Pedagogická fakulta
Faculty of Education

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích

Pedagogická fakulta

Katedra informatiky

Typografie a sazba odborných dokumentů v XeTeX

Typography and typesetting of documents in XeTeX

Bakalářská práce

Vypracoval: David Kocur
Vedoucí práce: PaedDr. Petr Pexa, Ph.D.

České Budějovice 2016

Prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury. Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích dne 2. června 2016

David Kocur

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH
Fakulta pedagogická
Akademický rok: **2014/2015**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**David KOCUR**
Osobní číslo:	**P13105**
Studijní program:	**B7507 Specializace v pedagogice**
Studijní obor:	**Informační technologie a e-learning**
Název tématu:	**Typografie a sazba odborných dokumentů v XeTeX**
Zadávající katedra:	**Katedra informatiky**

Zásady pro vypracování:

Cílem bakalářské práce je zpracovat typografickou Unicodovou technologii XeTeX, která je rozšířením TeX, podporuje speciální znakové sady pro písmo většiny světových jazyků a obsahuje i podporu pro aktuální písma TrueType, OpenType a AAT. Původně byla vyvíjena pouze pro operační systém Mac OS X, ale nyní je již k dispozici i pro další OS. Je distribuována pod licencí MIT, jedná se tedy o open-source platformu. Toretická část práce bude obsahovat popis a vývoj syntaxe XeTeXu a možnosti jeho implementace do TeXu, bude podrobně otestována funkčnost a typografická správnost speciálních znakových sad a provedeno porovnání s běžnými textovými editory, které podporují obdobné techniky práce se speciálními znaky. Součástí bude sada příkladů, která demonstruje praktické využití a funkčnost tohoto aktuálního rozšíření TeX pro OS Windows a další.

Rozsah grafických prací: **CD ROM**

Rozsah pracovní zprávy: **40**

Forma zpracování bakalářské práce: **tištěná**

Seznam odborné literatury:

1. **SIL International. Sil [online]. 2015 [cit. 2015-04-14]. Dostupné z: http://www.sil.org**
2. **XeTeX. The Comprehensive TEX Archive Network [online]. 2015 [cit. 2015-04-14]. Dostupné z: http://ctan.org**
3. **Unicode-based TEX. X?TEX [online]. 2007 [cit. 2015-04-14]. Dostupné z: http://xetex.sourceforge.net**
4. **TeX Users Group. XeTeX on the Web [online]. 2014 [cit. 2015-04-14]. Dostupné z: https://tug.org/xetex/**
5. **BODONI, Giambattista. Manual of Typography. [Köln: Taschen, 2010. ISBN 9783836505536.**

Vedoucí bakalářské práce: **PaedDr. Petr Pexa, Ph.D.**
Katedra informatiky

Datum zadání bakalářské práce: **27. dubna 2015**

Termín odevzdání bakalářské práce: **29. dubna 2016**

Mgr. Michal Vančura, Ph.D.
děkan

L.S.

doc. PaedDr. Jiří Vaníček, Ph.D.
vedoucí katedry

V Českých Budějovicích dne 27. dubna 2015

Poděkování

Rád bych chtěl poděkovat panu PaedDr. Petru Pexovi za odborné vedení, ochotu a spolupráci. Jeho rady a připomínky mi byly vždy inspirací, díky kterým jsem byl schopen tuto práci zpracovat.

Dále bych chtěl poděkovat mé rodině a kamarádům, kteří mi byli vždy oporou.

Abstrakt

V mé bakalářské práci se budu zabývat typografickým Unicodovým rozšířením technologii TEX XƎTEX, který podporuje speciální znakové sady pro písmo většinu světových jazyků, dále umožňuje podporu pro technologie písma jako TrueType a OpenType. Dále se budu zabývat historií vývoje XƎTEX a jeho implementací do TEX. Rovněž bude podrobně otestována funkčnost a typografická správnost speciálních znakových sad a porovnání s použitím běžných textových editorů, které se snaží zabývat podobnou problematikou se speciálními znaky. Součástí bude také samotná práce vypracovaná v XƎTEXu, která demonstruje praktické využití a funkčnost tohoto rozšíření.

Klíčová slova

TEX, XƎTEX, LATEX, typografie, písma, znakové sady

Abstract

In my thesis, I will deal with typographic Unicode extension of technology TEX XƎTEX, which support special charset for fonts of most world's languages, it also enables support for font technologies like TrueType and OpenType. I shall then address the history of develop of XƎTEX and his implementation into the TEX. I will also closely test functionality and typographical correctness of special charsets and comparison with usage of common text editors which is seeking to address similar issues with charsets. Part of the work itself will also be drawn up in the XƎTEXu, which demonstrates the practical use and the functionality of this extension.

Key words

TEX, XƎTEX, LATEX, typography, fonts, charsets

Obsah

1 Úvod

Psaní odborných prací a dokumentů je velkou součástí života mnoha lidí, ať už z důvodu pracovního či akademického. Proto je kladen značný důraz na korektní typografii a sazbu daných prací a dokumentů. Samotná sazba a typografie dnes už není zdaleka tak problematická jako dříve díky celé řadě editorů a jazyků, které uživateli značně usnadní práci. Řada editorů bohužel nepodporuje písma, která mají speciální formátování a nejsou součástí vestavěných písmových sad v operačním systému (například Graphite, OpenType, nebo AAT[1]).

Při psaní prací v TEXu tenhle problém začalo řešit rozšíření XETEX, které umožnilo podporu a okamžitý překlad unicode formátům a různým písmům, které se po přeložení nezobrazovala správně, nebo vůbec. Většina světových jazyků obsahuje ve své slovní zásobě nespočetně speciálních znaků a symbolů (v českém jazyce jsou to například háčky a čárky), které se v anglickém jazyku nenacházejí a tím je práce v TEXu v dané jazykové sadě značně omezena. Aby byla možnost použití těchto znakových sad a fontů, ne vždy stačí jazykový balíček, a proto je XETEX velice užitečným rozšířením, které je už dnes téměř nutností. Jedna z největších výhod je, že výstup XETEXu je vždy v unicodu, tím odpadají problémy s kódováním dokumentu.

XETEX je velice zajímavé téma pro bakalářskou práci, jeho využití se stává nedílnou součástí práce s TEXem, která mě od první chvíle zaujala. Na internetu není zatím moc prací, které se zabývají problematikou XETEXu, bohužel velká část těchto prací a dokumentů je vždy v anglickém jazyce a práce, které jsou v českém jazyce se XETEXem zabývají velice povrchně, nebo vůbec. Proto věřím, že má práce bude mít v budoucnu využití a bude užitečná pro české uživatele TEXu a XETEXu.

1.1 Cíle práce

Cílem bakalářské práce v teoretické části je seznámení s technologií TEX a jeho rozšiřovacím balíčkem XETEX, který umožňuje typograficky správnou

[1] Apple Advance Typography

sazbu odborných dokumentů za využití speciálních znakových sad. Dále se zaměřím na podrobný popis syntaxe a základních příkazů v XeTeX pro jednotlivé znakové sady jako jsou například OpenType nebo TrueType. V další části se budu zabývat historií a vývojem technologií XeTeX a jeho implementací do TeX. V praktické části otestuji XeTeX a jeho implementaci do TeX editorů, dále jej porovnám s technologií, která se zabývá podobnou problematikou. Na konec bude samotná práce vysázena a přeložena pomocí rozšíření XeTeX.

1.2 Východiska práce

Při vytváření odborných prací se klade velký důraz na korektní typografii a sazbu, v těchto pracích je mnohdy potřeba použití různých speciálních znakových sad, které mají odlišné řezy a velikosti, například písma různých světových jazyků, nebo různé matematické vzorce, kde je kladen velký důraz na sazbu jednotlivých symbolů. K psaní odborných prací je pravděpodobně nejlepší variantou použití technologii TeX, avšak ani tahle technologie neumožňovala typograficky korektní sazbu těchto speciálních znaků, ale jen do doby kdy byl vyvinut speciální balíček XeTeX, který umožnil podporu pro tyhle speciální znakové sady a písma jako TrueType, OpenType a AAT. V dnešní době se tahle technologie rozšiřuje také u nás, hlavně díky dostupnosti XeTeX i na jiných platformách než Mac OS X.

1.3 Metody práce

V úvodu bakalářské práce bude rozebrána problematika typografie speciálních znakových sad a písma tak, aby se čtenář dokázal orientovat v dáne problematice a byl schopen použít XeTeX ve vhodných situacích. V popisu bude rozebráno jaké jsou rozdíly a výhody používání XeTeXu , také jak byl samotný XeTeX vyvíjen a implementován do TeXu. Dále bude XeTeX porovnán s technologií LuaTeX, která se zabývá podobnou problematikou a na konec bude tato práce přeložena a vysázena pomocí rozšíření XeTeX.

2 Historie a základ XeTEX

2.1 O čem je XeTEX ?

XƎTEX byl vyvinut v SIL[2] autorem Jonathanem Kew. Jeden z nejdůležitějších cílů XƎTEXu bylo umožnit TEXu přímo používat písma, která jsou dostupná na operačních systémech. Technicky je to možné implementovat v TEXu použitím příkazu `\font` , který se zeptá operačního systému, aby nalezl určené písmo (za použití reálného názvu písma, které je známo operačnímu systému, na rozdíl od zakódovaných souborů) v jakékoliv dostupné sadě písma. To znamená, že všechny písma, které jsou dostupné v operačním systému a uživatelském rozhraní, se stanou použitelná pro sazbu v XƎTEXu se stejnými názvy, z toho důvodu už není zapotřebí použití specifických TEX příkazů. Když XƎTEX dostane pokyn k použití písma, tak lokalizuje samotný soubor daného písma (dokáže přečíst všechny varianty písma jako OpenType, PostScript Type 1 a TrueType) bez `.tfm` souboru. Při vytváření odstavců XƎTEXu získává metrické informace o znakové sadě přímo ze souboru daného písma. Kromě toho se musí postarat o složitost mapování znaků do sad, zejména u kurzívy a non-Latin textů. Proto XƎTEX nesestavuje své odstavce ze seznamu znaků, ale ze slov, z nichž se každé skládá z úplného chodu po sobě následujících znaků v daném písmu. Jazykové a typografické změny a efekty jsou delegovány na příslušný "layout engine" (XƎTEX má rozhraní k ATSUI[3], ICU[4] and SIL Graphite). Výsledkem je pole písmen s jejich pozicí, která reprezentuje slovo tak, jak je rozvržena použitím daného písma. Z tohoto seznamu slov, která jsou prokládána "lepidlem", je sestaven odstavec. Samozřejmě, když je dělení požadováno, slova mohou být rozebrána a dána dohromady za použití break pozicování. Nicméně základní myšlenka zůstává: shromáždit běh znaků, předat je jako úplné jednotky knihovnám písma, které jsou schopny pracovat s rozložením na úrovní jednotlivých slov.[1]

Ve výchozím stavu se LATEX NFSS[5] systém zabývá pouze s "makrosko-

[2]Summer Institute of Linguistics
[3]Apple Type Services for Unicode Imaging
[4]International Components for Unicode
[5]New font selections system

pickými“ variacemi písma, jako jsou váha, tvar a velikost. Fontspec rozšiřuje LaTeX zacházení s písmem poskytováním podpory pro “font features“, které umožňují uživateli v každém bodě dokumentu měnit širokou škálu typografických detailů pomocí využití různých instancí písma.[1]

2.2 Historie vývoje

- V dubnu 2004 byl vydán XeTeX 0.3, zatím jen pro Mac OS X,
  - zabudovaná podpora unicode,
  - přístup ke všem písmům nainstalovaných na operačních systémech,
  - AAT pro typografické vlastnosti,
  - Quicktime pro grafickou podporu.
- V únoru 2005 byla vydána verze XeTeX 0.9,
  - podpora OpenType,
  - kompatibilita s více důležitými LaTeX balíčky.
- V dubnu 2006 byl vydána verze podporující běh na Linuxu (Bacho TeX),
  - první veřejné oznámení o dostupnosti XeTeX.
- V červnu 2006 Akira Kakuto oznámil dostupnost XeTeX pro operační systémy Windows,
- V lednu 2007 byla vydána verze XeTeX 0.994 pro všechny binární platformy,
- V září 2007 verze XeTeX 0.997 je dostupna společně s MikTeX 2.7 beta.

3 Standardy a formáty písma

Na internetu je celá řada standardů a formátů písma. Tyto standardy se mohou lišit výstupem tisku a podporou znaků. Standard písma jde obvykle poznat podle souboru, ve kterém je font uložen. Tyto soubory se většinou liší koncovkou, která specifikuje daný standard. Například pro standard TrueType je koncovka souboru `.ttf`.[3]

3.1 TrueType

TrueType je digitální technologie písma, která byla vyvinuta společností Apple Computer a dnes je využívána Applem i Microsoftem v jejich operačních systémech. Micorsoft vytvořil miliony kvalitních TrueType fontů ve stovkách různých stylů, které jsou zahrnuty v nejpopulárnějších TrueType font balíčcích.[4]

TrueType fonty nabízí největší dostupnou kvalitu pro počítačové obrazovky a tiskárny. Jejich součástí je také celá řada feature, díky kterým je jejich použití velice snadné.[4]

Pokud používáte TrueType fonty na webové stránce nebo ve Word dokumentu, je možné je přidat k dotyčnému souboru, aby dané písmo mohlo být prohlíženo i uživateli, kteří tento font nemají nainstalovaný na svém počítači. Je také důležité zmínit, že TrueType verze Windows a Macintosh nejsou vzájemně kompatibilní.[7]

3.1.1 Historie TrueType

Apple poprvé zahrnul podporu TrueType fontů v operačním systému Macintosh, System 7 v roce 1991. Současný vývoj zahrnuje podporu TrueType GX, který rozšiřuje TrueType formáty jako novou součástí grafické architektury QuickDraw GX pro MacOS.[5]

Microsoft poprvé zahrnul podporu TrueType fontů v operačním systému Windows 3.1 v roce 1992. Krátce poté Microsoft začal TrueType přepisovat a vylepšovat jeho kvalitu, výkon a odstranil bugy. Nově upravená verze TrueType formátu byla poprvé zpřístupněna společně s operačním systémem

Windows NT.[5]

3.1.2 TrueType Rasterizer

Technologie TrueType fontů se skládá ze dvou hlavních části. První částí je samotný soubor, který obsahuje popis písma a druhou částí je program, který umožňuje čtení souboru s písmem a generuje bitmapy. Tento program se nazývá TrueType Rasterizer.[6]

TrueType Rasterizer je počítačový program, který je obvykle součástí operačního systému, nebo programu pro podporu tiskárny. Kvůli tomu byl tento program napsán s velmi kvalitně navrženým rozhraním a jednoznačnou modulární strukturou, která je přenosná v jazyce C.[6]

Práce TrueType Rasterizeru je generování znakových bitmap pro obrazovky a tiskárny. Dosahuje toho provedením následujících úkolů:

- čtení stručného popisu znaku ze souboru TrueType,
- škálování popisu obrysu znaku na požadovanou velikost a rozlišení zařízení,
- přizpůsobení obrysu pro pixelový grid,
- vyplnění přizpůsobeného obrysu pixely.

3.2 PostScript

PostScript je speciální jazyk určený k popisu tisknutelných dokumentů. Přesně popisuje konkrétní podobu stránky a některé (zpravidla ty dražší) tiskárny umí dokumenty v tomto jazyku přímo tisknout. Mezi výhody patří to, že s již hotovým dokumentem lze ještě částečně manipulovat (otáčet, zrcadlit, zvětšovat, dávat více stránek na jeden papír atd.) bez ztráty kvality.[8]

3.3 OpenType

OpenType je sjednocením dvou nejrozšířenějších formátů fontů v současnosti: PostScriptu a TrueTypu. Nebo také obohacením TrueType formátu o možnosti PostScriptu. Adobe a Microsoft totiž vyvinuly technologii, která přináší

nové typografické a sazební vymoženosti. OpenType by se měl stát standardem pro vysoce kvalitní písma nejen v tisku, ale i na webu.[9]

Dvě hlavní výhody standardu OpenType je jeho kompatibilita mezi platformami (stejný soubor fontu je podporován na operačních systémech Windows i Macintosh) a jeho schopnost podporovat široce rozšířené znakové sady a funkce, které poskytují bohatší jazykovou podporu a pokročilé typografické kontroly.[11]

3.3.1 Historie OpenType

V roce 1996 Adobe a Microsoft ohlásili, že budou společně vyvíjet nový formát fontu, který má spojit dvě dominantní technologie PostScript s TrueType. Tato nová technologie dostala název OpenType. Pro Adobe toto byl dobrý krok ke konverzi jíž existujících knihoven a stabilizaci ve Windows font marketu. Microsoft začal podporovat OpenType font formát od vydání operačního systému Windows 2000 s výhledem konkurovat na trhu, kde dominoval Macintosh.[12]

První OpenType fonty se na trhu objevili v roce 2000. Adobe přizpůsobili ATM[6], aby i ostatní operační systémy podporovali OpenType fonty vedle Windows 2000. Od té doby všechny operační systémy zahrnují nativní podporu pro OpenType.[12]

3.3.2 OpenType features

Pomocí OpenType features je umožněno písmu se chovat "chytře". Tyto funkce mohou umožnit velice jednoduché věci (například změny písmen na malé kapitálky), nebo mohou provádět složitější věci (například vkládání ligatur).[10]

Ústředním bodem diskuse o podpoře OpenType featuers spočívá mezi rozdíly znaků a piktogramů. Znaky jsou body kódu přidělené standardem Unicode, který představuje tu nejmenší sémantickou jednotku jazyka (například písmeno). Piktogramy jsou specifické formy, které mohou být těmito znaky použity. Jeden znak může odpovídat několika piktogramům (například

[6]Adobe Type Manager

malé písmeno c a kapitálka c jsou dva samostatné piktogramy, ale používají jeden stejný znak). Jeden piktogram může představovat více znaků, jako je tomu v případě ligatury ”ffi”, která odpovídá sledu tří znaků, ale jednomu piktogramu[11]

3.4 Graphite

Graphite je balíček, který může být použit k vytvoření ”inteligentních fontů”, které jsou schopny zobrazovat psací systémy s různými komplexními případy. Inteligentní font obsahuje nejen tvary písmen, ale i další instrukce uvádějící, jak kombinovat a umístit písmena v různých komplexních situacích.[13]

Graphite byl vyvinut s cílem zajistit flexibilitu pro menšinové jazyky, které potřebují být často psány s poněkud odlišnými pravidly než běžně používané jazyky, které využívají ten stejný skript.[13]

Graphite je schopen zvládnout následující komplexní případy:

- kontextová tvarování,
- ligatury,
- přeřazování,
- rozdělení piktogramů,
- diakritika,
- komplexní pozicování.

3.5 Unicode

Unicode je znakový kódovací standard, který má široké uplatnění. Software společnosti Microsoft využívá Unicode ve svém jádru. Počítače v podstavě pracují pouze s čísly. Ukládají písmena a další znaky tím, že pro každé z nich přiřadí číslo. Předtím, než byl Unicode vynalezen, tak zde byla celá řada kódovacích systémů pro přiřazování těchto čísel. Bohužel žádné kódování nemohlo obsahovat dostatek znaků, například Evropská unie sama potřebuje

několik různých kódování pro pokrytí všech jazyků, které se od sebe značně liší. Dokonce i pro jeden jazyk jako je angličtina, ani jedno kódování nebylo dostačující pro všechny písmena, interpunkce a technické symboly, které se běžně používají.[14]

Tyto kódovací systémy byly také v rozporu jeden s druhým. To znamená, že dvě kódování mohou používat stejné číslo pro dva různé znaky, nebo používat různá čísla pro stejný znak. Jakýkoliv počítač musí podporovat mnoho různých kódování, ale vždy, když jsou data předávány mezi různými znakovými sadami,nebo platformami, se vystavují riziku poškození.

Unicode poskytuje jedinečné číslo pro každý znak bez ohledu na platformu, bez ohledu na program a bez ohledu na jazyk. Standard Unicode byl přijat ve světě takovými firmami jako Apple, HP, IBM, JustSystems, Microsoft, Oracle, SAP, SunSybase...[14]

4 Příkazy XeTEXu

4.1 Příkaz \font

Tradiční příkazy v TEXu byly psány jako `\font\1=[název fontu]` s různými atributy připojeny jako `‘at 10pt’`, nebo `‘scaled 1.2’`, v tomhle případě upravující velikost písma. Teto příkaz by samozřejmě v XƎTEXu stále fungoval. VXƎTEXu byl tento příkaz velmi vylepšen a rozšířen.[2]

Nová rozšířená verze příkazu má následující tvar:

```
\font\1="{identifier}{options}:{features}"{TEX options}
```

Příklad 1: Nová verze příkazu

Jediná povinná část tohoto kódu je identifikátor, pokud by byl v hranatých závorkách, XƎTEX by ho přeložil jako název určitého písma. Pokud identifikátor nemá závorky, na zadaný název se dívá jako název souboru i jako název systémového písma. Když se používá název písma, písmo je hledáno přes operační systém za pomoci `fontconfig` knihovny. Spuštěním `fc-lsit` by mělo ukázat dostupná písma. [2]

```
\font\1="Liberation Serif"
```

Příklad 2: Hledání fontu naisntalované v OS

Písma mají celou řadu vnitřních názvů, které XƎTEX porovnává v následujícím pořadí:

- celý název,
- pokud název obsahuje spojovník, tak je rozdělen do Family-Style páru a poté porovnán,
- název PostScriptu,
- název rodiny, pokud nalezne více než jednu možnost.
  - podívá se po písmu s bitovým nastavením v operačním systému ”regular”, pokud není shoda,

– podívá se po písmu se styly v tomto pořadí "Regular", "Plain", "Normal", "Roman".

Když se používá název souboru, musí být použit `xdvipdfmx` ovladač (je automaticky nastaven). Je prohledán aktuální adresář a texmf strom pro shodné názvy souborů, nebo cesta k adresáři může být vložena v deklaraci názvu písma. [2]

```
\font\2="[lmroman10-regular]"
```

Příklad 3: Nalezne lmroman10-regular font v jakémkoliv stromě

```
\font\3="[/myfonts/fp9r8a]"
```

Příklad 4: Nalezne font fp9r8a ve složce myfonts

Pokud má soubor koncovku `.otf`, `.ttf`, nebo `.pfb,` XeTEX jej nalezne. Daná koncovka souboru může být uvedena výslovně. Jestli je soubor kolekcí fontů (např. `.ttc` nebo `.dfont`), index toho písma lze zadat pomocí dvojtečky, následovaný zero-based[7] indexem uvnitř hranatých závorek.

```
\font\4="[myfont.ttc:1]"
```

Příklad 5: Načte druhý font ze souboru myfont.ttc

4.2 Font options

{font options} jsou aplikované pouze tehdy, jestli je font vybrán z operačního systému. Může to být jakékoliv zřetězení z následujících příkazů:[2]

[7] index který začína od 0 nikoliv 1

/B	Použije tučnou verzi vybraného fontu
/I	Použije kurzívu vybraného fontu
/BI	Použije kurzívu a tučnou verzi vybraného fontu
/IB	Použije kurzívu a tučnou verzi vybraného fontu
/S=x	Použije verzi vybraného fontu odpovídající optické velkosti x pt
/AAT	Vykreslí font výhradně pomocí AAT (jen pro Mac OS)
/OT	Vykreslí font výhradně pomocí OpenType (ve verzi 0.9999)
/GR	Vykreslí font výhradně pomocí Graphite
/ICU	Vykreslí font výhradně pomocí OpenType (zastaralé ve verzi 0.9999)

Tabulka 1: Příkazy font options

4.3 Font features

{font features} je čárkou, nebo středníkem oddělující seznam aktivací a deaktivací různých variant OpenType, Graphite, nebo AAT vlastností, které se budou lišit podle vybraného písma. V porovnání s {font options}, {font features} funguje nezávisle na tom, zda-li je soubor vybrán přes operační systém, nebo podle jména.[2]

4.3.1 OpenType, Graphite a AAT "features"

OpenType font features jsou vybrány se standardními tagy[8]. Tyhle tagy mohou být odděleny čárkou nebo středníkem s prefixem + pro jejích spuštění a prefixem - pro jejich vypnutí, dále následuje = s zero-based indexem pro výběr z několika alternativích features.

```
\font\liber="Linux Libertine O/I=5:+smcp" at 12pt
\liber Slunce svítí nad hlavou
```

Příklad 6: Ukázka OpenType font features

Výstup tohoto kódu v XƎTEXu vypadá následovně:

[8] http://www.microsoft.com/typography/otspec/featuretags.htm

Slunce svítí nad hlavou

Rozdíly mezi features záleží na jazyku nebo na skriptu, malé množství OpenType features se (pokud existují) aktivuje automaticky.

```
\font\antt="Antykwa Torunska" at 12pt \antt 0
\font\antt="Antykwa Torunska:+aalt=0" at 12pt \antt 0
\font\antt="Antykwa Torunska:+aalt=1" at 12pt \antt 0
\font\antt="Antykwa Torunska:+aalt=2" at 12pt \antt 0
\font\antt="Antykwa Torunska:+aalt=3" at 12pt \antt 0
\font\antt="Antykwa Torunska:+aalt=4" at 12pt \antt 0
```

Příklad 7: Rozdíly mezi OpenType features

Výstup tohoto kódu v XeTeXu vypadá následovně:
0 o ⁰ ₀ ⁰ ₀

AAT font features a Graphite font features jsou specifikovány v rámci každého řetězce, raději než standardizovanými tagy.

```
\font\gra="Charis SIL/GR:Small Caps=True" at 12pt
\gra Já do lesa nepojedu
```

Příklad 8: Graphie font

Výstup tohoto kódu v XeTeXu vypadá následovně:
JÁ DO LESA NEPOJEDU

4.3.2 Nastavení pro všechny fonty

Některé font features mohou být aplikovány pro jakýkoliv font. Tohle je příklad takových features.

- mapping=<font map>
  - využívá specifické mapování fontů pro příslušné písmo. Při mapování použije na poslední chvíli TECKit engine pro změnu unicodových znaků ve fázi zpracování daného zdroje. Na příklad `mapping=tex-text` umožní klasické mapování z ascii "—" na řádné typografické symboly "—".

- color=RRGGBB[TT]
  - troji dvojce hexadecimálních hodnot, které specifikují barvu v RGB, s volitelnou hodnotou pro transparentnost.
- letterspace=x
  - přidá x/S mezeru mezi slova, kde S je velikost písma.
- embolden=x
  - zvýší obal každého písmena o zadanou hodnotu (jednotlivá slova vypadají tučněji). X = 0 ponechá písmo beze změny, x=1.5 je vhodná základní hodnota.
- extend=x
  - roztáhne každé písmeno horizontálně o x (x = 1 ponechá písmo beze změny).
- slant=x
  - klopí každé písmeno o zadanou hodnotu. X = 0 ponechá písmo beze změny, x = 0.2 je vhodná základní hodnota . Sklon je dán x = R/S kde R je umístění bodu horního okraje písmena a S je velikost v pt.

4.3.3 OpenType skripty a podpora jazyků

Chování jednotlivých OpenType features může být ovlivněno různými skripty[9], nebo jazykem[10]. Jsou reprezentovány určitým tagem.

- script=<script tag>
  - vybere požadovaný skript pro font.

[9] http://www.microsoft.com/typography/otspec/scripttags.htm
[10] http://www.microsoft.com/typography/otspec/languagetags.htm

- language=<lang tag>
  - vybere požadovaný jazyk pro font.

4.3.4 AAT font podpora

- height=x
  - zvolí hodnotu fontu pro výšku.
- width=x
  - zvolí hodnotu fontu pro šířku.
- optical size=x
  - zvolí optickou velikost v jednotce bodu.

5 Nové XeTEX příkazy

S vývojem XeTEXu přichází také celá řada nových příkazů, která ulehčuje práci v TEXu s OpenType , AAT a Graphite fonty. Také přináší vylepšení pro sázení matematických znaků.

5.1 Příkazy pro primitivní fonty

\XeTeXtracingfonts

- pokud má nenulovou hodnotu, tak nahlásí, kde se nachází fonty v souboru.

\XeTeXfonttype{font}

- rozšíří se k odpovídajícímu číslu , pro který překladač je používán.

```
\newcommand\whattype[1]{%
\texttt{\fontname#1} řpekládá
\ifcase\XeTeXfonttype#1\TeX\or AAT\or
OpenType\or Graphite\fi.\par}
\font\1="cmr10"
\font\2="Charis SIL"
\font\3="Charis SIL/OT" \
\whattype\1 \whattype\2 \whattype\3
```

Příklad 9: XeTeXfonttype

Výstup tohoto kódu v XeTEXu má následující podobu:
`cmr10` překládá TEX.
`"Charis SIL"` překládá OpenType.
`"Charis SIL/OT"` překládá OpenType.

\XeTeXfirstfontchar{font}

- rozšíří kód o první znak nacházející se v {font}.

\XeTeXlastfontchar{font}

- rozšíří kód o poslední znak nacházející se v {font}.

```
\font\1="Charis SIL"\1
První znak v Charis SIL je:
"\char\XeTeXfirstfontchar\1"a poslední znak je:
"\char\XeTeXlastfontchar\1"
```

Příklad 10: XeTeXfirstfontchar a XeTeXlastfontchar

Výstup tohoto kódu v XeTeXu má následující podobu:

První znak v Charis SIL je: " "a poslední znak je: "ט"

\XeTeXglyph{glyph slot}

- vloží písmeno z {glyph slot} aktuálního fontu. Mění se v závislosti na zvoleném fontu, tím pádem je možné na výstupu dostat písmeno v různých fontech a dokonce v různých verzí stejného fontu.

\XeTeXcountglyphs{font}

- číslo počtů písmen v určitém fontu {font}.

\XeTeXglyphname{font}{glyph slot}

- rozšíří název písmena v {glyph slot} dáného {font}. Mění se v závislosti na zvoleném fontu, tím pádem je možné na výstupu dostat písmeno v různých fontech a dokonce v různých verzích stejného fontu.

\XeTeXglyphindex"{glyph name}"{space}

- vloží k písmenu místo odpovídající (fontu specifickému) {glyph name} v aktuálně používaném fontu. Podporuje pouze TrueType písma, nebo TrueType-based a OpenType písma.

\XeTeXcharglyph{char code}

- vloží k původnímu písmenu číslo znaku {char code} v aktuální podobě nebo 0 jestli znak není dostupný v daném fontu.

\XeTeXglyphbounds{edge}{glyph slot}

- rozšiřuje k směru, který odpovídá jedné hranici písmena, kde {edge} je číslo od 1 do 4 udávající left/top/right/bottom hranici a {glyph slot}je číselná hodnota udávající index písmena ve stávajícím fontu (dostupné pouze pro non-TFM fonty).

\XeTeXuseglyphmetrics

- specifikuje zda-li výška a hloubka znaku jsou brány v úvahu při sazbě (1). V případě (< 1) je nastavena jednotná výška a hloubka pro celou abecedu, která je používána.

```
\XeTeXuseglyphmetrics=0\fbox{a}\fbox{A}\fbox{j}\fbox{J}
\XeTeXuseglyphmetrics=1\fbox{a}\fbox{A}\fbox{j}\fbox{J}
```

Příklad 11: XeTeXuseglyphmetrics

Výstup tohoto kódu v XƎTEXu má následující podobu:
a A j J vs. a A j J

5.2 Příkazy pro OpenType fonty

\XeTeXOTcountscripts{font}

- rozšíření k číslu skriptu v {font}.

\XeTeXOTscripttag{font}{integer , n}

- rozšíření n-tého skript tagu v {font}.

```
\XeTeXOTcountlanguages{font}{script tag}
```

- rozšíření počtu jazyků daného skriptu v {font}.

```
\XeTeXOTlanguagetag{font}{script tag}{integer, n}
```

- rozšíření n-tého tagu jazyka daného skriptu v {font}.

```
\XeTeXOTcountfeatures{font}{script tag}{language tag}
```

- rozšíření počtu features v jazyce daného skriptu v {font}.

```
\XeTeXOTfeatureta{font}{script tag}{language tag}{integer, n}
```

- rozšíření n-tého feature tagu v jazyce daného skriptu v {font}.

5.3 Příkazy pro AAT a Graphite fonty

5.3.1 Features

```
\XeTeXcountfeatures{font}
```

- rozšíření počtu feature daného {font}.

```
\XeTeXfeaturecode{font}{integer, n}
```

- rozšíření feature kódu pro n-tou feature daného {font}.

```
\XeTeXfeaturename{font}{feature code}
```

- rozšíření k jménu odpovídající {feature code} daného {font}.

```
\XeTeXisexclusivefeature{font}{feature code}
```

- rozšíření k číslu větší než 0 za předpokladu, že feature fontu je jedinečná.

```
\XeTeXfindfeaturebyname{font}{feature name}
```

- tento příklad poskytuje metodu dotazu jestli {feature name} odpovídá feature obsažené v {font}.

5.3.2 Feature selektory

```
\XeTeXcountselector{font}
```

- rozšíření počtu selektorů v feature daného {font}.

```
\XeTeXselectorcode{font}{feature code}{integer , n}
```

- rozšíření selektor kódu pro n-tý selektor ve feature daného {font}.

```
\XeTeXselectorname{font}{feature code}{selector code}
```

- rozšíření k jménu odpovídající {selector code} ve feature daného {font}.

```
\XeTeXisdefaultselector{font}{feature code}{selector code}
```

- rozšíření k číslu větší než 0 za předpokladu, že feature selektor daného fontu je zapnut.

```
\XeTeXfindselectorbyname{font}{feature name}{selector name}
```

- tento příklad poskytuje metodu dotazu jestli {selecotr name} odpovídá selektoru obsažené ve specifickém {feature name} daného {font}.

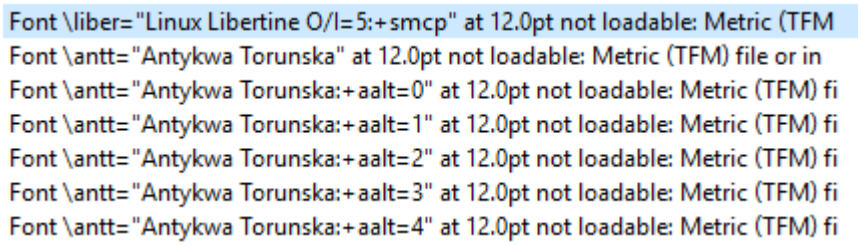
6 Testování XeTEXu

Samotné testování XETEXu jsem rozdělil do dvou částí. V první části testuji XETEX jako samotný, co je zapotřebí k tomu, aby uživatel byl schopen bez jakýchkoliv problémů použít XETEX, aby věděl jaké rozšiřující balíčky se musí použít k optimálnímu formátování a změny stylu svého dokumentu, ale také jaké omezení a nedostatky jsou s XETEXem spojeny.

Ve druhé části testování jsem se zaměřil na specifické editory, ve kterých je možné s XETEXu zpracovávat dokument. Protože editor sám o sobě na XETEX velký vliv nemá, hlavním cílem bylo otestovat jakým způsobem jednotlivé editory k XETEXu přistupují, co uživatel musí udělat, aby XETEX mohl použít a jakým způsobem ulehčí uživateli práci s XETEXem.

6.1 Výběr distribuce TEXu

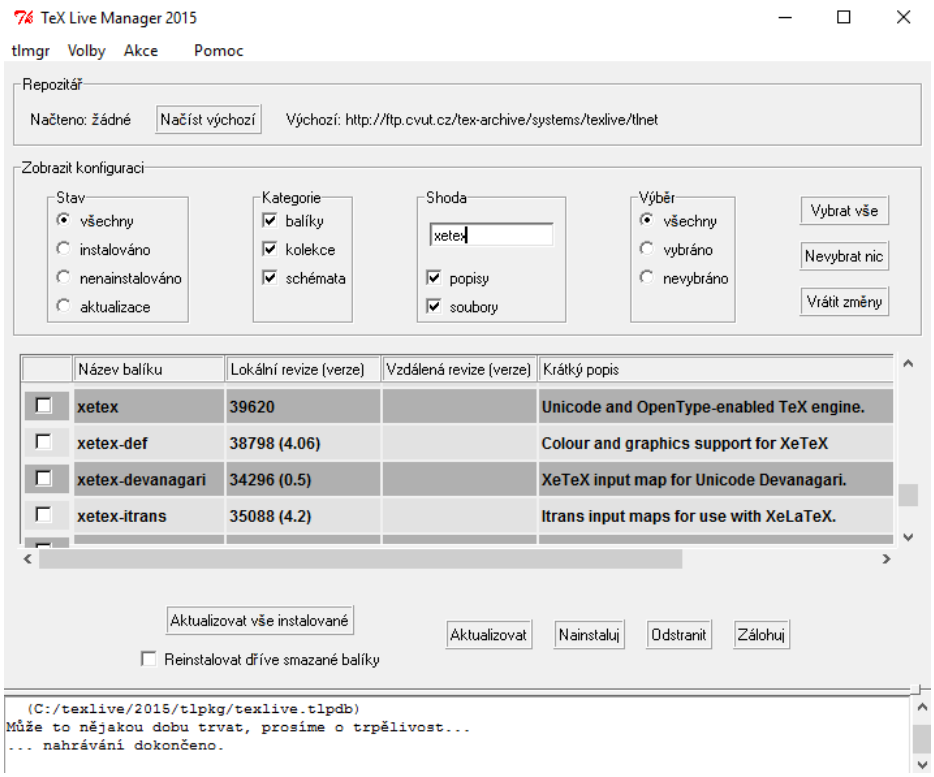
Základním stavebním kamenem pro práci s TEXem, LATEXem nebo XETEXem je distribuce TEXu. Tyto distribuce umožňují používanému enginu využívat nespočetnou řadu rozšíření, které jsou zapotřebí k typografii a sazbě dokumentů. V TEX komunitě jsou dnes nejpopulárnější a nejpoužívanější dvě distribuce TEXu a to jsou distribuce MiKTEX a TEXlive. Při testování XETEXu jsem ze začátku využíval distribuci MiKTEX, který byl pro svou jednoduchost, přehlednost a rychlost velice kompetentním prvkem při práci s XETEXem. Bohužel problém nastává v momentě, kdy jsem potřeboval použít balíček nebo libovolnou znakovou sadu, kterou standardní instalace MiKTEXu neinstalovala a v některých případech ani neobsahovala. Například jsem chtěl použít font Linux Libertine, který nebyl dostupný v rámci obvyklé nabídky operačního systému a MiKTEXu. V tomto případě bych musel složitě doinstalovávat jeden font do operačního systému nebo importovat do MiKTEXu ručně.



Obrázek 1: Chyba načtení písma

Kvůli tomuto problému jsem hledal co nejlepší a nejefektivnější řešení, díky kterému bych nemusel doinstalovávat jeden font individuálně, nebo hledat sadu fontů podle písma, které bych chtěl využít. Narazil jsem na alternativu distribuce MiKTEX, která se nazývá TEXlive. TEXlive kompletně nahrazuje funkcionalitu MiKTEXu, jeho největší výhodou je v první řadě to, že obsahuje všechny dostupné balíčky pro TEX, které jsou aktualizované na nejnovější dostupnou verzi, pokud by vyšla nová verze po nainstalovaní TEXlive, tak aktualizace jednotlivých balíčků je v tomto případě samozřejmostí. Dále také obsahuje sady OpenType, TrueType a AAT fontů, které jsou dostupné zdarma na internetu. Tím, že TEXlive obsahuje obrovské množství různých balíčku, není potřeba jejich žádná externí instalace. Na rozdíl od MiKTEXu, kde je jsou nainstalovány jen základní balíčky, z toho důvodu se velmi často stává, že uživatel musí během práce na dokumentu dohledávat potřebné balíčky a doinstalovávat je, což může být pro celou řadu uživatelů docela nepříjemné. Jedinou nevýhodu, u které jsem se setkal při používání TEXlive byla velice zdlouhavá instalace, která trvala několik hodin, což je docela pochopitelné vzhledem objemu dat, která jsou jeho součástí.

TEXlive také nabízí velice přehlednou databázi pro jednotlivé balíčky, která je velice přínosná pokud uživatel nezná přesný název balíčku, který by vyřešil jeho problém, tak stačí napsat do vyhledávání jakou problematikou by se měl požadovaný balíček zabývat a TEXlive vyhledá takové balíčky, které požadavku vyhovují, nebo jsou v některém směru podobné. Takto uživatel může narazit na celou řadu balíčků, na které by nenarazil kvůli neznalosti názvu.



Obrázek 2: TEXlive databáze

6.2 Základní balíčky

Většina XETEX příkazů je podporována samotným XETEXem a tudíž není zapotřebí importu jakéhokoliv balíčku kromě XETEXu. Při tvorbě dokumentu v XETEXu jsem narazil na několik balíčků, které uživateli ulehčí práci v XETEXu.

6.2.1 xunicode

Xunicode balíček zpřístupňuje pro dokument latinské akcenty jako jsou háčky nebo čárky a celou řadu dalších unicode znaků. Díky tomuto balíčku je TEX schopen překládat UTF-8 kódování a překládat unicode a OpenType fonty.

6.2.2 xltxtra

Při použití balíčku xltxtra se automaticky načtou balíčky fontspec, realscript a metalogo. Díky tomuto balíčku je možno používat doplňující XeTeX příkazy jako `\textsuperscript` a `\textsubscript`. Tento balíček je také zapotřebí k zobrazení XeTeX specifických log v dokumentu.

6.2.3 xevlna

Jelikož XeTeX automaticky neřeší problém jednoznakových slov na koncích řádků, tak tento balíček je důležitou součástí tvoření dokumentu v XeTeXu z typografického hlediska. Díky balíčku xevlna se za jednoznaková slova na konci řádku vkládá pevná mezera. Důležitý poznatkem je, že čeština má své specifické pravidla pro zalamování jednoznakových slov (například o proti anglické typografii), proto je nutné použít pravě balíček, který je určený pro český jazyk. Pokud nastane situace, kdy uživatel chce ve svém dokumentu, který je psán v češtině, psát část v jiném jazyce, nebo si nepřeje, aby se pevné mezery vkládaly do dokumentu, jednoduše toho docílí za použití příkazů `\xevlnaDisabled` a `\xevlnaEnabled`. Díky těmto příkazům si uživatel může vyhradit části dokumentu, kde xevlna použita nebude.

Pro vkládání pevných mezer je několik alternativních způsobů, které řeší tento typografický problém. Jednou z možností je ruční vkládání pevných mezer za jednotlivá jednoznaková slova (například pomocí příkazu `\nolinebreak`), avšak toto řešení není zcela ideální a praktické. Především v dokumentech, které jsou velice obsáhlé, se může snadno udělat chyba. Pokud by chtěl uživatel svůj text kopírovat a použít někde mimo XeTeX, tak by musel mazat všechny příkazy ručně, což je velice nepraktické.

Další alternativou může být použití podobného LaTeX balíčku encxvlna, který ve skutečnosti řeší problém jednoznakových slov na koncích řádků stejně jak balíček xevlna. Nevýhodou tohoto balíčku je velice složitý import a zprovoznění, u kterého je nutno zasahovat do interních souborů balíčku a ručně jej nastavit. V tomto je právě veliká výhoda XeTeXu, ve kterém lze využít balíček xevlna, který stačí pouze importovat v dokumentu a ostatní práci udělá za vás.

6.2.4 Balíček pro podporu jazyků

V TEXu je běžné používat balíček babel, který obsahuje vícejazyčnou podporu pro TEX. Nevýhodou balíčku babel je, že ačkoliv je velice dostačující pro práci s LATEXem, tak pokud by se balíček babel používal s XETEXem, u celé řady jazykových sad, které obsahují speciální znaky, babel nemá podporu. Proto byl vyvinut speciální balíček polygosia, který byl navrhnut přímo pro spolupráci s XETEXem a obsahuje kompletní podporu pro ty znaky, které babel nepodporuje. Pokud uživatel bude psát dokument v XETEXu a bude používat balíček babel, tak pro takové jazyky jako je ruština, arabština, hebrejština, nebo cyrilika nemá podporu. Polygosia je přímo vyvinuta pro tyto situace a při práci v XETEXu je výhodnější používat balíček polygosia, než babel.

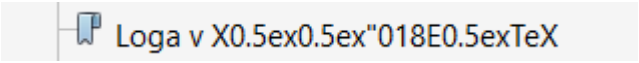
6.3 Nevýhody XeTEXu

6.3.1 XeTEX logo

Jednou ze základní charakteristiky vytváření dokumentů v TEXu jsou specifická loga jako TEX, nebo LATEX. Je tedy jen přirozené, že i XETEX má svoji verzi loga. V případě importu XETEX loga se vyskytuje několik problémů. XETEX logo bohužel není automaticky generováno při překladu jako je tomu při použití log TEX, nebo LATEX. Jedním způsobem jak zobrazit XETEX logo je pomocí příkazu `\XeTeX`, pro který je nutné použít balíček metalogo, který se součásti balíčku xltxtra.6.2.2 Toto řešení je samozřejmě velice nepraktické pokud máte rozsáhlejší dokument, ve kterém chcete zobrazit logo XETEX ve více případech. Bohužel je to momentálně nejjednodušší řešení jak XETEX logo zobrazit v dokumentu.

Alternativním řešením je možnost změnit styl a zarovnání jednotlivých písmen pomocí příkazů `\setlogokern` a `\setlogodrop`, které jsou součástí balíčku metalogo. Tento způsob je velice složitý a doporučil bych ho pouze v případě, když budete používat písmo takové, které má velice odlišné rysy a pozice jednotlivých znaků neodpovídá automaticky generovanému logu při použití přikazu `\XeTeX`.

Další problém nastává při překladu do pdf formátu. V samotném textu se XeTeX logo zobrazí bez jakéhokoliv problému, komplikace nastává tehdy, když potřebujete XeTeX logo zobrazit v nadpise, tak v osnově v pdf souboru se XeTeX logo nezobrazuje jako běžný text, ale jako interní kód příkazu \XeTeX. Na rozdíl od log TeXu, nebo LaTeXu, které v případě použití v nadpisu jsou automaticky zobrazována jako obyčejný text bez stylu.



Obrázek 3: Osnova s XeTeX logem v pdf

6.3.2 Problémy s překladem češtiny

Při překladu některých českých znaků má XeTeX problém tyto znaky číst, pokud využívá nějaké balíčky, které upravují nějakým způsobem písmo. Například balíček ae, který slouží k vyhlazení písma a lepší čitelnosti v XeTeXu nepřeloží háčky a ve výsledném dokumentu dotyčný znak vynechá.

7 Testování editorů s podporou XeTEXu

Dnes je dostupná celá řada TEX, nebo LATEX editorů, která umožňuje překlad daného dokumentu pomocí XETEXu. Testováním se snažím zjistit, do jaké míry jsou tyhle editory schopny rozšíření XETEX podporovat. Implementace XETEXu se pochopitelně liší podle používaného editoru, u některých editorů je používání XETEXu poměrně jednoduchou záležitostí, většina problémů může nastat v momentě, kdy chce uživatel používat pro tvorbu dokumentů nástroje, rozšíření či funkce, které nejsou vzájemně kompatibilní mezi sebou nebo se samotným XETEXem. V tomto případě je nutné najít nejideálnější řešení, které nijak neovlivní chod jednotlivých rozšíření, které uživatel používá při tvorbě dokumentů.

7.1 XeTEX v LYXu

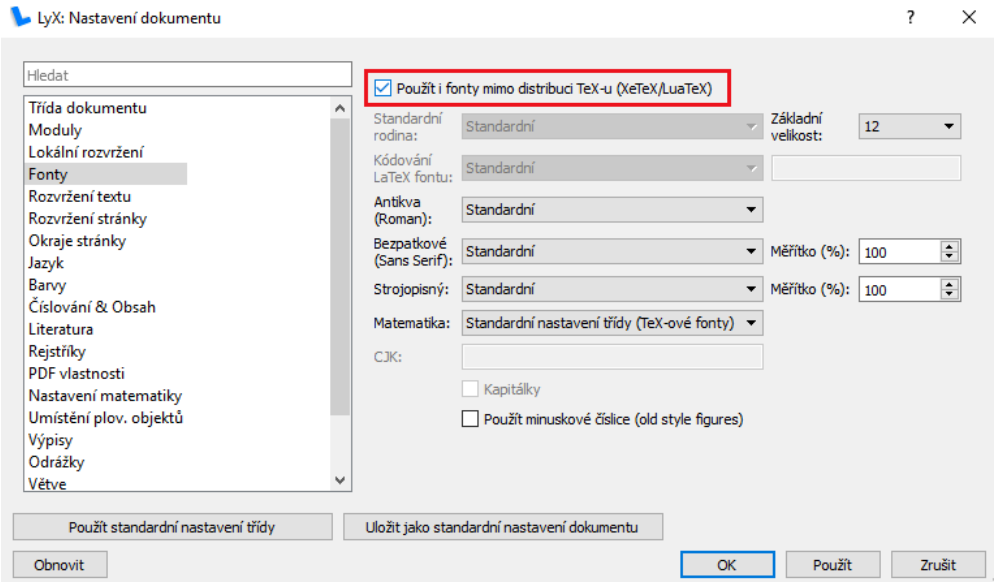
Jako první editor, který jsem zvolil pro testování XETEXu je TEX WISIWIG editor LYX. V tomto případě jsem se rozhodl samotnou bakalářskou práci napsat v XETEXu pomocí editoru LYX, jelikož jsem měl tuto jedinečnou možnost, byla by velká škoda ji nevyužít. Díky tomu, že je moje bakalářská práce psána XETEXem v LYXu, tak je samotný proces tvorby bakalářské práce výborným způsobem jak důkladně otestovat funkcionalitu LYXu společně s jeho možnostmi v oblasti práce s XETEXem. Na druhou stranu problém nastává v momentě, kdy narazím na určitý problém, který se v LYXu XETEXem nedá vyřešit, nebo jeho řešení je nedostačující. V tomto případě samotné přeložení dokumentu do výsledného formátu je nemožné a celý dokument s bakalářskou prací je nepoužitelný.

7.1.1 Nastavení LYXu pro XeTEX

V první řadě nejdůležitějším krokem pro úspěšné nastavení XETEXu je mít stáhnutou a nainstalovanou samotnou distribuci sázecího systému, která XETEX obsahuje. V tomto případě MiKTEX, nebo TEXlive.

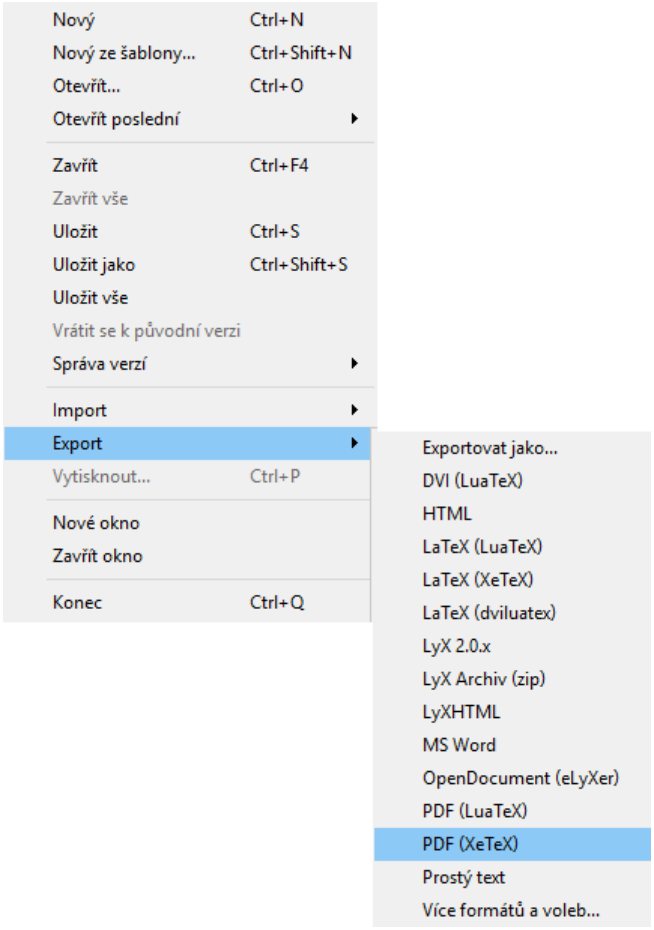
Implementace XETEXu se v LYXu liší na základě toho jakou distribuci TEXu využíváte. Když jsem používal jako sázecí systém MiKTEX, tak pro

překlad a export mého dokumentu do formátu pdf XeTeX nebyla možná kvůli chybějící nabídce pro překlad do požadujícího formátu. Proto, abych byl měl možnost překládání dokumentů do XeTeXu s MiKTeXem je potřeba povolit LyXu čerpat ze sad fontů, které se mohou používat i mimo distribuci TeXu, to jsou právě ty sady fontů, pro jejichž běh je nutnost využití XeTeXu.



Obrázek 4: Výběr non-TeX fontů v LyXu

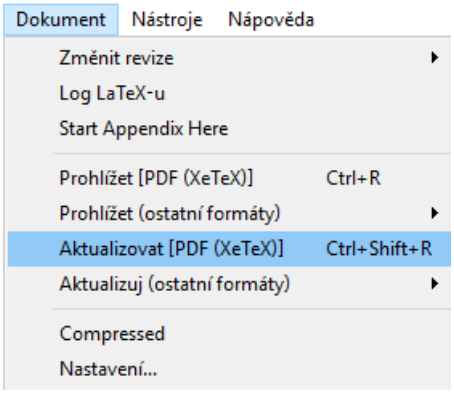
Potom co se zaškrtne tuto možnost, LyX nám automaticky nabídne převod do formátů, které se překládají XeTeXem. V tento moment je už výsledný dokument překládán XeTeXem a je možné přímo používat specifické příkazy a sady fontů pro XeTeX . Stále ale musíme brát na vědomí, že celou řada příkazů a fontů XeTeX nemůže přeložit sám, ale je zapotřebí specifického balíčku, který umožní XeTeXu přeložit náš dokument do výsledné podoby.



Obrázek 5: Export XeTEXu v LyXu

7.1.2 Aktualizace XeTEXu

LyX jako jeden z mála TEX editorů nabízí možnost přímé aktualizace XƎTEXu v jeho editoru. U ostatních editorů je třeba aktualizovat XƎTEX pomocí používané distribuce TEXu, která nám aktualizuje XƎTEX na nejnovější dostupnou verzi. LyX má v sobě tuto funkci nastavenou implicitně, a na základě toho přes jaké prostředí překládáme náš dokument, tak nám umožní aktualizaci přímo v editoru během naší práce. Tato funkce je velice praktická a uživatel nemusí opouštět prostředí LyX kvůli aktualizacím proXƎTEX.



Obrázek 6: Aktualizace XeTEXu v LYXu

7.1.3 Vkládání XeTEX kódu

LYX jako WYSIWIG editor nepožaduje použití jakýchkoliv příkazů pro tvorbu dokumentů, které jsou nahrazeny funkcemi rozhraní LYXu. Pro XETEX příkazy bohužel nejsou žádné specifické funkce, které by byli v editoru pro usnadnění práce. Proto jediným možným způsobem jak použit příkazy XETEXu je přes funkci vkládání TEX kódu. Problém může nastat v momentě, kdy máme obsahově rozsáhlý TEX kód, který vkládáme do LYXu. Tento kód není žádným způsobem formátovaný a jednotlivé příkazy od sebe nejsou odlišné co se stylu písma a barvy týče. Tohle je pro uživatele velice nepříjemné prostředí pro psaní kódu a v tomto případě bych jednoznačně doporučil psát XETEX kód v editoru, který tyto nedostatky neobsahuje.

Pro import balíčku a kódu, který bude mít vliv na celý dokument je v LYXu možnost úpravy preambule, ve které jsou automaticky použity příkazy uvnitř preambule.

7.2 XeTEX v Texmakeru

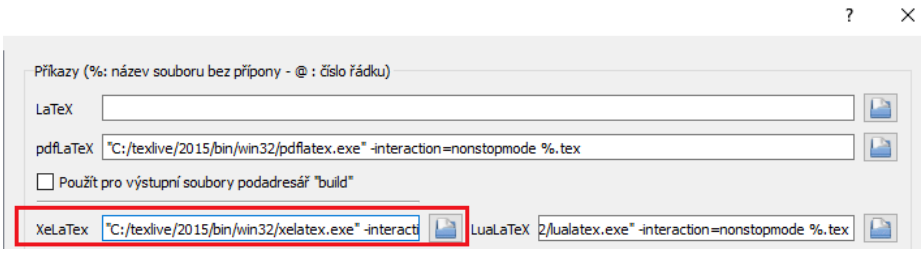
Texmaker je editor, který pracuje s makry LATEXu v programu TEX. Protože je to LATEX editor, tak klasická verze XETEXu na něm není možná použít. Aby bylo možné využít XETEX v Texmakeru, je nutné mít jeho LATEX verzi XELATEX, která je co se týče funkčnosti zcela totožná s XETEXem. Z praktického hledika je XELATEX jen XETEX přizpůsobený pro práci v LATEXu. Jediný

rozdíl tedy může být v syntaxi jednotlivých příkazů, zatímco při psaní kódu v XeTeXu je nutné psát příkazy plain-TeX formátu, tak v XeLaTeXu může být použit jak LaTeX tak plain-TeX. Například příkaz pro nastavení písma v dokumentu v XeTeXu je `\font,` zatímco v XeLaTeXu je to `\setmainfont`.

7.2.1 Nastavení TeXmakeru pro XeLaTeX

Podobně jak tomu bylo v případě LyXu, tak základním prvkem pro práci s Texmakerem a XeLaTeXem je mít nainstalovanou distribuci TeXu, která bude obsahovat samotný XeTeX (v tomto případě XeLaTeX). Je tedy velice důležité zajistit, aby byl v dané distribuci stáhnutý a nainstalovaný XeLaTeX. V tomto případě si editor vystačí pouze s XeLaTeXem a samotný XeTeX není zapotřebí instalovat. Výběr distribuce, se kterou budete v Texmakeru pracovat je zcela na tom, čemu dává samotný uživatel přednost. Texmaker pracuje bez jakýchkoliv problému, jak s MiKTeXem tak TeXlive.

Pro spolupráci XeLaTeXu s Texmakerem je nutné ukázat Texmakeru, kde se nachází. Texmaker totiž není schopen bez určité cesty vyhledat v počítači adresář, kde je XeLaTeX nainstalovaný. Proto je třeba zadat v nastavení Texmakeru přesnou cestu, kde se nachází spouštěcí soubor XeLaTeXu.



Obrázek 7: Nastavení XeLaTeXu v Texmakeru

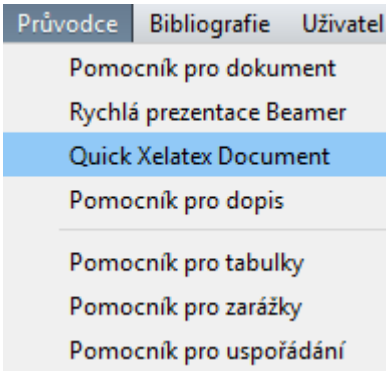
7.2.2 Aktualizace XeLaTeXu

Jako tomu bylo v LyXu, kde je možnost aktualizace XeTeXu přímo uvnitř editoru, tak Texmaker tuto možnost bohužel nenabízí. Pokud tedy uživatel chce aktualizovat XeLaTeX na nejnovější verzi, musí prostřednictvím distribuce TeXu, nebo ručním stáhnutím a importem požadované verze XeLaTeXu. Tato možnost aktualizace se může zdát nedůležitá, externí aktualizace mimo

editor není natolik složitá, ale možnost pro uživatele aktualizovat XeLaTeX přímo v editoru, aniž by musel opouštět rozhraní, ve kterém právě pracuje, je jednoznačně pohodlnější a praktičtější.

7.2.3 Vkládání XeLaTeX kódu

Na rozdíl od WISIWIG LyXu, Texmaker je zcela běžný editor, do kterého se příkazy kódu vkládají ručně. Texmaker jako jeden z mála editorů nabízí pro XeLaTeX nějakou formu podpory strukturováním dokumentu. V Texmakeru je možné vygenerovat základní kostru kódu pro XeLaTeX, která obsahuje některé ze základních prvků, které jsou nutností pro překlad dokumentů v XeLaTeXu.



Obrázek 8: Generování XeLaTeX syntaxe

V této kostře jsou vygenerovány balíčky pro XeLaTeX jako xunicode, xltxtra, fontspec, nebo polygossia. Import balíčku fontspec není tolik důležitý, protože tento balíček už je součástí balíčku xltxtra, oba tyto balíčky mohou být bez jakéhokoliv problému s kompatibilitou importovány společně. Pokud budete psát dokument v češtině, je ještě nutné doplnit kostru o balíček xevlna pro typografickou korektnost.

Jednou z velice praktických funkcí Texmakeru je vnitřní překládání dokumentu, tudíž vzhled našeho výsledného dokumentu můžeme prohlížet přímo uvnitř Texmakeru namísto v externím programu mimo editor.

8 Porovnání XeTEXu s LuaTEXem

XeTEX a LuaTEX jsou dvě velice podobné technologie. Obě dvě tyto technologie se zabývají překladem unicodových formátů písma v TEXu. I když jsou obě tyto technologie zaměřené na stejnou problematiku, každá má svůj způsob přístupu k sázení typografie v TEXu. Protože se zabývají stejnými problémy, je tedy velice zajímavé jejich vzájemné porovnání, jakým způsobem řeší tuto problematiku, jaké mají výhody nebo čím se od sebe liší.

8.1 Rozdíl přístupu k TEXu

XeTEX přistupuje k TEXu pomocí specifických systémových knihoven, ze kterých velice jednoduchým způsobem načítá systémové fonty, případně další UTF-8 sady. Tudíž pro koncového uživatele je poměrně snadné použít XeTEX, protože je navrhnut tak, aby byl co nejvíce přizpůsoben práci s operačním systémem. Na druhou stranu tento přístup může být nevýhodný pro uživatele, kteří chtějí mít absolutní kontrolu nad programem a XeTEX v tomto směru není až tolik flexibilní.

I když LuaTEX ve výsledku nabízí podobné řešení pro podporu UTF-8 formátů, jeho přístup se o něco liší od přístupu XeTEXu. Hlavním rozdílem je to, že LuaTEX pracuje na úrovni skriptovacích jazyků a sám využívá vlastní skriptovací jazyk Lua. LuaTEX je jedinečný v tom, že má přístup přímo do interních příkazů, které TEX obsahuje. Díky tomu má LuaTEX velice možností v tomto směru. Pokud uživatel LuaTEXu nalezne nějakou problematiku, která není LuaTEXem podporována, může upravit některé příkazy pomocí skriptů tak, aby pracovali v prostředí LuaTEX.. Nevýhodou je jednoznačně to, že funkce psaní vlastních skriptů vyžaduje znalosti programování skriptů v Lua jazyku. Tudíž pro běžného uživatele, který neovládá jazyk Lua, je tato funkce zbytečná. Naopak pro uživatele, který je schopný tyto skripty v jazyce Lua vytvářet se otevírá celá řada možností, jak LuaTEX využít.

8.2 Překlad fontů

Obě dvě tyto technologie jsou výjimečné tím, že jsou schopny překládat v TEXu znaky s kódováním UTF-8 a fonty typu OpenType, TrueType, nebo Graphite. Proto jsem se rozhodl otestovat a porovnat způsob jakým XETEX a LuaTEX tyto fonty překládají a poukázat na případné rozdíly nebo nedostatky mezi těmito dvěma technologiemi. Pro testování jsem vybral fonty takové, které nejsou ve stejném typu standardu fontu, čili jsem testoval pro standard OpenType, TrueType a Graphite vždy jeden font, aby testování bylo co nejpřesnější a nejdůkladnější. Fonty, které jsem vybral jsou Linux Libertine pro OpenType, CMU Serif pro TrueType a Charis SIL pro Graphite.

8.2.1 Testování velkých a malých písmen

Jako první jsem otestoval, jak si XETEX s LuaTEXem poradí s překladem malých a velkých písmen.

Překladu TrueType fontu CMU Serif v XETEXu a LuaTEXu vypadá následovně:

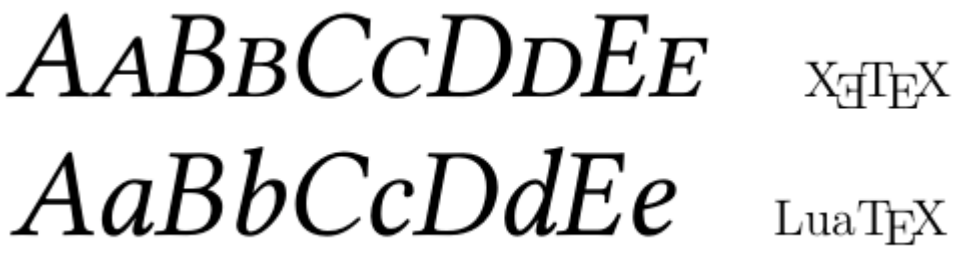
AaBbCcDdEe XETEX

AaBbCcDdEe LuaTEX

Obrázek 9: Překlad malých a velkých písmen pomocí TrueType fontu

Jak lze vidět na obrázku, tak překlad TrueType fontu v XETEXu a LuaTEXu proběhne bez jakýchkoliv potíží a výsledek je zcela totožný.

Překlad OpenType fontu Linux Libertine v XETEXu a LuaTEXu vypadá následovně:



Obrázek 10: Překlad malých a velkých písmen pomocí OpenType fontu

Při překladu fontu standardu OpenType nastávají první rozdíly a problémy. Zatímco velká písmena jsou v obou případech přeložena totožně a bez problémů, tak v případě malých písmen je jejich zobrazení odlišné. LuaTEX při překladu malých písmen využije právě ty znaky, které jsou určeny pro zobrazení malých písmen v fontu Linux Libertine. XeTEX v tomto případě místo toho, aby použil malá písmena z sady písmen fontu, tak pouze změní velikost velkých písmen na menší a malá písmena vůbec nevyužije. Při použití příkazu `\font\small caps=true/false` je možné využit speciální font features, která nařídí XeTEXu změnu na malá písmena. Bohužel ani pomocí této varianty XeTEX tento OpenType font není schopen přeložit korektně. Je třeba dodat, že tento problém se nevyskytuje u všech OpenType fontů.

Překlad Graphite fontu Charis SIL vypadá následovně:

Obrázek 11: Překlad malých a velkých písmen pomocí Graphite fontu

Co se týče překladu fontu standardu Graphite, tak XeTEX i LuaTEX přeložili malá a velká písmena se stejným výsledkem a bez komplikací.

8.2.2 Testování diakritiky

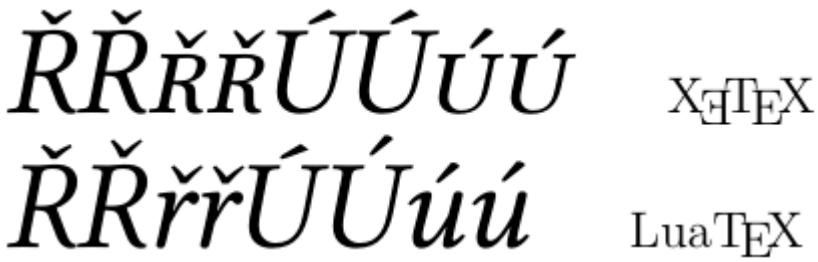
V další části testování jsem se zaměřil na překlad diakritiky. Diakritika je nedílnou a velice důležitou součástí českého jazyka, proto je třeba klást velký důraz na to, aby sázení znaků jako jsou háčky, nebo čárky bylo korektně překládáno. Při testování diakritiky u jednotlivých fontů jsem vždy vybral jedno malé písmeno a jedno velké písmeno, kde je háček, nebo čárka implicitně nastavena. Potom jsem to stejné písmeno přeložil ještě jednou, ale tentokrát jsem háček a čárku přidal pomocí příkazu `\char`, kde se nastaví znak z unicode sady.

Překlad TrueType fontu CMU Serif v XETEXu a LuaTEXu vypadá následovně:

Obrázek 12: Překlad diakritiky pomocí TrueType fontu

Jak lze vidět na obrázku, tak XETEX přeložil háčky i čárky korektně u malých i velkých písmen a u písmen, kde byla diakritika přidána pomocí příkazu. V případě LuaTEXu byli bez problému přeloženy všechny malá písmena a velká písmena, kde byl háček a čárka implicitně nastavena. Problém nastal v situaci, kde LuaTEX musel přidat háček a čárku ke znaku pomocí příkazu. LuaTEX sice přidal háček i čárku k písmenu, ale už neupravil jejich pozici pro velká písmena, proto se háčky a čárky překrývají se samotným písmenem. Jelikož LuaTEX přeložil háčky a čárky na úrovni malých písmen, tak jejich překlad je zcela v pořádku.

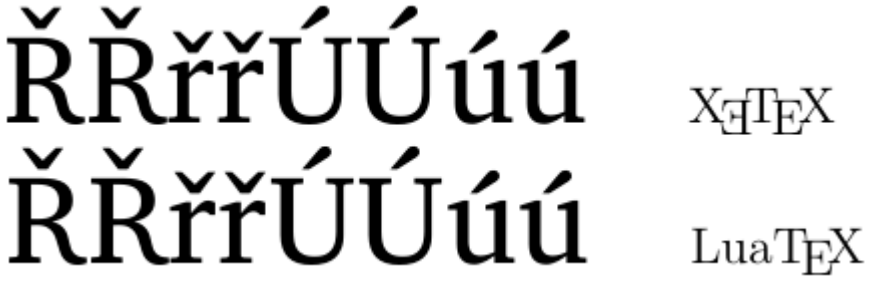
Překlad OpenType fontu Linux Libertine v XETEXu a LuaTEXu vypadá následovně:



Obrázek 13: Překlad diakritiky pomocí OpenType fontu

Při překladu OpenType fontu Linux Libertine v XeTeXu jsou háčky a čárky zcela v pořádku přeloženy, bohužel stále přetrvává problém s přeložením malých písmen z předchozí kapitoly8.2.1. LuaTeX při překladu fontu Linux Libertine tentokrát háčky i čárky správně přemístí i nad velká písmena. Jediná nesrovnalost se nachází ve tvaru háčku a čárky při překladu velkých písmen. Při přidaní diakritiky ručně k velkému písmenu je sice háček i čárka přeložena a posunuta na správné místo, ale je použita čárka s háčkem, která je určena pro malá písmena.

Překlad Graphite fontu Charis SIL vypadá následovně:



Obrázek 14: Překlad diakritiky pomocí Graphite fontu

XeTeX i LuaTeX nemají nejmenší problémy s překladem Graphite fontu Charis SIL. V případě LuaTeXu nejsou ani problémy v oblasti háčků a čárek, které jsou přidány příkazem a výsledek je totožný s XeTeXem.

8.2.3 Styly písma

Během testování stylů písma v XeTeXu a LuaTeXem jsem narazil na problém při překladu stylů písma jako je tučné písmo, kurzíva a podtržené písmo. XeTeX a LuaTeX jsou bez sebemenších problémů schopny tyto styly překládat, pokud není nastavena jiná velikost písma než základní. Problém nastává v momentě, kdy je velikost písma nastavena ručně u fontu (např. \font="Linux Libertine" at 30pt), potom XeTeX i LuaTeX při použití stylu kurzíva a tučné písmo změní velikost na původní. Jediné podtržené písmo si zachová tu velikost, která mu byla nově nastavena. V tomto případě nezáleží na tom, jaký druh nebo typ standardu písma je použitý, tento problém se vyskytuje ve všech případech při překládání v XeTeXu a LuaTeXu.

Na obrázku lze vidět příklad stylů se základní velikostí a s nově nastavenou velikostí.

Obrázek 15: Styly písma v XeTeXu a LuaTeXu

8.3 Překlad ligatur

Ligatury jsou znaky, které spojí sousední písmena v jeden znak. Při překladu těchto sousedních písmen může být automaticky vyvolána ligatura i v případě, kdy si ji uživatel nevyžaduje. Proto jsem se snažil otestovat, které ligatury se vyskytují při překladu v XeTeXu a LuaTeXu. Tato problematika se vztahuje jen pro některá písmena, nejčastěji se ligatury vyskytuji v kombinaci písmen f, nebo i.

Během testování jsem zjistil, že výskyt ligatur nezávisí jen na tom, jestli je použit XeTeX či LuaTeX, ale také na tom, v jakém fontu je používaný znak překládán. I když výběr fontu hraje zásadní roli v překladu ligatur, tak role XeTeXu s LuaTeXem je také velice důležitá. Během překladu může nastát

situace, kdy na jednom konkrétním fontu oba překladače ligatury zobrazí a na fontu jiném nezobrazí. Hlavní rozdíly mezi XeTeXem a LuaTeXem nastávají pochopitelně v případě, kdy jsou ligatury přeloženy odlišně i za použití fontu stejného.

Na prvním obrázku můžete vidět, že LuaTeX v tomto případě ligatury ignoruje a překládá písmena jako samostatné znaky.

Obrázek 16: Ligatury v LuaTeXu

Na druhém obrázku je použita stejná verze kódu a fontu s použitím překladu XeTeXu. Na první pohled je vidět jednoznačný rozdíl v překladu. XeTeX v tomto případě písmena i a f spojí dohromady a vytvoří ligatury. XeTeX spojí i více písmen za sebou do ligatury, pokud je to možné.

Obrázek 17: Ligatury v XeTeXu

8.3.1 Odstranění ligatur

Když se nám v dokumentu začnou objevovat ligatury, tak přirozenou otázkou je, jakým způsobem toto spojení písmen odstranit. Během testování jsem narazil na několik variant, které se snaží tento problém vyřešit, ale pouze jedna varianta byla spolehlivá ve všech případech. Pro odstranění ligatury je nutné mezi znaky, které se spojují vložit následující symbol bez uvozovek`''\/''`. Takže výsledné slovo bude mít následující podobu muf\/f\/in.

Toto řešení se může zdát nepraktické v obsáhlejších dokumentech, ale zatím je to jediné plně funkční řešení pro oba překladače.

9 Závěr

XeTeX je už dnes nedílnou součástí TeX technologií. Díky XeTeXu je dnes možné zpracovávat dokumenty ve znakových sadách a standardů písma, které dříve hledali podporu v TeXu marně. Typografie a sazba odborných dokumentů nebyla nikdy snazší a profesionálnější pro uživatele, kteří využívají jiné písmo než latinku. Tato technologie bohužel také disponuje problémy, především v překladu některých znaků a příkazů, které ještě XeTeXem nejsou plně podporovány.

V rámci bakalářské práce jsem splnil všechny stanovené cíle. Seznámili jsme se s TeXovým rozšířením XeTeX a jeho historií. Dále byly podrobně představeny příkazy, které přišli s XeTeXem jako novinka, ale také staré příkazy, které nabyli nových funkcionalit a vylepšeních díky XeTeXu. Byli jsme seznámení se speciálními formáty a standardy fontů, které jsou v dnešní době nejrozsáhlejší.

Bylo provedeno testování samotné technologie XeTeX, otestoval jsem implementaci XeTeXu a jeho základních balíčků do TeX editorů pod různými distribucemi. Porovnal jsem XeTeX s technologií LuaTeX, která se zabývá podobnou problematikou. V poslední řadě tato práce bylo vytvořena a přeložena pomocí XeTeXu a také je první svého druhu, která byla napsána na toto téma v českém jazyce.

XeTeX je technologie, která se stále aktualizuje a vyvíjí, a proto věřím, že do budoucna bude tato práce přínosem pro každého uživatele XeTeXu.

Použitá literatura a zdroje

[1] GOOSSENS, Michel. The XeTEX Companion. In: The LaTeX Graphics Companion [online]. CERN, 2011 [cit. 2016-03-14]. Dostupné z: http://xml.web.cern.ch/XML/lgc2/xetexmain.pdf

[2] ROBERTSON, Will. XeTEX ref: Reference documentation of XETEX [online]. CTAN, 2013 [cit. 2016-05-31]. Dostupné z: http://ftp.cvut.cz/tex-archive/info/xetexref/xetex-reference.pdf

[3] WALSH, Norman. NW. Nwalsh [online]. 1996 [cit. 2016-06-01]. Dostupné z: http://nwalsh.com/comp.fonts/FAQ/cf_15.htm

[4] Microsoft Typography: What is TrueType ? [online]. Microsoft, 2009 [cit. 2016-06-01]. Dostupné z: https://www.microsoft.com/en-us/Typography/WhatIsTrueType.aspx

[5] Microsoft Typography: TrueTypeHistory [online]. Microsoft, 1997 [cit. 2016-06-01]. Dostupné z: https://www.microsoft.com/typography/TrueTypeHistory.mspx

[6] Microsoft Typography: The True Type rasterizer [online]. Microsoft, 1997 [cit. 2016-06-01]. Dostupné z: https://www.microsoft.com/typography/TrueTypeRasterizer.mspx

[7] ROUSE, Margaret. What is TrueType? In: WhatIs [online]. 2005 [cit. 2016-06-01]. Dostupné z: http://whatis.techtarget.com/definition/TrueType

[8] PostScript. Fykos [online]. ©FYKOS [cit. 2016-06-01]. Dostupné z: http://fykos.cz/o-nas/co-je-ps

[9] OpenType. Typo [online]. o. s. TYPO, 2009 [cit. 2016-06-01]. Dostupné z: http://www.typo.cz/databaze/pismolijny-a-distributori/tvorba-a-editace-fontu/opentype/

[10] OpenType Cookbook. OpenTypeCookBook [online]. Type Supply LLC, 2009 [cit. 2016-06-01]. Dostupné z: `http://opentypecookbook.com/index.html`

[11] Adobe fonts: OpenType. Adobe [online]. Adobe Systems Incorporated, 2016 [cit. 2016-06-01]. Dostupné z: `http://www.adobe.com/products/type/opentype.html`

[12] LAURENS, Leurs. The history of fonts. In: PrePressure [online]. 2013 [cit. 2016-06-01]. Dostupné z: `http://www.prepressure.com/fonts/basics/history`

[13] Graphite. In: SIL NON-ROMAN SCRIPT INITIATIVE [online]. SIL International, 2012 [cit. 2016-06-01]. Dostupné z: `http://scripts.sil.org/cms/scripts/page.php?site_id=projects&item_id=graphite_about`

[14] What is Unicode? In: Computers & Writing Systems [online]. SIL International, 2007 [cit. 2016-06-01]. Dostupné z: `http://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&id=UTConvertQ1`

Seznam příkladů

Seznam obrázků

Seznam tabulek

Přílohy

1. CD s plným zněním bakalářské práce v PDF