše dělá sám za pomocí programů v Lua. V LuaTEXu implicitně není primitiv `\font` vybaven schopností číst OpenType fonty, ale pomocí `\directlua` lze vložit Lua-kód, který to zařídí. Vyštípal jsem ze

souboru `luaotfload.sty` potřebné řádky Lua-kódu a vložil jsem je do souboru `luafonts.tex`. Nejsem si ale jist, zda vývojáři LuaTEXu neudělají v Lua-funkcích, které jsou tím Lua-kódem volány, nějakou nekompatibilní změnu a ono to přestane v budoucnu fungovat. Bohužel mi na konferenci k LuaTEXu nebyli schopni dát záruku trvanlivosti tohoto řešení.

Píšete-li svůj fontový soubor, je tedy potřebné načíst soubor `luafonts.tex`. Po provedení Lua-kódu z tohoto souboru se primitiv `\font` chová analogicky, jako v XeTEXu. V $\mathcal{CS}$plainu je soubor `unifam.tex`, který načítá jeden obvyklý kvartet OpenType fontů (plus tt a plus small caps). Můžete se podívat tam, jak je to uděláno.

Protože má LuaTEX odlišně implementovaný fontový loader, interpretuje bohužel jinak a jiné ⟨*fontfeatures*⟩. Implicitně nefungují v LuaTEXu v Unicodových fontech žádné ligatury, teprve po modifikátoru `+tlig` začnou fungovat ligatury typu `---` → — a dále modifikátor `script=latn` probudí k životu ligatury typu fi. Tyto a mnohé další ⟨*features*⟩ jsou popsány v dokumentaci `luaotfload.pdf`.

Jak bylo řečeno v sekci 7, $\mathcal{CS}$plain načítá v Unicode za jistých okolností Latin Modern fonty a TEXgyre fonty. Dělá to interně pomocí souboru `unifam.tex`. Než tyto fonty načtete (pomocí `\input lmfonts`, `\input cs-heros` atd.), je možné definovat makro `\fontfeatures` s modifikátory fontu. Není-li toto makro definováno, soubor `unifam.tex` si je dodefinuje takto:

```
\def\fontfeatures{mapping=tex-text;script=latn;+tlig}
```

10. Více jazyků

Implicitně načítá $\mathcal{CS}$plain při generování formátu následující vzory dělení:

- `\enPatt=0` ... (implicitní vzor z plainTEXu) v ASCII kódování
- `\csILtwo=5` ... čeština v ISO-8859-2
- `\skILtwo=6` ... slovenština v ISO-8859-2
- `\csCork=15` ... čeština v T1 kódování
- `\skCork=16` ... slovenština v T1 kódování
- `\csUnicode=115` ... čeština v Unicode (jen v 16bitové mašině)
- `\skUnicode=116` ... slovenština v Unicode (jen v 16bitové mašině)

Jednotlivé vzory dělení se v dokumentu zapínají pomocí `\enlang`, `\cslang` a `\sklang`. Příkaz `\enlang` zapne také `\nonfrenchspacing` a ostatní zapínají `\frenchspacing`. Jsou zachovány staré názvy přepínačů: `\ehyph=\enlang`, `\chyph=\cslang` a `\shyph=\sklang`. Přechod k novým názvům s aspoň dvěma písmenky pro jazyk je důsledkem nové možnosti v $\mathcal{CS}$plainu použít 54 různých jazyků. Jedním písmenkem je rozlišit nemůžeme.

V roce 2012 byl zcela přepracován soubor `hyphen.lan`, který je použit při generování formátu $\mathcal{CS}$plain a řeší čtení jednotlivých vzorů dělení slov. Na řádcích

48 až 160 tohoto souboru jsou za dvojtečkami skryty názvy vzorů dělení rozličných jazyků v kódováních T1 (Cork) nebo Unicode. Část výše jmenovaného souboru vypadá takto:

```
\let\csCork=y    % Czech
\let\skCork=y    % Slovak
:\let\deCork=y   % German
:\let\frCork=y   % French
:\let\plCork=y   % Polish
:\let\cyCork=y   % Welsh
:\let\daCork=y   % Danish
...
:\let\saUnicode=y   % Sanskrit
:\let\ruUnicode=y   % Russian
:\let\ukUnicode=y   % Ukrainian
:\let\hyUnicode=y   % Armenian
:\let\asUnicode=y   % Assamese
```

Stačí si vybrat jazyky, odstranit příslušnou dvojtečku a znovu vygenerovat formát $\mathcal{CS}$plain. Jazyky s abecedou, která se vejde do T1 kódování, mají připraveny vzory dělení `\*Cork` i `\*Unicode`, zatímco jazyky, které se do T1 kódování se svou abecedou nevejdou, mají připraveny vzory dělení jen `\*Unicode`.

Není nutné modifikovat soubor `hyphen.lan`. Požadované vzory dělení lze specifikovat na příkazové řádce při generování formátu pomocí `\let\vzor=y`. Třeba příkaz:

```
pdftex -ini -enc "\let\plCork=y \let\enc=u \input csplain.ini"
```

zavede navíc vzory polštiny v kódování T1. Nebo:

```
pdftex -ini -enc "\let\allpatterns=y \let\enc=u \input csplain.ini"
```

zavede navíc všechny dostupné vzory dělení `\*Cork`. Naproti tomu 16bitová mašina příkazem:

```
luatex -jobname pdfcsplain -ini "\let\allpatterns=y \input csplain.ini"
```

zavede všechny vzory dělení `\*Unicode`.

Jakmile je načten vzor dělení pro nový jazyk, je automaticky připraven k tomu odpovídající přepínač jazyka `\*lang`, tedy například `\delang` po načtení `\deCork` nebo `\pllang` po načtení `\plCork`.

Příkazy `\cslang` a `\sklang` implicitně přepínají do češtiny a slovenštiny s kódováním ISO-8859-2 a `\enlang` přepíná do angličtiny. Je-li zaveden další vzor dělení, který se vejde do ASCII, přepínač odpovídajícího jazyka také funguje, protože tyto vzory dělení pracují nezávisle na zvoleném kódování. Například `\itlang` (italština). Ostatní přepínače jazyků v kódování ISO-8859-2 nefungují.

Teprve po `\input t1code` začne `\cslang` a `\sklang` přepínat do vzorů kódovaných v T1 a budou fungovat i další jazyky se vzory dělení typu `\*Cork` načtenými při inicializaci formátu. Například `\delang`, `\pllang` atd. Konečně po `\input ucode` budou tyto přepínače jazyků přepínat do vzorů v Unicode.

Deklarace kódování pomocí `\input t1code` nastaví `\lccode` všech znaků, které jsou v T1 kódování definovány, takže všechny vzory dělení typu `\*Cork` budou fungovat. Na druhé straně `\input ucode` nastaví `\lccode` jen znaků české a slovenské abecedy. Používáte-li v 16bitové mašině další jazyk, je potřeba mu nastavit `\lccode` znaků jeho abecedy explicitně, jinak budou načtené a inicializované vzory dělení slov takového jazyka málo platné.

$\mathcal{CS}$plain definuje pro každý jazyk s načtenými vzory dělení makro `\lan:`⟨*číslo*⟩ jako ⟨*zkratku jazyka*⟩, například `\lan:5` stejně jako `\lan:15` expandují na `cs`. Programátor maker toho může využít. Programátor maker dále může využít toho, že přepínače jazyků `\czlang`, `\delang`, `\itlang`, `\frlang` atd. volají makro `\initlanguage{`⟨*zkratka jazyka*⟩`}`. Toto makro implicitně neudělá nic, ale programátor maker si je může předefinovat dle svého. Protože je `\initlanguage` zavolán těsně za `\language=`⟨*číslo*⟩ v kontextu:

```
\*lang ->  \language=⟨číslo⟩\relax
           \initlanguage{⟨značka⟩}\frenchspacing
           \lefthyphenmin=⟨lhm⟩\righthyphenmin=⟨rhm⟩%
           \message{⟨text⟩}
```

je možné, aby programátor maker odebral další inteligenci maker `\*lang` a převzal je do vlastních rukou. Třeba definuje:

```
\def\initlanguage #1#2#3\message#4{#3\csname lg:#1\endcsname}
```

Toto řešení přebírá z makra `\*lang` jen nastavení registrů `\lefthyphenmin` a `\righthyphenmin`, ale ruší implicitní `\message` i nastavení `\frenchspacing`. Místo toho se předpokládá, že budou definována makra `\lg:`⟨*značka*⟩, ve kterých budou tyto věci (a možná mnoho dalších jako třeba nastavení implicitního fontu, nastavení `\lccode`) řešeny pro každý jazyk individuálně.

$\mathcal{CS}$plain definuje kódovací přepínače pro vzory dělení slov: `\corklangs`, `\iltwolangs` a `\unicodelangs`. Funguje to takto: `\corklangs` ... `\cslang` (nyní je aktivní vzor dělení `\csCork`) a dále třeba ... `\iltwolangs` ... `\cslang` (nyní je aktivní vzor dělení `\csILtwo`). Makra `\iltwolangs`, `\corklangs`, a `\unicodelangs` jsou spuštěna při `\input il2code`, `\input t1code` a `\input ucode`, takže uživatel to typicky nemusí řešit.

Dále je potřeba vědět, že v 16bitové mašině můžete načíst vzory dělení `\*ILtwo` a `\*Cork` korektně jen pro češtinu a slovenštinu, protože další vzory dělení jsou čteny pomocí triku s aktivními znaky, který dekóduje 8bitový vstup. Nicméně typicky v 16bitovém TEXu není inicializován 8bitový vstup, takže se to poláme.

S volbou jazyka souvisí též správné nastavení automaticky generovaných slov, jako třeba Kapitola/Chapter, Obrázek/Figure. Tuto problematiku řeší makro OPmac a je to popsáno v technické dokumentaci k makru v sekci 3.5.

11. Balíček OPmac

Od prosince 2012 je součástí $\mathcal{CS}$plainu soubor maker `opmac.tex`. To neznamená, že bude tento soubor maker zaveden do formátu. Soubor `opmac.tex` bude jen přítomen v balíčku $\mathcal{CS}$plain a uživatelé jej budou moci jednoduše použít pomocí `\input opmac`. Momentálně je tento soubor maker ve fázi beta testování. Vlastnosti maker OPmac jsou popsány v samostatném článku a jsou také k dohledání na `http://petr.olsak.net/opmac.html`.

Summary: $\mathcal{CS}$plain of Year 2012

$\mathcal{CS}$plain existed since 1994 and it is a *gentle* extension of plainTEX so that it is possible to work in Czech and Slovak with it. This was true until October 2012, when the significant revisions and additions of $\mathcal{CS}$plain was made. The main change was done as a result of the decision to set the default input encoding of $\mathcal{CS}$plain to UTF8. Moreover $\mathcal{CS}$plain has many new properties now: the possibility of loading of all sorts of hyphenation patterns available for many languages, interoperability with 16-bit TEX engines (LuaTEX, XeTEX), an effective work with fonts including mathematical fonts, easy switching of internal encoding including Unicode, user-friendly macro package OPmac. In the default configuration $\mathcal{CS}$plain remains still as gentle extension of plainTEX backward compatible with previous versions. New options are easily accessible via `\input` and if they are used it is far to speak only about *gentle* extension. On the contrary, it is a strong competitor of all huge macro extensions of TEX and it wins by simplicity, efficiency of solutions, directness and ease of use. New $\mathcal{CS}$plain is available at `http://petr.olsak.net/csplain.html`.

Petr Olšák
petr@olsak.net