Ročník 12

OBSAH

Petr Olšák: Úvodníček	109
Zápis z valného shromáždění Československého sdružení uživatelů $\text{\TeX}$ u	111
Stanovy Československého sdružení uživatelů $\text{\TeX}$ u	116
Zpráva o hospodaření k 31. 10. 2002	120
Bacho $\text{\TeX}$ 2003	121
Miroslav Hrad, Petr Sojka: Automatizace sazby a skenování formulářů	123
Vít Zýka: $\text{\TeX}$ a PDF	140
Tomáš Hála: Korektury a korektoři v 21. století	152
Jiří Rybička: Automatizovaná tvorba grafů v systémech na bázi $\text{\TeX}$ u	160
Petr Olšák: Makro OFS	172
Zdeněk Wagner: Využití XML a $\text{\LaTeX}$ u při sazbě odborných knih	188
Ján Buša: Použitie balíka <code>pdfscreen.sty</code>	212
Rejstřík	219
Petr Sojka: Kolofónek aneb závěrečná ediční poznámka	220

Zpravodaj Československého sdružení uživatelů $\text{\TeX}$ u je vydáván v tištěné podobě a distribuován zdarma členům sdružení. Po uplynutí dvanácti měsíců od tištěného vydání je poskytován v elektronické podobě (PDF) ve veřejně přístupném archívu dostupném přes <http://www.cstug.cz/>.

Své příspěvky do Zpravodaje můžete zasílat v elektronické podobě, nejlépe jako jeden archivní soubor (`.zip`, `.arj`, `.tar.gz`). Postupujte podle instrukcí, které najdete na stránce <http://bulletin.cstug.cz/>. Pokud nemáte přístup na Internet, můžete zaslat příspěvek na disketě na adresu:

Zdeněk Wagner
Vinohradská 114
130 00 Praha 3

Disketu formátujte nejlépe pro DOS, formáty Macintosh 1.44 MB a EXT2 jsou též přijatelné. Nezapomeňte přiložit všechny soubory, které dokument načítá (s výjimkou standardních součástí $\text{\LaTeX}$ u), zejména v případě, kdy vás nelze kontaktovat e-mailem.

ISSN 1211-6661 (tištěná verze)

ISSN 1213-8185 (online verze)

První sním této zimy někteří z nás zaznamenali v sobotu 9. listopadu 2002 v Seči u Chrudimi. Probíhal tam už třetí ročník semináře o Linuxu a $\text{T}_{\text{E}}\text{X}$ u – $\text{S}_{\text{L}}\text{T}$ 2002. Jako pořadatelé jsme byli na tomto semináři příjemně překvapeni rychlostí nabídek od přednášejících a rovněž dostatkem těchto nabídek, ačkoli jsme oznámili konání semináře (i potenciálním přednášejícím) na poslední chvíli. Na druhou stranu jsme byli trochu zaskočeni nízkou účastí na semináři – účastníků se sešla necelá stovka. Přitom zrovna hotel v Seči měl poměrně velké kapacity a disponoval i přednáškovým sálem pro 300 lidí a množstvím menších učeben. Ten největší sál jsme nakonec nevyužili.

Hotel v Seči na mnohé působil asi poněkud nezvykle. Byla to taková monumentální socialistická architektura skládající se z několika budov v rozsáhlém areálu. Na budovách se v současné době projevuje výrazné chátrání. Možná si někdo vzpomíná, že v době rozkvětu socialismu se mohli ti nejlepší pionýři zúčastnit mezinárodního pionýrského tábora v Seči. Tyto tábory se určitě dělaly právě v tomto hotelu. Na druhé straně cena za ubytování a stravování byla velmi příznivá, což asi souvisí s tím, že jsou schopni současně ubytovat několik konferencí najednou. Ani my jsme tam nebyli zdaleka sami. Když odmyslím organizaci stravování typu „školní jídelna“, pak mohu asi za mnohé prohlásit, že jídlo bylo vynikající. Jediným méně světlým bodem byl ranní párek hned v pátek. Tehdy jsem jim na výdejně položil otázku, zda si také ráno dávají na snídani párek a zda jim to nezpůsobuje problémy s metabolismem. Bylo nicméně patrné, že metabolismus obsluhy výdeje není pro určování jídelníčku pro hosty nejpodstatnější a že za ten párek asi může někdo jiný. Od té doby jsem jim ale později už nemohl nic vytknout, naopak bych je měl asi pochválit.

Cenu pro účastníky jsme vypočítali tak, aby nám těsně pokryla náklady s akcí spojené. Přitom to byla cena pro ty, kteří nejsou členy žádného ze dvou pořádajících spolků: Československého sdružení uživatelů $\text{T}_{\text{E}}\text{X}$ u a Českého sdružení uživatelů operačního systému Linux. Pro členy jsme nabídli slevy ve výši zhruba ročního členského příspěvku. Účastníků nárokových slevu přijela, podle očekávání, většina. Musíme tedy počítat se ztrátou, kterou si rozdělí podle dohody obě pořádající sdružení napůl. Ztráta, kterou v době psaní tohoto úvodníku ještě nemáme vyčíslenou, není něco, čím bychom byli na poslední chvíli zaskočeni. Byl to záměr dotovat z rezerv obou sdružení účast svých členů. Přitom počet členů CSTUGu a CZLUGu byl mezi účastníky $\text{S}_{\text{L}}\text{T}$ 2002 zhruba vyrovnán.

Podle ohlasů připojených některými účastníky k hodnocení přednášejících mohu směle prohlásit, že $\text{S}_{\text{L}}\text{T}$ 2003 byla úspěšná akce. Poděkování si zaslouží

všichni pořadatelé a dále hlavně přednášející, kteří kupodivu i v tak šibeničném termínu dokázali připravit příspěvek a dodat jej do sborníku. Vesměs všichni přednášející dodali příspěvek v editory požadovaném formátu – v $\text{T}_{\text{E}}\text{X}$ u. Vzhledem k tomu, že akce nebyla čistě $\text{T}_{\text{E}}\text{X}$ ová, jsem byl touto skutečností mile překvapen. Poděkovat bych chtěl za všechny i editorům sborníku, kteří přizpůsobili všechny došlé příspěvky jednotnému vzhledu. Na technické realizaci tohoto úkolu Petru Sojkovi pomáhal též David Antoš.

Většinu příspěvků z $\text{T}_{\text{E}}\text{X}$ ové části sborníku $\text{S}_{\text{L}}\text{T}$ můžete najít v tomto dvojčíslu našeho Zpravodaje.

Další významnou akcí koncem roku bylo valné shromáždění našeho sdružení, na kterém byla schválena změna sídla na adresu Technická 2, 166 27 Praha 6.

Před valným shromážděním proběhla již tradiční přednáška, tentokrát o českých písmech a o písmech Střešovické písmolijny zejména. Přednášku proslovil pan Otakar Karlas, který spolupracuje s majitelem písmolijny Františkem Štormem a který přednáší o písmu mimo jiné na Vysoké škole umělecko-průmyslové. Myslím, že kdo se přišel podívat, neprohloupil. Dozvěděli jsme se mnoho zajímavého z historie české typografie a názor na písma od fundovaného člověka doprovázený názornými ukázkami byl pohlazením na duši.

Po přestávce pokračovalo valné shromáždění našich členů. Kromě obvyklých bodů, jakými jsou zprávy o činnosti, hospodaření a revizi, byl na programu bod: změna stanov. Kromě již zmíněné změny sídla byly valným shromážděním odsouhlaseny změny týkající se vzniku a zániku členství. Podle nových stanov člen přestává být členem automaticky, když do stanoveného termínu nezaplatí členský příspěvek. Není tedy nutné nějaké vylučování a případné evidování neplatícího člena několik let. Podle nových stanov budeme mít konečně jasno, kdo je v daném okamžiku členem našeho sdružení a komu tedy je potřeba rozeslat například další číslo našeho Zpravodaje.

Valné shromáždění také potvrdilo výši členských příspěvků, tj. tyto příspěvky pro rok 2003 budou stejné jako v roce minulém. Řádní členové zaplatí 300 Kč, studenti a důchodci 200 Kč, členové ze Slovenska pak zaplatí stejnou částku, ale ve slovenské měně: 300 Sk nebo 200 Sk. Kolektivní členové mají členský příspěvek ve výši 1850 Kč a kolektivní členové – gymnázia zaplatí 650 Kč. Kolektivní členové dostávají 5 kusů Zpravodajů (místo jednoho) a také dostávají další materiál. Letos jsme jim například rozeslali sborníky ze $\text{S}_{\text{L}}\text{T}$ 2002 a dále jim každý rok rozesíláme kolekce CéDéček CTAN a $\text{T}_{\text{E}}\text{X}$ live.

Zápis z valného shromáždění společně s dalšími dokumenty včetně textu nových stanov našeho sdružení najdete rovněž v tomto čísle Zpravodaje.

21. 12. 2002

Zápis z valného shromáždění Československého sdružení uživatelů T_EXu

1. Schválení programu

navržený program byl schválen

2. Volba zapisovatele

zapisovatelem byl zvolen Jiří Kosek [pro: 38, proti: 0, zdrželo se: 1]

3. Volba sčítací a mandátové komise

do sčítací a mandátové komise byli zvoleni Jiří Matyáš a Josef Svoboda [pro: 39, proti: 0, zdrželo se: 1]

Zpráva sčítací a mandátové komise

Na tomto valném shromáždění je registrováno 24 řádných členů a 5 zástupců kolektivních členů. Podle čl. 10 odst. „Organizační struktura“ stanov platných do 30.11. 2002 má řádný člen 1 hlas a zástupce kolektivního člena 3 hlasy, dohromady tedy 39 hlasů. Podle stejného článku 30 minut po zahájení valného shromáždění rozhoduje prostá většina přítomných hlasů, tedy 20 hlasů. Veškeré hlasování o nových stanovách proběhne až po této lhůtě.

4. Zpráva o činnosti sdružení

zprávu přednesl předseda sdružení Petr Olšák:

Převedení administrativy do Prahy

- Paní Holovská se nabídla, že pro nás bude dělat administrativní práce
- Smlouva s účetní firmou ECQUO s. r. o, Spálená 21, pan Pavel Vlček za vedení účetnictví včetně ročních závěrek a daňových přiznání 1000 Kč/měsíc.

Udržování WWW, platba domény, administrátor pan Jiří Matyáš

Časopis Zpravodaj 1 a 2 / 2002

Aktivní účast na EuroT_EX 2002 v Polsku, hrazení výdajů účastníkům

Příprava včetně lisování CD T_EXlive v České republice, pouze jedno ze dvou CD.

Objednání a rozeslání CD-set CTAN 2002

Spoluorganizování S_LT 2002 s CZLUGem, ztráty si rozdělíme napůl.

- sazba sborníku pro S_LT 2002
- pomoc při organizaci

Plán

- Zhruba pokračovat v tomto duchu.
- Aktivní účast na EuroT_EX 2003 ve Francii.

Jiří Matyáš seznámil shromáždění s programem přednášek o T_EXu.

Petr Sojka seznámil shromáždění se zahraničními aktivitami C_STUGu.

Valné shromáždění vzalo na vědomí zprávu o činnosti [pro: 39, proti: 0, zdrželo se: 0].

5. Zpráva o hospodaření, vyjádření revizorů k účetnictví za rok 2001

Zprávu o hospodaření od účetní firmy prezentoval Petr Olšák.

Valné shromáždění vzalo na vědomí zprávu o hospodaření k 31.10. 2002 s následující připomínkou: Zpráva o hospodaření za celý rok 2001 bude zveřejněna ve Zpravodaji [pro: 39, proti: 0, zdrželo se: 0].

6. Zpráva revizora, revize účetnictví za rok 2001

D. Antoš přečetl Zprávu revizora účetnictví T. Hály za rok 2001.

Valné shromáždění bere na vědomí revizní zprávu [pro: 39, proti: 0, zdrželo se: 0].

7. Volby revizorů účetnictví

Valné shromáždění schválilo jako dalšího revizora sdružení Pavla Sekaninu [pro: 38, proti: 0, zdrželo se: 1].

8. Diskuse ke změně stanov a změna stanov

Číslované návrhy změn byly prodiskutovány na elektronické diskusní skupině, do které mohl přispívat každý člen sdružení. Přítomní členové a zástupci kolektivních členů dostali po jednom výtisku těchto návrhů změn a podle něj hlasovali.

Návrhy 1 (číslování hlavních článků) [pro: 39, proti: 0, zdrželo se: 0],

2 (čl. 2.2: změna sídla na Technická 2, 166 27, Praha 6) [pro: 39, proti: 0, zdrželo se: 0],

3 (čl. 7.3) [pro: 39, proti: 0, zdrželo se: 0],

4 (čl. 3.2) [pro: 39, proti: 0, zdrželo se: 0],

5 (čl. 4.4) [pro: 38, proti: 0, zdrželo se: 1],

6 (čl. 4.5) [pro: 39, proti: 0, zdrželo se: 0],

7 (čl. 4.6) [pro: 35, proti: 0, zdrželo se: 4],

protinávrh 8 (čl. 4.8: výši členských příspěvků a termín zaplacení schvaluje valné shromáždění) [pro: 35, proti: 1, zdrželo se: 0],

9a (čl. 5.2: vyhodit „s přesahem nejvýše dvou měsíců“) [pro: 34, proti: 0, zdrželo se: 3],

10 (čl. 5.6) [pro: 32, proti: 0, zdrželo se: 1],

11 (čl. 4.2: vyhodit „s členstvím je neslučitelné. . . “ [pro: 22, proti: 10, zdrželo se: 1]

byly přijaty valným shromážděním.

Návrh 8 byl zamítnut valným shromážděním.

Valné shromáždění přijalo nové znění článku [pro: 39, proti: 0, zdrželo se: 0]

Čl. 4.8. Výši členských příspěvků i termín zaplacení příspěvku pro následující kalendářní rok schvaluje valné shromáždění. Pokud výbor úda je o výši členských příspěvků a termínu zaplacení neprojedná, tyto údaje se pro následující rok nemění.

Valné shromáždění odhlasovalo zrušení článku 6.3 [pro: 37, proti: 0, zdrželo se: 0].

Návrh 9 byl přijat v následujícím znění [pro: 34, proti: 0, zdrželo se: 3]:

Čl. 5.2. Text „Výbor musí svolávat . . . do dalšího roku.“ přesunout do dalšího odstavce, nyní označovaného jako 2a. V definitivní verzi pak přechíslovat všechny odstavce níže o jeden.

Čl. 5.2a. Výbor musí svolávat valné shromáždění alespoň jednou ročně v příslušném kalendářním roce. Pozvánky na valné shromáždění musí být rozeslány podle požadavků jednotlivých členů písemně nebo elektronicky nejpozději 4 týdny před termínem valného shromáždění. Pokud je na programu valného shromáždění závažný bod k projednání: změna stanov, volba výboru nebo likvidace sdružení, musí být tato skutečnost v pozvánce výslovně uvedena.

Valné shromáždění odhlasovalo následující znění čl. 5.5:

Čl 5.5. Výbor řídí sdružení v období mezi valnými shromážděními a za svou činnost je odpovědný valnému shromáždění. V období mezi schůzemi výboru řídí činnost sdružení předseda výboru. Výbor jedenkrát ročně podává členům informaci o hospodaření sdružení a o rámcových plánech činnosti a navrhuje výši členských příspěvků. Výbor může z naléhavých důvodů kooptovat další řádné členy sdružení. Toto je povinen oznámit na nejbližším valném shromáždění.

Návrh 12 byl přijat v následujícím znění [pro: 32, proti: 0, zdrželo se: 1]:

Čl. 4.2. Řádným členem sdružení se může stát každá osoba, která se přihlásí k myšlence poslání sdružení.

Bod 4.3 byl ponechán v původním znění [pro: 30, proti: 0, zdrželo se: 2].

Návrhy 13 a 13a byly přijaty v následujícím znění [pro: 28, proti: 0, zdrželo se: 3]:

3.1. Sdružuje uživatele a odborníky využívající programové vybavení pro stolní tisk, zejména založené na využívání jazyka $\text{\TeX}$. Sjednocuje jejich úsilí o efektivní propojení s typografickými podniky, které jsou zaměřeny na publikování v českém a slovenském jazyce. Chrání zájmy uživatelů vědecké a technické literatury využívajících náročnou technickou sazbu.

Byl přijat návrh na změnu čl. 5.3. ve znění [pro: 32, proti: 0, zdrželo se: 0]:

5.3. Výbor sdružení volí ze svého středu předsedu. Pověřuje jednotlivé své členy vybranými úkoly tak, aby byla zajištěna operativní a efektivní činnost sdružení.

Byly přijaty návrhy změn článků 5.2, 5.3 a 5.4 v následujícím znění [pro: 32, proti: 0, zdrželo se: 0]:

5.2. Výkonným orgánem sdružení je výbor volený v počtu 10–15 členů. Výbor je volen a odvoláván valným shromážděním sdružení, a to nadpoloviční většinou. Volby mohou být organizovány korespondenčně. Funkční období voleného výboru je nejvýše tříleté. Funkce člena výboru není honorována, členům výboru lze proplácet nutné náklady spojené např. s cestou na zasedání apod. Výbor pověřuje jednotlivé své členy vybranými úkoly tak, aby byla zajištěna operativní a efektivní činnost sdružení.

5.3. Výbor sdružení volí ze svého středu předsedu. Ve funkčním období výboru je možno rozhodnutím výboru obsazovat funkci předsedy na kratší období, nejméně však 1 rok. Nemůže-li předseda ze závažných důvodů dočasně vykonávat své povinnosti, musí pověřit se souhlasem výboru zastupováním jiného člena výboru a toto oznámit členům sdružení.

5.4. Výbor musí svolávat valné shromáždění alespoň jednou ročně v příslušném kalendářním roce. Pozvánky na valné shromáždění musí být rozeslány podle požadavků jednotlivých členů písemně nebo elektronicky nejpozději 4 týdny před termínem valného shromáždění. Pokud je na programu valného shromáždění závažný bod k projednání: změna stanov, volba výboru nebo likvidace sdružení, musí být tato skutečnost v pozvánce výslovně uvedena.

Byla přijata změna článku 5.1 ve znění [pro: 32, proti: 0, zdrželo se: 0]:

5.1. Nejvyšším orgánem sdružení je valné shromáždění členů.

Byla přijata změna článku 5.10 ve znění [pro: 32, proti: 0, zdrželo se: 0]:

5.10. O závažných otázkách valné shromáždění hlasuje. Právo hlasovat mají řádní členové a pověření zástupci kolektivních členů, jejichž hlas má váhu 3. Není-li výslovně stanoveno jinak, rozhoduje 30 min. po zahájení řádně svolané schůze prostá většina hlasů přítomných členů. Jeho rozhodnutí jsou pro výbor sdružení závazná.

Valné shromáždění ruší platnost stanov Československého sdružení uživatelů $\text{\TeX}$ u z roku 1990 pozměněných v roce 1996 a usneslo se na platnosti nových stanov Československého sdružení uživatelů $\text{\TeX}$ u, viz příloha. Od chvíle tohoto usnesení platí stanovy nové. [pro: 32, proti: 0, zdrželo se: 0]

9. Různé

Členové sdružení budou obesláni dotazem, zda mají zájem o CD-ROM CTAN.

Valné shromáždění schvaluje návrh výboru na výši členských příspěvků pro kalendářní rok 2003, která zůstane ve všech bodech stejná, jako v roce minulém: řádní členové: 300 Kč, řádní členové studenti: 200 Kč, řádní členové ze Slovenska: 300 Sk, řádní členové ze Slovenska studenti: 200 Sk, kolektivní členové: 1850 Kč, kolektivní členové gymnázia: 650 Kč. [pro: 32, proti: 0, zdrželo se: 0]

Valné shromáždění rozhoduje, že termín zaplacení členského příspěvku podle nových stanov je 30. červen. [pro: 32, proti: 0, zdrželo se: 0]

Valné shromáždění ukládá povinnost výboru a administrativě zaslat před termínem zaplacení členského příspěvku všem členům, kteří dosud nezaplatili, e-mailovou výzvu k zaplacení alespoň třikrát s časovým rozestupem aspoň jeden měsíc. Členové bez e-mailové adresy a kolektivní členové, kteří nezaplatili, budou vyzváni alespoň jednou klasickou poštou. [pro: 32, proti: 0, zdrželo se: 0]

Valné shromáždění ruší platnost Přílohy ke stanovám. [pro: 23, proti: 0, zdrželo se: 9]

Valné shromáždění se usnáší, že řádní členové budou nadále dostávat jeden výtisk každého čísla Zpravodaje a kolektivní členové pět výtisků. Ve stejném poměru bude členům zasílán i CDROM $\text{\TeX}$ live. Kolektivní členové navíc dostanou jedno balení CDROM CTAN, které každoročně vydává DANTE – německé sdružení uživatelů $\text{\TeX}$ u. Výbor může rozhodnout o dalším vhodném materiálu k rozesílání. [pro: 32, proti: 0, zdrželo se: 0]

Zapsal: Jiří Kosek, jirka@kosek.cz

Ověřil: Petr Olšák, olsak@math.feld.cvut.cz, předseda

Stanovy Československého sdružení uživatelů T_EXu

1. Poslání Československého sdružení uživatelů T_EXu.

Československé sdružení uživatelů T_EXu je dobrovolným sdružením ve smyslu Zákona o sdružování občanů 83/1990 a jeho novely 68/1993 Sb.

Základním posláním Československého sdružení uživatelů T_EXu je vytvářet předpoklady pro všestranné využívání a další rozvoj jazyka počítačové typografie T_EX a příbuzného programového vybavení pro stolní tisk (tzv. desktop publishing, krátce DTP) zejména českými a slovenskými uživateli.

2. Základní ustanovení.

1. Československé sdružení uživatelů T_EXu je dobrovolná společnost uživatelů programového vybavení pro stolní tisk. Je především zaměřeno na ta vybavení, která jsou založena na jazyce T_EX. Působí na území České republiky.

2. Československé sdružení uživatelů T_EXu je samostatnou právnickou osobou se sídlem v Praze; sídlí na adrese Technická 2, 166 27 Praha 6.

3. V mezinárodní oblasti sdružení vystupuje pod názvem Czech and Slovak T_EX Users Group a užívá zkráceného označení CSTUG.

3. Činnost sdružení.

1. Sdružuje uživatele a odborníky využívající programové vybavení pro stolní tisk, zejména založené na využívání jazyka T_EX. Sjednocuje jejich úsilí o efektivní propojení s typografickými podniky, které jsou zaměřeny na publikování v českém a slovenském jazyce. Chrání zájmy uživatelů vědecké a technické literatury využívajících náročnou technickou sazbu.

2. Spolupracuje s institucemi využívajícími výpočetní techniku v oblasti typografie, vydavatelstvími odborných časopisů a odbornými společnostmi.

3. Spolupracuje s obdobně zaměřenými společnostmi vytvářenými na základě územní či jazykové jednoty a společností T_EX Users Group se sídlem v USA s celosvětovou působností.

4. Pořádá školení, diskuse, exkurze, konference, sympozia a semináře. Vyvíjí expertní, publikační, vydavatelskou a další odbornou činnost v oboru kvalitní

typografie a náročné technické sazby a DTP. Produkuje nebo zadává vytváření programového vybavení pro oblast využívání $\text{\TeX}$ u a toto vybavení udržuje a distribuuje členům.

5. Podporuje vytváření archivů softwaru (zejména $\text{\TeX}$ ware) dostupných prostřednictvím sítě a jejich udržování.

6. Zajišťuje základní administrativu sdružení případně prostřednictvím síly či sil najímaných za úplatu.

4. Členství v Československém sdružení uživatelů $\text{\TeX}$ u, práva a povinnosti členů.

1. Československé sdružení uživatelů $\text{\TeX}$ u má členy řádné a členy kolektivní z České republiky i ze zahraničí, zejména ze Slovenské republiky.

2. Řádným členem sdružení se může stát každá osoba, která se přihlásí k myšlence posílání sdružení.

3. Kolektivním členem sdružení se může stát každá instituce nebo podnik se zájmem o využívání typografie založené na použití $\text{\TeX}$ u, pokud se ztotožní s posláním sdružení.

4. Řádné i kolektivní členství v Československém sdružení uživatelů $\text{\TeX}$ u vzniká tak, že zájemce o členství vyplní předepsaný formulář (v písemné nebo elektronické podobě) a zaplatí členský příspěvek na kalendářní rok v plné výši podle typu členství.

5. Řádný člen má právo volit a být volen do orgánů sdružení. Za kolektivního člena má právo volit jeho pověřený zástupce s váhou 3 hlasy.

6. Členství v Československém sdružení uživatelů $\text{\TeX}$ u zaniká vystoupením, vyloučením člena nebo nezaplacením členského příspěvku ve stanoveném termínu. Vyloučit člena může výbor sdružení, jestliže pro toto rozhodnutí hlasuje aspoň polovina jeho členů, a to z vážných důvodů, zejména pro dlouhodobé neplnění povinností člena sdružení.

7. Všichni členové sdružení mají právo být pravidelně informováni o činnosti orgánů sdružení. Informace jsou šířeny převážně prostřednictvím sítě Internet, jsou publikovány ve Zpravodaji sdružení a v závažných případech rozesílány poštou.

8. Výši členských příspěvků i termín zaplacení příspěvku pro následující kalendářní rok schvaluje valné shromáždění. Pokud údaje o výši členských příspěvků a termínu zaplacení neprojedná, tyto údaje se pro následující rok nemění.

5. Organizační struktura sdružení.

1. Nejvyšším orgánem sdružení je valné shromáždění členů.
2. Výkonným orgánem sdružení je výbor volený v počtu 10 – 15 členů. Výbor je volen a odvoláván valným shromážděním sdružení, a to nadpoloviční většinou. Volby mohou být organizovány korespondenčně. Funkční období voleného výboru je nejvýše tříleté. Funkce člena výboru není honorována, členům výboru lze proplácet nutné náklady spojené např. s cestou na zasedání apod. Výbor pověřuje jednotlivé své členy vybranými úkoly tak, aby byla zajištěna operativní a efektivní činnost sdružení.
3. Výbor sdružení volí ze svého středu předsedu. Ve funkčním období výboru je možno rozhodnutím výboru obsazovat funkci předsedy na kratší období, nejméně však 1 rok. Nemůže-li předseda ze závažných důvodů dočasně vykonávat své povinnosti, musí pověřit se souhlasem výboru zastupováním jiného člena výboru a toto oznámit členům sdružení.
4. Výbor musí svolávat valné shromáždění alespoň jednou ročně v příslušném kalendářním roce. Pozvánky na valné shromáždění musí být rozeslány podle požadavků jednotlivých členů písemně nebo elektronicky nejpozději 4 týdny před termínem valného shromáždění. Pokud je na programu valného shromáždění závažný bod k projednání: změna stanov, volba výboru nebo likvidace sdružení, musí být tato skutečnost v pozvánce výslovně uvedena.
5. Výbor řídí sdružení v období mezi valnými shromážděními a za svou činnost je odpovědný valnému shromáždění. V období mezi schůzemi výboru řídí činnost sdružení předseda výboru. Výbor podává jedenkrát ročně členům informaci o hospodaření sdružení, o rámcových plánech činnosti a navrhuje výši členských příspěvků. Výbor může z naléhavých důvodů kooptovat další řádné členy sdružení. Toto je povinen oznámit na nejbližším valném shromáždění.
6. Za Československé sdružení uživatelů T_PXu jedná předseda nebo jiný výborem pověřený člen sdružení. O svých rozhodnutích podává členstvu zprávu ve Zpravodaji sdružení, zpravidla formou sdělení předsedy.
7. Ze zasedání výboru je nutno pořídit zápis, který podepisuje člen výboru pověřený jeho zpracováním a autorizuje předseda. Členové výboru se dohodnou na technické realizaci způsobu zasedání výboru a pořizování zápisů ze zasedání. Do zápisů zasedání výboru může kdykoli nahlédnout kterýkoli člen sdružení.
8. V rámci sdružení pracují pracovní skupiny zřizované výborem, které sdružují členy podle specifických zájmů, resp. na regionálním principu. Tyto skupiny se řídí organizačním řádem schvalovaným výkonným výborem.

9. K řešení specifických problémů jsou výborem jmenovány komise. Sdružení dle potřeby zaměstná k výkonu administrativních prací apod. pracovníky na základě smlouvy, kterou schvaluje výbor sdružení.

10. Kontrolními orgány sdružení jsou revizoři volení valným shromážděním.

11. O závažných otázkách valné shromáždění hlasuje. Právo hlasovat mají řádní členové a pověřeni zástupci kolektivních členů, jejichž hlas má váhu 3. Není-li výslovně stanoveno jinak, rozhoduje 30 min. po zahájení řádně svolané schůze prostá většina hlasů přítomných členů. Jeho rozhodnutí jsou pro výbor sdružení závazná.

12. K řešení sporů uvnitř sdružení volí valné shromáždění rozhodčí komisi, která je podřízena přímo valnému shromáždění sdružení.

6. Hospodaření sdružení.

Činnost sdružení je hospodářsky zajišťována

- (a) z příspěvků členů,
- (b) z výtěžku služeb poskytovaných sdružením,
- (c) z darů a odkazů.

1. Za finanční hospodaření sdružení zodpovídá výbor sdružení, dispoziční právo s účty má předseda sdružení. Výbor zpravidla pověřuje svého člena (hospodáře), aby pomáhal předsedovi s hospodářskou agendou a případně s řízením práce kvalifikované externí síly, pověřené péčí o účetní doklady sdružení. Ve věcech finančních zodpovídá výboru za využívání bankovních účtů předseda sdružení, a to vždy v příslušném funkčním období.

2. Zpráva o hospodaření se předkládá pravidelně jednou ročně členstvu, zpravidla na valném shromáždění sdružení.

7. Závěrečná ustanovení.

1. Stanovy Československého sdružení uživatelů $\text{T}_{\text{E}}\text{X}$ u mohou být měněny pouze rozhodnutím valného shromáždění sdružení, jestliže se pro změnu vysloví více než polovina odevzdaných hlasů.

2. Sdružení zaniká usnesením valného shromáždění sdružení, vysloví-li se pro tento návrh aspoň dvě třetiny odevzdaných hlasů. Všechny majetek, který zůstane po likvidaci, bude odevzdán orgánům určeným usnesením valného shromáždění o likvidaci.

3. Tato verze stanov navazuje na stanovy sdružení registrované 9. května 1990 a pozměněné 8. července 1996. Poslední (v pořadí druhé) změny stanov sdružení byly schváleny valným shromážděním sdružení dne 30. listopadu 2002.

Zpráva o hospodaření k 31. 10. 2002

Údaje jsou v korunách.

Vybrané údaje z rozvahy:

stav	k 1. 1. 2002	k 31. 10. 2002
běžného účtu	177 984,83	47 815,24
termínovaného vkladu	450 000,00	600 000,00
pokladny CZK	996,00	1 343,20
pokladny SKK	2 941,10	nedodány doklady

Náklady:

nákup spotřebního materiálu	443,00
spotřební materiál na Zpravodaj	13 272,00
poštovné	16 739,20
T _E X CTAN 2001	7 934,77
účetnictví 2002 (ECQUO)	6 300,00
účetnictví 2001 + pokladna 2002	22 381,00
platba domény CSTUG.CZ	840,00
mzdové náklady	15 000,00
služební cesta EuroT _E X 2002	24 157,36
pokuty a penále	305,00
poplatky za vedení účtu	5 106,80
Celkem	112 479,13

Výnosy:

členské vklad – kolektivní	80 208,00
– individuální	88 740,00
úroky z běžného účtu	256,87
úroky z termínovaného vkladu	12 590,00
kusové zisky	177,45
Celkem	181 972,32

Zpracoval: Pavel Vlček (ECQUO) dne 17. 11. 2002

BachTeX 2003

The 11th annual meeting of Polish Tex Users Group GUST will traditionally be held at the Bachotek lake resort (Brodnica Lake District, Poland) on May 1–3, 2003. TeX users from around the world are cordially invited. Polish and English are the official meeting languages. The Web page of the meeting:

<http://www.gust.org.pl/BachTeX/2003en>.

Please note that one of the conference days will be devoted to L^AT_EX with special emphasis on most common typesetting problems, problem solving techniques, implementing own layouts with new font families, styles, classes etc. Details can be found on the Web page.

Planned lectures:

- TeXLive on the net – the concept and usage by Janusz Bien
- TypeScripts: the context way of combining fonts families by Hans Hagen
- DocBook, a practical exploitation of the XML standard by Tomasz Przechlewski
- Matrix diagram construction using XY-pic by Romuald Slupski
- MWcls in practice by Marcin Woliński

Participants presentations:

- Typesetting tables with METAPOST – Piotr Bolek
- T_EX and inter-zone telephone connections – Piotr Bolek
- How to make Jerzy love T_EX again? – Hans Hagen
- Latin Modern, a rich extension of the Computer Modern
- Family with accented characters using METATYPE1 – Bogusław Jackowski
- METAPOST macros for dashing closed areas – Bogusław Jackowski
- The Future of European LUGs – Jerzy Ludwichowski
- Installing outline fonts – Janusz Nowacki
- Polishing T_EX – Jerzy Ludwichowski, Andrzej Odyniec
- The history of a certain Web page or is L^AT_EX useable for WWW services – Wojciech Myszka
- Towards a better quality of PDF documents (configuring pdfTeX/dvipdf/dvips and ghostscript to use vector fonts) – Grzegorz Sapijaszko
- Managing software configurations with respect to T_EX – Tomasz Serafiński

Also: discussions with friends (free), bonfires (free), swimming (free), canoeing (free), singing at nights (free and obligatory), beer and similar (at own expense), new initiatives (dear to the organisers).

Paper submissions (with abstracts) and proposals for tutorials should be sent by March 20, 2003 to the Program Committee Chairman Andrzej Borzyszkowski (Email: BachTeX@gust.org.pl).

Prices:

The conference fee.....	100 EUR
Special fee for members of TeX users groups	80 EUR
Students fee	50 EUR
Students –TUG/LUG/GUST members	40 EUR

The fee includes registration and conference materials for regular participants. Accomodation and full board (dinner, breakfast and lunch) is at 19 EUR per day.

For payment after March 31st and before April 18th please add 10 EUR (students – 5 EUR) and for late payment (after April 18) please add 20 EUR (students – 10 EUR), respectively. Please note that on-site payments can only be made in cash, in PLN (at the current rate).

The deadline for registration is April 18th, 2003.

The participants are expected on April 30th (Wednesday, afternoon) but it is possible to extend the stay in Bachotek from April 29 (Tuesday, afternoon) to May 4 (Sunday).

For more information, please contact the GUST secretary:

*Jola Szalatynska, Uczelniane Centrum Informatyczne UMK, Gagarina 7
87-100 Torun, POLAND, phone: +48 (56) 611-45-16
Email: secretary@gust.org.pl*

Automatizace sazby a skenování formulářů¹

MIROSLAV HRAD, PETR SOJKA

V článku je popsána technologie SCAT pro přípravu a skenování formulářů, která byla v minulých třech letech použita na Masarykově univerzitě a v dalších organizacích na zpracování téměř statisíce formulářů vysazených $\text{\TeX}$ em a zpracovaných za použití nástrojů otevřených systémů. Jsou diskutovány prvky navrženého systému směřující k minimální chybovosti (použití čárových kódů), rychlosti, anonymitě, maximální bezpečnosti a automatizaci zpracování.

Klíčová slova: SCAT, skenování, rozpoznávání, formuláře, čárové kódy, $\text{\TeX}$, Linux, automatizace testů

1. Úvod

Vkládání dat člověkem do počítače je činnost častokrát časově náročnější než jejich samotné počítačové zpracování. Ruční vkládání dat s sebou navíc přináší velké procento chyb a jejich odstraňování bývá často netriviální záležitostí. Nejinak je tomu i v případě vytváření, zpracování a vyhodnocování písemných testů.

1.1. Potřeba automatizace

Vytváření, zpracování a vyhodnocování testů je běžnou stereotypní prací všech, kteří zadávají písemné zkoušky na všech typech škol. Může se jednat jak o obyčejné zkoušení studentů – třeba na střední škole, tak o přijímací zkoušky na vysokou školu.

1.2. Východiska

Použití všech nabízených komerčních řešení dostupných na trhu selhávalo na nemožnosti přizpůsobení konkrétním požadavkům. Outsourcing každodenních potřeb se nejeví jako ideální stav. Vhodnost použití „in-house“ řešení, jak se ukázalo později, umožňuje *mnohem* vyšší komfort a flexibilitu vyhodnocení testů než řešení založené na uzavřených systémech.

V rámci série bakalářských a diplomových prací [4, 2] se postupně vyvíjely algoritmy a ověřovala se technika potřebná pro realizaci automatizace zpracování agendy přijímacího řízení nebo vyhodnocení písemných testů. Postupným

¹Článek byl poprvé publikován ve sborníku Jan Kasprzak, Petr Sojka: SLT 2002, KONVOJ 2002, ISBN 80-7302-043-2, na stranách 11–26. Publikováno se souhlasem vydavatele a autorů.

vývojem a dlouhým testováním systému se nakonec ukázalo, že je možno vytvořit spolehlivý systém a technologii nevyžadující vysoce specializovaný a drahý hardware.

1.3. Členění článku

V oddíle 2 je definován proces vytváření a zpracování formulářů a jsou popsána východiska projektu. V sekcích 3 a 4 je diskutována problematika návrhu a sazby testů s použitím technologie čárových kódů. Oddíl 5 rozebírá algoritmy rozpoznávání symbolů a textů ve formulářích. V oddíle 6 je pak zmíněn způsob vyhodnocení a zajištění anonymity zpracování až do finálního zveřejnění výsledků testu.

2. Proces vytváření a zpracování formulářů

Technologie pro automatizované vyhodnocování testů s využitím skeneru s podavačem musí řešit tyto etapy:

1. analýza specifických požadavků projektu
2. vytvoření testů a jejich testování, sazba
3. návrh odpovědních formulářů
4. sazba a tisk odpovědních formulářů
5. distribuce a sběr testů a formulářů
6. skenování formulářů
7. vyhodnocení naskenovaných formulářů
8. vyhodnocení výsledků testu a jejich prezentace
9. vyhodnocení statistických parametrů testových otázek a příkladů
10. zařazení formulářů se zadáními pro dlouhodobou archivaci.

2.1. Technologie SCAT

Při vzniku projektu SCAT se vycházelo z dosavadního způsobu vyhodnocování přijímacího testu na fakultu informatiky a pedagogickou fakultu MU v Brně. Byl používán výsledkový list, který měl dvě části. Část s identifikačními údaji a část s oválky pro odpovědi na otázky. Tyto dvě části byly oddělitelné a logicky je spojovalo pouze shodné, předem vygenerované číslo. Tento způsob byl použit právě z důvodu anonymity vyhodnocování.

2.2. Cíle a požadavky při návrhu systému SCAT

Pro projekt jsme si stanovili následující cíle a požadavky na systém:

- flexibilní vytváření vlastních testových formulářů,

- jednoduchá integrace vytvořených formulářů do systému zpracování,
- bezchybné rozhodování – zpracování testů,
- zvýšení rychlosti zpracování,
- vytvoření logistiky zpracování pro přijímací zkoušky,
- zaručení bezpečnosti – neovlivnitelnosti výsledků,
- standardizovaný výstup, aby bylo možné výsledky exportovat do současných systémů,
- jednoduché ovládání,
- otevřenost systému vůči novým prvkům a přístupům,
- minimální hardwarové požadavky,
- případná přenositelnost na jiné systémy, standardně bude použit operační systém Linux.

Požadovali jsme systém, který umožní vyhodnotit *více správných odpovědí*, tedy bude zpracovávat každý vyhodnocovací prvek – čtvereček a určí o něm jeho stav (vyplněn/nevyplněn) bez ohledu na ostatní odpovědi v dané otázce.

2.3. Výpočetní technika a software systému SCAT

Požadovali jsme, aby bylo v maximální míře využito všeobecně dostupného hardware a software, s maximálním podílem otevřeného software, což zajišťuje vysokou míru přenositelnosti a rozšiřitelnosti.

Využíváme běžných počítačů PC s operačním systémem Linux a skenerů běžné cenové kategorie. Nárazové nasazení (přijímací zkoušky) systému tedy nevyžaduje (s možnou výjimkou skeneru) vyšší finanční nároky.

Pro implementaci jádra systému SCAT jsme zvolili jazyk C. Ostatní komponenty (především logistika systému) je napsána ve skriptovacím jazyce Perl. Sazba všech výstupů včetně formulářů je realizována sázecím systémem T_EX a programem METAFONT.

3. Návrh vyhodnocovacího formuláře

Významným omezením některých softwarových produktů pro vyhodnocování testů je fixní množina předtisknutých formulářů – například Recognita Test. Naším cílem bylo navrhnout technologii dostatečně flexibilně, bez omezení na počet otázek, odpovědí, textů či způsob identifikace.

Při vyhodnocování písemných testů pro přijímací řízení je nutné maximálně eliminovat možnost záměrného zásahu do objektivnosti zjištěných výsledků, tedy mimo jiné dosáhnout anonymity. To lze zajistit například tím, že oddělíme odpovědní část formuláře od části identifikační. Obě části pak zpracováváme samostatně nebo identifikační část zpracováváme až po zpracování části odpovědní (ve stavu, kdy už nelze zjištěné výsledky ovlivňovat). Tento způsob zpracování

ovšem vyžaduje navrhnout formuláře tak, aby části byly jednoduše oddělitelné (lze zajistit třeba perforací formuláře), ale především, abychom byli schopni přiřadit výsledky z odpovědní části k identifikaci. Jedním ze způsobů, jak zajistit onu *logickou vazbu*, je obě části vybavit shodným číslem – *vazebním číslem*. Je zřejmé, že jednotlivé testové formuláře pak musí mít vždy vzájemně rozdílná vazební čísla. Z této nutnosti plyne potřeba generovat a tisknout jednotlivé formuláře, naopak není možné formuláře kopírovat. Vazební číslo je vhodné navrhnout tak, aby se dalo využít technologie čárových kódů [1], neboť rozpoznávání čárových kódů je při dodržení určitých podmínek výrazně méně chybové než technologie OCR².

V případě, že není nutné zajišťovat anonymitu v průběhu vyhodnocování, a lze tedy zpracovávat formulář, který obsahuje jak odpovědi, tak identifikaci, pak mohou být testové formuláře vzájemně shodné a lze vytvářet jejich kopie a ty potom zpracovávat. Takový formulář využijeme třeba při běžném písemném zkoušení. Při vytváření nám odpadne potřeba vazebních čísel a také vyhodnocení je jednodušší, identifikace i odpovědi se zpracovávají zároveň – při jednom průchodu skenovacím zařízením.

4. Sazba formulářů testů

Na rozhodnutí, jakým způsobem a kterým sazečím systémem budeme vytvářet testové formuláře, měly vliv následující potřeby:

- automatické generování formulářů, jejich čísel a čárových kódů,
- možnost převedení informací z výstupu na souřadnice pro systém,
- nezávislost přípravy formulářů na operačním systému.

Sazba čárových kódů *závisí* na použitém výstupním zařízení a tiskové technice (laserová tiskárna, ofsetový tisk) a je třeba provést *kalibraci* (bar correction) tlouštěk čar sazených čárových kódů.

4.1. Návrh testů

Nejsnazším způsobem je test navrhnout jako sadu příkladů s možností výběru z N správných řešení, „multiple-choice test“, N obvykle 3–5. Pro minimalizaci možnosti opisování vedle sebe sedících zkoušených je vhodné mít testy generované zcela individuálně [7], nebo mít alespoň několik variant testu s vzájemně permutovaným pořadím otázek pro znesnadnění podvádění.

Detailní rozbor návrhu obsahu testů je mimo rozsah tohoto příspěvku, každopádně je vhodné softwarově podporovat alespoň statistické vyhodnocení složitosti jednotlivých příkladů z realizovaného testu.

²Optical character recognition

4.2. Generování testů

Až na výjimky je třeba připravit různé, případně i plně individuální verze testových odpovědních formulářů nebo i samotných testů. Zejména pro možnost vysázení individuálních testů (databázového publikování [11]) je ideální a plně se osvědčil sázecí systém $\text{T}_{\text{E}}\text{X}$ ³. $\text{T}_{\text{E}}\text{X}$ dokonce může sám permutovat příklady [7] a zapisovat soubory pro vyhodnocení, ale toto lze většinou realizovat místo složitěho makroprogramování externím skriptem ve skriptovacím jazyce (obvykle Perl).

Ani sazba čárových kódů EAN nepředstavuje problém, existují například makra popsaná v článku [6]. Pro námi požadovaný rozsah kódovaných čísel postačuje například čárový kód EAN-13, používaný i v systému ISBN.

Při návrhu formuláře musí existovat způsob, jak komunikovat prvky formuláře a jejich umístění vyhodnocovací části systému. V otevřeném a dobře dokumentovaném systému, jako je $\text{T}_{\text{E}}\text{X}$, to není problém.

4.2.1. Vložení vlastní informace pro generování souřadnic

Do zdrojového souboru pro sazbu $\text{T}_{\text{E}}\text{X}$ em můžeme vložit příkaz `\special` [3, kapitola 21]. Tento příkaz, jehož argument nebude sázen, je beze změny přenesen do souboru DVI.

Pro automatický vznik souřadnic a pro správné ohodnocení odpovědních čtverečků potřebujeme do zdroje pro sazbu vložit informaci o počtu odpovědí v otázce. Tedy např. `\special{pocet_ctv 3}` znamená, že budou následovat otázky se třemi odpovědními čtverečky – A, B, C . Tuto informaci musíme vložit před každou změnu počtu odpovědí. Standardně je možné nastavit až deset odpovědí v rámci jedné otázky.

4.2.2. Soubor DVI

Výstupní soubor $\text{T}_{\text{E}}\text{X}$ u je binární soubor DVI⁴, který čtou ovladače jednotlivých výstupních zařízení, například ovladač obrazovky, ovladač tiskárny. Jeden z takových ovladačů je program `dvitype`, který převede všechny informace z binárního souboru do čitelné podoby – textového souboru, který obsahuje všechny údaje o pohybu bodu sazby a o kódech jednotlivých znaků v relativně srozumitelné podobě.

Pokud se provádí sazba znaku například povel `set_char_i`, znak se umístí tak, aby se jeho referenční bod kryl s bodem sazby a po vykreslení znaku se bod sazby posune o šířku znaku doprava. K tomu ovladač potřebuje znát šířky všech znaků, které jsou uloženy v TFM. Ovladači může postačit formát PK, protože tam

³Používáme distribuci $\text{T}_{\text{E}}\text{X}$ Live

⁴DVI – Device Independent – formát nezávislý na zařízení

jsou údaje o šířkách znaků (nikoli však o výškách a hloubkách) duplikovány se stejnou přesností jako v TFM.

4.2.3. Shrnutí vzniku systému souřadnic

Zaměřovací prvky, zaškrťovací prvky a prvky pro identifikaci vytvoříme v METAFONTu, jako METAFONTové znaky. Na obrázku 1 jsou zobrazeny připravené METAFONTové znaky. Ty pak můžeme jednoduše používat pro tvorbu nových testových formulářů. Příklady sazby formuláře jsou uvedeny na straně 136 a straně následující. Pomocí sázecího systému T_EX získáme obraz vysázeného formuláře ve formátu DVI. Z formátu DVI jsou pak zjistitelné pozice sazby na straně s využitím programu `dvitype`.

Obrázek 1: Prvky v METAFONTu

Například víme, že zaškrťovací prvek je v METAFONTu reprezentován jako znak číslo dvě v daném fontu. Zvolíme úroveň výpisu programu `dvitype` alespoň tři a ve výpisu potom hledáme řetězec „`setchar2`“, např.:

```
4654: setchar2 h:=2660904+792488=3453392, hh:=219
```

Tímto získáme horizontální souřadnici h v kartézském systému souřadnic (h, v) , resp. přímo souřadnici hh v pixelech. Souřadnici v , resp. souřadnici vv v pixelech, najdeme ve výpisu jako poslední změnu vertikální pozice před místem sázení znaku „`setchar2`“, např.:

```
4649: push level 6: (h=--2797019,v=15156135,w=0,x=0,
y=0,z=0,hh=--177,vv=960)
```

Takto můžeme programově získávat údaje z formuláře do konfiguračního souboru skenovacího programu.

4.2.4. Specifikace požadavků tisku

Protože jde o relativně velké tiskové objemy, je třeba si před hromadným tiskem odsouhlasit přesný seznam formulářů a jejich parametrů. Ukázalo se nanejvýš vhodné, aby se z jednoho a téhož souboru tiskly jak zkušební sada formulářů, tak jejich seznam s jejich počty. Vše je zadáno v textovém souboru, ze kterého se údaje čtou pro různé výstupy (stačí příkazem `\let` změnit před načtením chování hlavního makra).

```

\datum{červen 2002}      % nastavení implicitní hodnoty data
\setcounter{nanswers}{4} % počet možných odpovědí
\vartext{Fakulta X}      % variabilní text
\idtext{}                % identifikační text
\id = 0                  % počáteční nastavení čítače čísel testů

```

```

\obor{Písemný test zaměřený na ověření obecných
studijních předpokladů}
\studium{\TeX u a Linuxu}
\sada 10 1 99 1000      % počet testů, kód, verze, typ studia
\idtext{20.\,6.\,2002}

```

```

\obor{Přijímací písemná zkouška na obor anglický jazyk
pro základní školy}
\studium{bakalářské studium kombinované}
\sada 20 1 76 300      % počet testů, kód, verze, místnost
...

```

Na straně 136 najdete příklady vysázených formulářů.

5. Skenování

Úspěšnost skenování je vysoce závislá na schopnosti ovládat a využít technické parametry dostupné techniky a doladit parametry použitého software [9, 10]. Ceny specializovaných vysokokapacitních skenerů dosahují statisícových částek, a proto jsem se zaměřili na ověření možnosti využít skenerů cenových kategorií v řádu desetitisíců korun českých.

5.1. Skenovací zařízení

Pro vývoj našeho systému jsme použili skenovací zařízení firmy Hewlett Packard HP 6100 C s automatickým podavačem a skenovací zařízení HP 6350 C s automatickým podavačem ScanJet ADF. Zařízení byla připojena přes SCSI řadič.

Podavač skeneru HP 6100 C má kapacitu maximálně 50 listů. Rychlost podávání a skenování jedné černobílé strany při rozlišení 300 dpi je 27–30 sekund. Předloha ani snímač se ve skeneru nepohybují. Pohybuje se čtecí paprsek, úzký paprsek světla usměrňovaný soustavou zrcadel. Množství světla, které se od snímaného předmětu odrazilo, se měří a digitalizuje.

Skener HP 6350 C je dodáván s automatickým podavačem dokumentů na 25 stran. Rychlost podávání a skenování jedné černobílé strany při rozlišení 300 dpi je 17–18 sekund. Komunikační rozhraní je USB nebo SCSI.

Skener HP 6350 C (obdobně jako jeho předchůdce) ke snímání používá pouze jednu řadu CCD senzorů. Zásadní rozdíl oproti skeneru HP 6100 C je v použitém

podavači. Automatický podavač ScanJet ADF, jenž je dodáván ke skeneru HP 6100 C, nejprve papírový dokument zavede do skeneru a potom skenuje, zatímco u skeneru HP 6350 C je dokument skenován v průběhu vedení podavačem. Tato změna přinesla výhodu ve formě zrychlení podávání, nicméně ohýbání dokumentu se děje nad čtecím prostorem a proto je vhodné pravidelně po několika stovkách naskenovaných listů toto místo vyčistit. Pro testové formuláře je vhodné právě z tohoto důvodu používat kvalitní papír, který se otírá minimálně a je bez kazů.

5.2. Varianty zaškrťovacích prvků

Na obrázku 2 jsou zobrazeny příklady zaškrťovacích prvků – mohou být použity různé tvary. Oválky byly voleny pro jednodušší vyplňování. Klasickým tvarem pro zaškrťování je čtvereček. Pro snížení tmavosti strany, resp. pro zvýraznění vybraných odpovědí, byly navrženy čtverečky kreslené přerušovanou čarou.

Obrázek 2: Varianty zaškrťovacích prvků

5.3. Čím zaškrťávat, jak vyhodnocovat

Ideálním prostředkem pro zaškrťování jsou tenké fixy, propisovací tužky a pera v barvách černá, modrá, červená, zelená. Program je navržen tak, že rozpoznává zaškrtnutí i tenkou obyčejnou tužkou (pentelkou). Lze tedy vyplňovat téměř jakýmkoliv běžným prostředkem pro psaní po papíru s výjimkou nestandardních psacích pomůcek jako jsou žlutý či oranžový zvýrazňovač.

Významnou otázkou bylo, co vlastně považovat za vyplněný (zaškrtnutý) čtvereček. Při návrhu a měření schopností jsme se dostali na hranici, kdy je rozpoznána i drobná tečka, malá čárka a podobně. Nicméně již ve fázi testování se ukázalo, že některé odpovědi jsou takto drobně označeny jenom proto, že nad nimi student přemýšlel, ale nelze je považovat za zaškrtnuté.

Hodnotová funkce f je počítána z obou oblastí *hodnotaIN* a *hodnotaOUT* tak, jak je zobrazeno na obrázku 3. Majoritní úlohu ve výpočtu hodnotové funkce má oblast *hodnotaIN*, takže i stejná značka (čárka, tečka, malý křížek) má v oblasti bližší ke středu větší hodnotu, než kdyby byl umístěn na okraji. Obor hodnot funkce f je $(0, \dots, 5600)$, kde 0 je nevyplněný a 5600 je zcela vyplněný čtvereček. Běžně, křížkem vyplněný čtvereček má hodnotu cca 1600–1800.

Drobná tečka cca 400. Současná *prahová hranice* pro rozhodnutí je nastavena na 960.

Obrázek 3: Výpočet hodnotové funkce

5.4. Posun a pootočení při tisku a skenování

Posun a pootočení naskenovaného obrazu vůči očekávaným pozicím je jednou z nejmarkantnějších nepřesností, které musíme řešit. Z převodu DVI očekáváme přesné pozice umístění vytisknutých prvků na papíru, ty se však v závislosti na tiskárně, kopírovacím stroji i skeneru liší. Posun i pootočení je obvykle způsobeno podavačem, většinou se jedná o několik milimetrů až centimetrů. Právě řešení tohoto problému je zcela nezbytné k dosažení potřebné přesnosti.

5.5. Detekce a korekce změn velikostí

Při tisku a zvláště při kopírování testových formulářů může docházet na některých zařízeních k zvětšení nebo zmenšení tiskového obrazu vůči standardním rozměrům daným v souboru DVI, resp. vůči originálu při kopírování. Změna je tak ve shodném poměru v ose x i y .

$$\frac{x'}{x} = \frac{y'}{y}$$

Může taktéž docházet ke změně velikosti pouze v ose y , tedy k jakémusi protažení obrazu. Tento defekt se projevuje právě u použitého skeneru HP 6350 C. Skener má podavač, jenž při průchodu papíru zároveň daný dokument skenuje. Při nepřesném podávání tak dochází k nechtěnému protažení, jak je ukázáno na obrázku 4.

$$\frac{x'}{x} = 1$$

$$\frac{y'}{y} = k, \text{ kde } k \text{ je např. } 1,007$$

Při větším množství zpracovávaných testů (více než několik tisíc) může docházet i k následnému opotřebení zařízení podavače. Vodící kolečka pro podávání papíru vlivem opotřebení mírně zmenšují svůj průměr a načítání dokumentu tak trvá delší čas, což se projeví protažením naskenovaného obrazu ve směru osy y , a tím se tedy mění i k . Hodnota k se průběžně počítá ze zaměřovacích prvků.

Obrázek 4: Zvětšení v osách x , y a pouze y

5.6. Zaměřovací prvky

Zaměřovacími prvky mohou být buď zaměřovací křížky nebo čtverečky. Slouží k jednoznačnému a přesnému zaměření nasnímaného obrazu tak, aby mohly být přepočteny všechny souřadnice.

Nalezení prvků musí být bezchybné, jednoznačné a rychlé. Byly vyzkoušeny různé varianty, nicméně zaměřovací čtvereček je nejvíce odolný vůči různým nežádoucím poruchám snímání i nepřesnému zaostření. Na obrázku 5 jsou zobrazeny zvětšeniny zaměřovacích prvků tak, jak je nasnímá skener. Okraje prvků jsou nerovné, neostře. Vzhledem k tomu, že nalezení a kontrola nalezení černého čtverečku je i algoritmicky jednoduché, je v současnosti používán právě tento zaměřovací prvek.

Ze zaměřovacího prvku zjistíme *bod zaměření*, a to pro každý zaměřovací prvek. V našem případě obvykle používáme dva v protilehlých rozích testu, jak je vidět v příkladech na straně 136 a dalších. Ze zjištěných údajů a očekávaných údajů (vzniklých při převodu DVI na souřadnicový systém) vypočteme hodnoty zpětného posunutí, pootočení a zvětšení (respektive zmenšení). Tím z nasnímaného obrazu získáme očekávaný obraz.

Obrázek 5: Varianty zaměřovacích prvků

5.7. Identifikace pomocí osobního čísla a čárového kódu

Na obrázku 6 je zobrazena identifikační část formuláře, který byl použit pro přijímací zkoušky v roce 2000. Vyplňující vypíše své příjmení, jméno, adresu, rodné číslo a především své osobní číslo. Na identifikační části i na části odpovědní je vytisknut shodný čárový kód. Zpracování tohoto formuláře spočívá v ručním opisování osobního čísla studentů (6 znaků) a kódového čísla formuláře (4 znaky). Tato identifikační část je oddělená od odpovědní části perforací. Formulář se po vyplnění rozdělí na dvě samostatné části, které jsou propojeny pouze logickou identifikací – předem na obě části vygenerovaným shodným čárovým kódem. Zpracování jednotlivých částí pak probíhá odděleně, a tím tedy i anonymně.

Přijímací zkouška – písemná část

červen 2000

Údaje v části nad čarou vyplňte čitelně hůlkovým písmem, oddělte a ústřížek na pokyn odevzdejte.

Příjmení: jméno:

adresa:

rodné číslo: UUUUUU / UUUU osobní číslo (viz pozvánka): UUUUUU 0001

Obrázek 6: Identifikace s využitím čárového kódu

5.8. Identifikace ve formě digitálního čísla

Příklad na obrázku 7 ukazuje zapsání UČO: 003908 a číslo písemky: 724 do formy digitálních čísel.

JMÉNO: _____ Č. PÍSEMKY: _____
 UČO: _____ DATUM: _____
 PŘEDMĚT: _____

Podle tohoto vzoru: 0:123456789 vyplňte následující údaje:

UČO	písemka
<div style="display: flex; justify-content: space-around; font-size: 2em;"> 003908 </div>	<div style="display: flex; justify-content: space-around; font-size: 2em;"> 724 </div>

Obrázek 7: Identifikace zapsaná ve formě digitálního čísla

Ve výstupu zpracování digitálního čísla se může objevit i znak „–“, který znamená nevyplněné digitální číslo. Chyba je označena znakem „x“ a hlášením na standardní chybový výstup.

Příklad výstupu zpracování:

–50821,176, 1.A: 2.A: 3.C: 4.D: 5.D: 6.–: 7.C: 8.–: 9.B:
 10.ACD: 11.–: 12.D: 13.B: 14.AD: 15.D: 16.–: 17.B: 18.D:
 19.D: 20.–: 21.D: 22.ABCD: 23.–: 24.D: 25.–:

V reálném testování se jako nejpřesnější ukázal algoritmus pracující na základě hledání souvislostí mezi jednotlivými body (rohy) digitálního čísla. Je odolný vůči různým intenzitám vyplnění – od obyčejné tužky až po pero, propisku či tenkou fixu.

6. Vyhodnocení výsledků a kontroly

Pro nasazení našeho systému v reálném prostředí zpracování testů v přijímacím řízení bylo rozhodnuto, že budeme skenovat pouze část pro odpovědi. Část pro identifikaci bude zadávána ručně. Systém ukládá do souboru výsledky zpracování naskenovaných odpovědí, je schopen načíst vzorové odpovědi a uložit je do souboru. Na obrázku 8 je identifikační část označena zkráceně „ID.“, odpovědní část „ODP.“ a čárový kód jako „EAN“. Náš systém na základě zadání souboru výsledků a zadání souboru správných vzorových odpovědí vyhodnotí bodovou (procentuální) úspěšnost a je schopen uložit tyto body spolu s identifikační částí čárového kódu do souboru. Za správně zodpovězenou otázku se považuje shodnost všech odpovědních prvků (obvykle tří, čtyř, resp. pěti čtverečků) se vzorovým řešením a bude započtena 1 bodem, v opačném případě 0 body. Lze nastavit i jiné bodové ohodnocení odpovědí (třeba záporné body za chybné odpovědi).

System upozorňuje obsluhu na případné automaticky zjistitelné chyby, např. nesprávně naskenovaný formulář, nečitelný – poškozený čárový kód EAN, poškozené zaměřovací čtverečky a podobně. System umožňuje obsluze kontrolovat správnost rozpoznání odpovědí.

6.1. Testování v reálném prostředí

V letech 2000, 2001 i 2002 byl systém SCAT nasazen v reálném prostředí přijímacích zkoušek na vysokou školu. V roce 2001 byl i pro testovací účely využit jako prostředek vyhodnocení písemného zkoušení na vysoké škole. Každým rokem bylo vyhodnoceno cca 30 000 testových formulářů.

6.1.1. Slabá místa systému

Časově nejnáročnější je proces samotného skenování, které je vždy omezeno rychlostí podavače skeneru. Vzhledem k cenové dostupnosti tohoto zařízení lze však použít skenerů několik a tím celý proces zpracování výrazně urychlit.

Z důvodu anonymity se nejčastěji používá pro účely přijímacích zkoušek systém, kde je fyzicky oddělena identifikační a odpovědní část. Ukázalo se, že ruční zpracování identifikační části testu obvykle nečiní žádné potíže. Náročné na logistiku může být na některých fakultách vyhodnocení relací různých oborových testů, toto však je zcela nezávislé na technologii zpracování odpovědí.

Obrázek 8: Logistika přijímacího řízení

Institut mezioborových studií
Brno

PŘIJÍMACÍ ZKOUŠKA

studijní program **Pedagogika**, obor **Sociální pedagogika**

ak. rok 2002/2003

0 143820 210515

Údaje v části „A“ (nad čarou) vyplňte čitelně hůlkovým písmem. Ústřížek na pokyn odtrhněte a odevzdejte.

Část „A“

Příjmení:

Jméno:

Rodné číslo: /

Registrační číslo:

1051

Část „B“

	A	B	C		A	B	C		A	B	C		A	B	C		A	B	C
1	☐	☐	☐	21	☐	☐	☐	41	☐	☐	☐	61	☐	☐	☐	81	☐	☐	☐
2	☐	☐	☐	22	☐	☐	☐	42	☐	☐	☐	62	☐	☐	☐	82	☐	☐	☐
3	☐	☐	☐	23	☐	☐	☐	43	☐	☐	☐	63	☐	☐	☐	83	☐	☐	☐
4	☐	☐	☐	24	☐	☐	☐	44	☐	☐	☐	64	☐	☐	☐	84	☐	☐	☐
5	☐	☐	☐	25	☐	☐	☐	45	☐	☐	☐	65	☐	☐	☐	85	☐	☐	☐
6	☐	☐	☐	26	☐	☐	☐	46	☐	☐	☐	66	☐	☐	☐	86	☐	☐	☐
7	☐	☐	☐	27	☐	☐	☐	47	☐	☐	☐	67	☐	☐	☐	87	☐	☐	☐
8	☐	☐	☐	28	☐	☐	☐	48	☐	☐	☐	68	☐	☐	☐	88	☐	☐	☐
9	☐	☐	☐	29	☐	☐	☐	49	☐	☐	☐	69	☐	☐	☐	89	☐	☐	☐
10	☐	☐	☐	30	☐	☐	☐	50	☐	☐	☐	70	☐	☐	☐	90	☐	☐	☐
11	☐	☐	☐	31	☐	☐	☐	51	☐	☐	☐	71	☐	☐	☐	91	☐	☐	☐
12	☐	☐	☐	32	☐	☐	☐	52	☐	☐	☐	72	☐	☐	☐	92	☐	☐	☐
13	☐	☐	☐	33	☐	☐	☐	53	☐	☐	☐	73	☐	☐	☐	93	☐	☐	☐
14	☐	☐	☐	34	☐	☐	☐	54	☐	☐	☐	74	☐	☐	☐	94	☐	☐	☐
15	☐	☐	☐	35	☐	☐	☐	55	☐	☐	☐	75	☐	☐	☐	95	☐	☐	☐
16	☐	☐	☐	36	☐	☐	☐	56	☐	☐	☐	76	☐	☐	☐	96	☐	☐	☐
17	☐	☐	☐	37	☐	☐	☐	57	☐	☐	☐	77	☐	☐	☐	97	☐	☐	☐
18	☐	☐	☐	38	☐	☐	☐	58	☐	☐	☐	78	☐	☐	☐	98	☐	☐	☐
19	☐	☐	☐	39	☐	☐	☐	59	☐	☐	☐	79	☐	☐	☐	99	☐	☐	☐
20	☐	☐	☐	40	☐	☐	☐	60	☐	☐	☐	80	☐	☐	☐	100	☐	☐	☐

0 143820 210515

Jednu správnou odpověď zaškrtněte takto: ☒

Výsledkový formulář vyplňte přesně podle pokynů zkoušejícího!

Pedagogická fakulta MU v Brně

Přijímací písemná zkouška na obor technická výchova

prezenční studium

červen 2002

Údaje v části nad čarou vyplňte čitelně hůlkovým písmem, oddělte a ústřížek na pokyn odevzdejte.

Příjmení: jméno:

rodné číslo: /

číslo přihlášky: 1 3 0 1

A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D				
1	☐	☐	☐	☐	11	☐	☐	☐	☐	21	☐	☐	☐	☐	31	☐	☐	☐	☐
2	☐	☐	☐	☐	12	☐	☐	☐	☐	22	☐	☐	☐	☐	32	☐	☐	☐	☐
3	☐	☐	☐	☐	13	☐	☐	☐	☐	23	☐	☐	☐	☐	33	☐	☐	☐	☐
4	☐	☐	☐	☐	14	☐	☐	☐	☐	24	☐	☐	☐	☐	34	☐	☐	☐	☐
5	☐	☐	☐	☐	15	☐	☐	☐	☐	25	☐	☐	☐	☐	35	☐	☐	☐	☐
6	☐	☐	☐	☐	16	☐	☐	☐	☐	26	☐	☐	☐	☐	36	☐	☐	☐	☐
7	☐	☐	☐	☐	17	☐	☐	☐	☐	27	☐	☐	☐	☐	37	☐	☐	☐	☐
8	☐	☐	☐	☐	18	☐	☐	☐	☐	28	☐	☐	☐	☐	38	☐	☐	☐	☐
9	☐	☐	☐	☐	19	☐	☐	☐	☐	29	☐	☐	☐	☐	39	☐	☐	☐	☐
10	☐	☐	☐	☐	20	☐	☐	☐	☐	30	☐	☐	☐	☐	40	☐	☐	☐	☐

Právě jednu správnou odpověď zaškrtněte takto: ✕
Výsledkový formulář vyplňte přesně podle pokynů zkoušejícího!

7. Závěr

7.1. Probíhající vývoj

Vývoj systému SCAT se nezastavil. Dále probíhá řešení rozpoznávání ručně psaných čísel. Testují se nové algoritmy, zkoušejí se nové přístupy, aby se eliminovala chybovost při analýze čísel. Jedna z možností je kontrolovat ihned při skenování (resp. zpracování) relace typu *rodné číslo – číslo přihlášky* a připravit uživatelské rozhraní pro odpovídající zásahy obsluhy.

Jinou oblastí vývoje, ve které se také pokračuje, je hledání nových skenovacích zařízení. Nové skenery umožňují zajímavé zrychlení. Skener HP 7450 C skenuje cca 10 stran za minutu, skener Brother MFC9180 cca 8 stran za minutu. Vývoj možná povede i do oblasti digitální fotografie – tedy formulář by se již neskenoval po řádcích, ale fotografoval jako jeden celek. To by mohlo přinést opět výrazné urychlení celkového zpracování. Dále chceme systém rozšířit o možnost rozpoznávání čísel [5] a optimalizovat nastavení prahovacích konstant pomocí algoritmů z [8].

7.2. Shrnutí výsledků

Naším cílem bylo prokázat, že lze s využitím současných, běžně dostupných zařízení automatizovat vytváření, zpracování a vyhodnocení písemných testů. Díky automatickému způsobu převodu souřadnic přímo z elektronického obrazu formuláře, zaměření a srovnání naskenovaného obrazu jsme dosáhli požadovanou přesnost vyhodnocování. Vypracovali jsme způsob rozpoznání digitálních čísel. Navrhli jsme a vytvořili prostředí a logistiku vyhodnocování tak, že vznikl robustní a flexibilní systém. Systém byl úspěšně použit na zpracování téměř sta tisíce formulářů.

Reference

- [1] Adriana Benadiková, Štefan Mada, a Stanislav Weinlich. *Čárové kódy, automatická identifikace*. Grada Publishing, 1994.
- [2] Miroslav Hrad. Automatizace vytváření, zpracování a vyhodnocení písemných testů. Diplomová práce, Masarykova univerzita v Brně, Fakulta informatiky, duben 2001.
- [3] Donald Erwin Knuth. *The T_EXbook*. Addison-Wesley, Reading, Massachusetts, 1984.
- [4] Miroslav Maťák. Automatizované rozpoznávání testů. Bakalářská práce, Masarykova univerzita v Brně, Fakulta informatiky, 1999.
- [5] Roman Mikolaj. Gravitačný algoritmus rozpoznávania znakov. Diplomová práce, MFF UK, Bratislava, katedra informatiky, březen 1998.

- [6] Petr Olšák. The EAN barcodes by T_EX. *TUGboat*, 15(4):459–464, 1994.
- [7] Jordi Saludes. Fast and secure multiple-option tests. *TUGboat*, 17(3):310–319, 1996.
- [8] Michail I. Schlesinger a Václav Hlaváč. *Deset přednášek z teorie statistického a strukturního rozpoznávání*. Vydavatelství ČVUT, Praha, 1999.
- [9] Petr Sojka. An Experience from a Digitization Project. *Cahiers GUTenberg*, (28–29):276–281, březen 1998.
- [10] Petr Sojka. Publishing Encyclopaedia with Acrobat using T_EX. V *Towards the Information-Rich Society. Proceedings of the ICCC/IFIP conference Electronic publishing '98*, strany 217–222, Budapest, Hungary, duben 1998. ICC Press.
- [11] Petr Sojka, Rudolf Červenka, a Martin Svoboda. T_EX for database publishing. V Jiří Zlatuška, editor, *Proceedings of the 7th European T_EX Conference, Prague, 1992*, strany 53–58, Brno, září 1992. Masarykova univerzita v Brně.

Summary: Batch Typesetting and Scanning of Forms

The paper describes SCAT (SCAnning Test) technology for batch typesetting and scanning of forms on a desktop PC. SCAT has been used for typesetting and scanning of almost 100,000 pieces of forms at the Masaryk University and other institutions in the last three years. The system is based on T_EX and other Open Source tools. The design of the system that aims at minimal error rate (using bar code technology) is presented. Speed, security and maximal automatization aspects of SCAT are discussed.

Miroslav Hrad, Petr Sojka
Masarykova univerzita v Brně
Fakulta informatiky, Botanická 68a, 602 00 Brno
Email: xhrad@fi.muni.cz sojka@fi.muni.cz
a
AMD, v. o. s., Návrší Svobody 26, 623 00 Brno
Email: psojka@mistral.cz

Program TeX je systém pro velmi kvalitní počítačovou sazbu vyznačující se přenositelností zdrojového kódu mezi platformami i v čase. Portable Document Format (PDF) je formát pro elektronické publikování. Tento příspěvek ukazuje možnosti elektronického publikování v PDF uživateli zvyklými na TeX.

1. Troška historie

Není mým cílem zde rozebírat historii programů TeX, dvips a pdfTeX stejně jako historii formátů PostScript (dále jen PS) a PDF. Kdo by chtěl nahlédnout do podmínek vzniku těchto softwarových nástrojů počítačové sazby, tomu doporučuji skvělý článek Philipa Taylora [16].

2. Co přináší PDF oproti PostScriptu?

Koncem osmdesátých let byl PS prakticky standardem pro popis vzhledu stránek. Přesto firma Adobe začala od roku 1990 pracovat na specifikaci jiného standardu. V té době již bylo zřejmé, že komunikace a výměna dokumentů prostřednictvím sítí je mnohem flexibilnější než práce s dokumenty v tištěné podobě. Chyběl však formát, který by v sobě spojoval typografickou kvalitu PS a

¹Článek byl poprvé publikován ve sborníku Jan Kasprzak, Petr Sojka: SLT 2002, KONVOJ 2002, ISBN 80-7302-043-2, na stranách 69–77. Publikováno se souhlasem vydavatele a autorů.

hypertextové a multimediální možnosti elektronického publikování. Tyto okolnosti vedly ke vzniku PDF. Pokusme se charakterizovat jeho vlastnosti [15, 18].

1. Začneme tím, v čem je rozdílnost minimální. Je to *kreslicí model*, který se v PS osvědčil, umožňující komplexní grafický popis stránky. Oba standardy používají stejnou afinní transformaci pro převod mezi uživatelským (na výstupním zařízení nezávislým) souřadným systémem a systémem výstupního zařízení. Používat lze bitmapovou i vektorovou grafiku. Základním kamenem pro vektorový popis textu i grafiky jsou kubické Bézierovy křivky. Volit lze z různých druhů tahů, způsobů napojení a rastrování rohů. Uzavřené křivky lze vyplňovat vzory. Používat lze různé barevné modely (RGB, CMYK, CIE). Verze PDF 1.4 umí pracovat s průhledností.
2. Velké rozdíly najdeme v *jazyku* dokumentu. Zatímco PS je plnohodnotný programovací jazyk zásobníkového typu, PDF neobsahuje procedury, řídicí struktury a proměnné. Jde o seznam grafických operací. V PS se operátory vyšší úrovně vyjadřují přímo jazykem PS z elementárních grafických příkazů. V PDF musí být tyto vyšší operace přímo implementovány aplikací zpracovávající dokument. Důsledkem je efektivnější zpracování a zobrazování PDF dokumentu. Na druhou stranu je snížena flexibilita vyjádření. Například balík PSTricks [23] expanduje kreslicí příkazy do PS kódu. Tento kód je relativně složitý a obsahuje programové konstrukce, např. cykly. Předpokládá se, že nakonec bude dokument zpracován nějakým PS RIPem. Protože např. pdfTeX takový RIP neobsahuje, nelze tento balík v pdfTeXu použít. Řešením je vložit takový obrázek do zvláštního dokumentu, ten zpracovat standardním TeXem, pomocí dvips -E vyrobit encapsulated PS, ten převést skriptem epstopdf využívajícím GhostScript do PDF a načíst do našeho dokumentu. O zautomatizování tohoto procesu se snaží balík PDFTRICKS [12]. Spouští při tom externí programy přímo z TeXu pomocí konstrukce `\write18{command}`. Tuto možnost nabízí např. web2c instalace TeXu. V našem případě spustíme následující tři programy:

```
\write18{(la)tex tmp.tex; dvips -E tmp.dvi; ps2pdf tmp.ps}
```

3. PS byl původně textový kód bez jakékoliv pevné struktury. Dodatečně vydaná technická zpráva [7] doporučuje strukturní konvence PS (Document Structuring Conventions, DSC) dokumentů a značkování pomocí komentářů. Pak lze oddělit jednotlivé stránky dokumentu bez nutnosti interpretovat PS. Na přítomnosti těchto komentářů je založen balík PSUTILS [2]. Náhodný přístup k libovolné části dokumentu řeší až pevná struktura PDF. Dokument PDF se skládá z objektů, jejichž umístění v dokumentu popisuje tabulka křížových referencí. Každá stránka je také samostatným objektem a krom sazby textu obsahuje i seznam svých zdrojů (fonty, anotace), informace o velikosti stránky ap. Díky tomu stačí pro zobrazení stránky interpretovat jen ty objekty, které tvoří tuto stránku. Také je snadné s do-

kumentem manipulovat, měnit pořadí stránek a jejich velikost. Také můžeme snadno vložit stránku jako obrázek do jiného dokumentu (toho využívá balík pdfpages [9]). Další užitečnou vlastností PDF je možnost jeho jednorůchodového generování. Je to zajištěno systémem nepřímých odkazů na objekty. Například potřebujeme-li zadat uvnitř objektu A jeho velikost, kterou před jeho uzavřením ještě neznáme, uvedeme v něm pouze nepřímý odkaz na objekt B. Objekt B bude obsahovat velikost objektu A. Příklad nejjednoduššího PDF dokumentu ukazuje obrázek 1.

<pre>%PDF-1.3 3 0 obj << /Length 101 >> stream 1 0 0 1 91.925 759.927 cm BT /F51 9.963 Tf 0 0 Td[(Hallo)-250(PDF!)]TJ 211.584 -654.747 Td[(1)]TJ ET endstream endobj 2 0 obj << /Type /Page /Contents 3 0 R /Resources 1 0 R /MediaBox [0 0 595.276 841.89] /Parent 5 0 R >> endobj 1 0 obj << /Font << /F51 4 0 R >> /ProcSet [/PDF /Text] >> endobj 5 0 obj << /Type /Pages /Count 1 /Kids [2 0 R] >> endobj</pre>	<pre>6 0 obj << /Type /Catalog /Pages 5 0 R >> endobj 7 0 obj << /Producer (pdfTeX-0.14h) /Creator (TeX) /CreationDate (D:20021104154200) >> endobj xref 0 8 0000000004 65535 f 0000000278 00000 n 0000000167 00000 n 0000000009 00000 n 0000000000 00000 f 0000000346 00000 n 0000000403 00000 n 0000000452 00000 n trailer << /Size 8 /Root 6 0 R /Info 7 0 R >> startxref 546 %%EOF</pre>
--	--

Obrázek 1: Nejjednodušší PDF Hallo PDF připravené pdf_TE_Xem pomocí kódu `\pdfcompresslevel=0 \font\tenrm=tir\tenrm Hallo PDF!\end.`

4. PDF podporuje standardní kompresní formáty (JPEG, CCITT Group 3 a 4, LZW a Run Length). Jejich dekompresi musí zajistit aplikace zpracovávající PDF. Kromě výrazného zmenšení dokumentu (důležité pro přenos po síti) a nezmenšeném konfortu pro uživatele (žádné odzipovávání) je díky této podpoře snadné i vkládání komprimovaných bitmapových obrázků (TIFF, PNG, JPEG). Nevýhodou ovšem je, že nelze snadno na úrovni $\text{T}_{\text{E}}\text{X}$ ových maker vyhledávat a měnit řetězce textu tak, jak to například dělá balík PSFRAG [3] pro editaci popisek obrázků často generovaných

programy s méně variabilním grafickým výstupem (fonty, matematická sazba). I zde by se pro pdf $\text{\TeX}$ hodila makra podobná PDFTRICKS pro automatické externí zpracování obrázku přes PS.

5. Z běžných formátů fontů umožňuje PS pracovat hlavně s Type 1 fonty. Bitmapové fonty rastrované METAFONTEM jsou vkládány ve formátu Type 3. PDF zvládá oba tyto formáty, a navíc umí pracovat s True Type formátem. Musíme jen vyrobit $\text{\TeX}$ ovou metriku pomocí dvojice programů `ttf2afm` a `afm2tfm`.

Každá aplikace PDF musí obsahovat 14 základních fontů. Tyto fonty se standardně nevkládají do dokumentu. Bohužel, nemůžeme se spoléhat na to, že metriky těchto fontů jsou ve všech aplikacích stejné. Liší se dokonce verze od verze Adobe Acrobatu. Chceme-li mít jistotu, že se náš dokument všude vysází stejně, musíme i tyto fonty vložit. Zařídíme to v konfiguračním souboru fontů `psfonts.map` odkázaném v konfiguračním souboru pdf $\text{\TeX}$ u `pdf $\text{\TeX}$ .cfg`. Na řádce příslušného fontu provedeme dvě změny: a) změníme název fontu a b) vložíme font pomocí dvouznaku `<<`. Pro příklad uvedeme řádek pro Times-Roman:

```
ptmr8r Times-RomanEmbed <8r.enc << utmr8a.pfb
```

6. PDF podporuje předtiskovou úpravu dokumentu (hranice media, stránky a zobrazení, tisk ořezových značek, barevné separace, ...).
7. Hlavním přínosem PDF bylo jeho interaktivní rozšíření využitelné při elektronickém zobrazení dokumentu. Jde o hypertextové prvky usnadňující orientaci (odskoky, záložky, náhledy), dále zpřístupnění videa a zvuku a možnost vyplňování a zasílání formulářů.
8. V PDF existuje podpora pro indexaci a vyhledávání slov.
9. Dokument je možné také modifikovat a to pouze připojením změněných nebo nových objektů na konec dokumentu spolu s novou tabulkou křížových referencí. Původní část dokumentu tedy zůstává nedotčena. Snadno se pak můžeme vrátit k předchozí verzi.
10. Formát byl navržen tak, aby byl systémově nezávislý, šlo jej rozšířit o nové technologie se zachováním zpětné kompatibility.

Po přečtení tohoto výčtu můžeme nabýt dojmu, že PDF má vše, co potřebujeme pro potřeby tisku, elektronického publikování nebo prezentací. Specifikace formátu je však věc jedna a jeho implementace a dostupnost vhodných nástrojů věc druhá. Sama Adobe se snažila prosadit PDF tím, že zdarma poskytla nástroj Adobe Acrobat Reader pro prohlížení a tisk PDF dokumentů na různých platformách. Díky tomu se jí také *de facto* podařilo prosadit PDF jako standard elektronického dokumentu. Přesto existují problémy, které používání formátu ztěžují. Napadají mě tyto:

1. Volně šířený Acrobat Reader ani plný Acrobat umožňující modifikovat PDF dokument neimplementuje celou specifikaci formátu (uvedme např. nastavení hlasitosti v `/Sound` anotaci, grafické znázornění `/Sound` anotace

podle jejího stavu, specifikace `/Sound` anotace podle URL, vložení video souboru pro `/Movie` anotaci interně do dokumentu). Kromě toho existují části specifikace systémově závislé (spouštění externích programů, zvukové a video anotace).

2. Implementace Acrobatu obsahuje odlišnosti od specifikace (chyby). Například obsahuje-li encoding vektor fontu vynechané nedefinované znaky, tiskne Acrobat verze pět chybně. Dále aplikování transformační matice je podle specifikace možné několika způsoby, Acrobat však rozumí jen některým. Že software obsahuje chyby, je vcelku pochopitelné, co je však pobuřující, je postoj Adobe k jejich odstraňování (a bohužel typický pro velkou komerční firmu). Znalým uživatelem pohrdají, chybu většinou nepřiznají, a pokud ano, její odstranění trvá léta.
3. Podle specifikace musí mít každý interpret PDF k dispozici 14 základních fontů. Je to proto, aby malé dokumenty zbytečně nenarůstaly vkládáním fontů. Co si však myslet o tom, že Adobe přibaluje k různým verzím Acrobatu tyto fonty s různými metrikami?
4. Adobe preferuje v některých oblastech podivné strategie vedoucí k mizerné kvalitě dokumentu. Příkladem je záměrně špatná rasterizace Type 3 fontů na obrazovku Acrobatem.
5. Ačkoliv je Acrobat Reader distribuován pro různé platformy, na některé je k dispozici s výrazným časovým skluzem (Linux) a pro jiné se vývoj zastavil (OS/2).

Přenositelnost dokumentu není z těchto důvodů naplněna. Je pravda, že pokud se omezíme na jednoduchý hypertext bez videa a externích volání, vložíme všechny fonty do dokumentu a přizpůsobíme svůj program tak, aby negeneroval chybu v prohlížeči, i když je syntakticky správně, slušné záruky o přenositelnosti dokumentu dosáhneme. Problém nevidím ve specifikaci PDF, ale v implementaci této specifikace. Konkurence z oblasti open-source (GhostScript, xpdf) udělala velký kus práce, ale i tak je v rozsahu pokrytí specifikace za Adobe. Chybí hlavně podpora prezentačního módu, videa a java skriptu. Nechtěje to výzva pro open-source programátory.

3. Tři cesty od $\text{T}_{\text{E}}\text{X}$ u k PDF

Existují v podstatě tři cesty, jak vytvořit PDF dokument z $\text{T}_{\text{E}}\text{X}$ ového zdrojového kódu:

1. Generovat jej přímo ze zdrojového kódu. Umí to modifikace $\text{T}_{\text{E}}\text{X}$ u zvaná `pdf $\text{T}_{\text{E}}\text{X}$` . Program vznikl na Masarykově univerzitě jako magisterská a doktorská práce Hàn Thê Thànha [19, 20].
2. Standardním $\text{T}_{\text{E}}\text{X}$ em vytvořit DVI a zkonvertovat jej do PDF, například programem `dvipdf` [22].

3. Vytvořit z DVI PS a ten pak převést do PDF pomocí GhostScriptového skriptu `ps2pdf` nebo komerčním Adobe Distillerem.

Vkládání interaktivních prvků je v pdfT_EXu umožněno zavedením nových primitiv. Zbylé dva způsoby využívají značkování T_EXovým primitivem `\special{...}` a PS operátorem `/PDFMark`.

S programem `dvipdf` nemám osobní zkušenost, takže jej hodnotit nebudu. Ale je to nejméně používaná cesta. Kvalita převodu pdfT_EXem a přes PS je výborná. Distiller měl oproti GhostScriptu navrch, ale v poslední době se tento rozdíl stírá. Vyplatí se mít GhostScript nejnovější verze. Distiller umožňuje optimalizovat dokument z hlediska jeho použití (osvit, tiskárna, obrazovka). Pokud by chtěl tyto zařízení rozlišit uživatel pdfT_EXu, musel by ručně přizpůsobit každý obrázek (rozlišení, barevný model), probrat specifikaci PDF a nastavit patřičné parametry.

Při generování PDF je dobré se vyhnout Type 3 fontům, tj. fontům generovaných METAPOSTem. Dnes již většina METAPOST fontů existuje v dobré kvalitě i jako Type 1. Pro první a třetí způsob generování PDF se nastavení PS fontů zajistí na úrovni konfiguračních souborů (ve web2c instalaci je to implicitní nastavení `pdftex.map` uvedený v `pdftex.cfg`, respektive `config.pdf` pro DVI ovladač `dvips` spuštěný s parametrem `-Ppdf`).

Výhody pdfT_EXu oproti výrobě PDF přes DVI jsou následující:

- Můžeme vkládat bitmapovou grafiku ve formátech JPEG, PNG a TIFF. Při klasickém překladu do PS jsme většinou omezeni převodem obrázků do EPS, čímž jejich velikost velmi naroste (běžně 10×) a naroste tak i velikost výsledného PS. V případě jednostránkových plakátů tak běžně jde o PS soubor veliký stovky MB. Také musíme započítat paměťovou a časovou režii při převodu obrázků.
- Můžeme používat TrueType fonty.
- Můžeme využívat mikrotypografická rozšíření, kterými se zabývá Thàn-hova disertační práce [20]. Jde o znaky visící z bloku textu a horizontální zvětšování/zmenšování znaků u řádků s vysokou hodnotou škaredosti (badness). Jejich cílem je kompaktnější a homogennější globální vzhled odstavce při okem nepostřehnutelných lokálních odchylkách. Obě techniky jsou implementovány do T_EXového optimalizačního algoritmu zlomu odstavce. Více viz v sekci 4.1 v bodě 4.
- Lze uložit aktuální bod sazby a znovu jej nastavit. Tato funkce je užitečná pro sazbu blokových diagramů nebo podbarvení odstavců s možným stránkovým zlomem a plovoucími objekty.

V další sekci se zaměřím na přímé generování PDF pdfT_EXem.

4. Program pdfTeX

Program pdfTeX má svoji vlastní elektronickou diskusní skupinu vedený v anglickém jazyce. Její adresa je `pdfTeX@tug.org` a její archiv lze nalézt na `http://tug.org/pipermail/pdfTeX/`. Přihlásit se lze na stránce `http://tug.org/mailman/listinfo/pdfTexsvětového Sdružení uživatelů TEXu TUG`.

4.1. Nová primitiva

Jak jsme již uvedli, pdfTeX umožňuje využívat možností PDF pomocí nových primitiv. Jejich popis (bohužel neúplný) uvádí pdfTeX manual [21], úplný seznam čtenář najde v [17]. My si je zde funkčně roztřídíme bez požadavku na úplnost:

1. Hlavní přepínač `\pdfoutput`, jehož kladná hodnota přepíná výstup z DVI do PDF. Jeho nastavení má význam jen před výstupem první stránky pomocí `\shipout`. Tento registr se testuje, chceme-li automaticky rozlišit, zda je dokument zpracováván klasickým T_EXem nebo pdfT_EXem s výstupem do DVI či pdfT_EXem s výstupem do PDF:

```
\newif\ifPDF
\ifx\pdfoutput\undefined
\else\ifnum\pdfoutput>0 \PDFtrue\fi
\fi
```

2. Rozměrové parametry (např. `\pdfpagewidth`, `\pdfhorigin`). Pozor! Pokud nastavíte `\pdfhorigin` nebo `\pdfvorigin` na nulu, pdfT_EX ji změní na 1 in. Proto v takovém případě použijte např. `\pdfhorigin=1sp`).
3. Parametry dokumentu (komprese `\pdfcompresslevel`, numerická přesnost `\pdfdecimaldigits`, informační údaje `\pdfinfo`, způsob zobrazování dokumentu [15, str. 83] `\pdfcatalog`) a jednotlivých stránek [15, str. 88] (`\pdfpagesattr`, `\pdfpageattr`).
4. Mikrotypografická rozšíření.

- Visící znaky (protrude characters). Každému znaku lze přiřadit hodnotu, o kolik má tento znak přesahovat levý (`\lpcode`) a pravý (`\rprcode`) okraj bloku textu. Syntax obou primitiv je stejná jako u primitiva `\catcode`. Hodnota 1000 znamená převis o 1 em. Visící znaky zapínáme kladnou hodnotou `\pdfprotrudechars`. Je-li jinak menší jak 2, pak se řádky nalámou standardním algoritmem T_EXu a následně se doplní převis. Je-li 2 a více, optimalizuje se zlom i s aktuálními hodnotami visících znaků. Rozhoduje hodnota tohoto registru při uzavření odstavce. Příklad: chceme-li vpravo přesah tečky o 5 % em a rozdělovacího znaménka o 8 % em v aktuálním fontu, napíšeme:

```
\pdfprotrudechars=2
```



```
\rptcode\font'\.=50
\rptcode\font\hyphenchar\font=80
```

Mezi visící znaky pečliví typografové zařazovali interpunkční znaménka (tečky, čárky, uvozovky, rozdělovací znaménka) pro jejich přílišnou světlost narušující vertikální hranu textu. Příklad použití je v tomto odstavci.

- Horizontální zvětšování/zmenšování znaků (font expansion). Někdy je obtížné zlomit odstavec tak, aby neobsahoval velké díry mezi slovy. Často se to stává při sazbě do úzkého sloupce. Horizontální zvětšování/zmenšování znaků nám přidává další stupeň volnosti pro rovnoměrné vyplnění řádky. Každému fontu přidělíme maximální hodnoty roztažení/stažení pomocí primitivu `\pdffontexpand`. Také se zde nastavuje krok, protože stažení i roztažení se děje diskrétně. Změna velikosti znaku musí však být prováděna s citem, aby čtenář nic nepoznal, i když se pod sebou sejde nejvíce protažená a smrsknutá řádka. Doporučené hodnoty jsou okolo 2 %. I zde máme možnost ovlivnit, o kolik se může jedno písmeno roztáhnout/stáhnout vůči globálním hodnotám pomocí hodnoty `\efcode`. Přepínač se jmenuje `\pdfadjustspacing` a i zde má tři polohy stejně jako u visících znaků. Ke každému takto použitému fontu musíme připravit metriky a jde-li o bitmapové (METAPOST) fonty, musíme zajistit i vygenerování těchto bitmap. Syntax názvu fontů je následující: `fontname-shrink.tfm` nebo `fontname+stretch.tfm`. Příklad:

```
\newcount\N
\loop\efcode\font\N=1000\advance\N by 1
\ifnum\N<256 \repeat
\pdfadjustspacing=2
\pdffontexpand\font 20 20 5 1000
```

Číselné hodnoty u `\pdffontexpand` mají následující význam: roztažení, stažení, krok a měřítko.

5. Primitiva pracující s fonty (`\pdffontname`, `\pdffontobjnum`, `\pdfincludechars`).
6. Parametry bodu sazby (`\pdfsavepos`, `\pdflastxpos`, `\pdflasttypos`).
Není zatím dokumentováno.
7. Primitiva vkládající obrázky [24] či boxy sazby (`\pdfximage`, `\pdfrefximage`, `\pdflastximage`, `\pdfimageresolution`, `\pdfxform`, `\pdfrefxform`, `\pdflastxform`). Málo známý je registr `\pdflastximagepages`, který udává počet stran PDF dokumentu načteného pomocí `\pdfximage`.

8. Primitiva vytvářející hypertextové odkazy [25] (odkazy `\pdfstartlink`, `\pdfendlink`, `\pdflinkmargin`, doskoky `\pdfdest`, `\pdfdestmargin`, záložky `\pdfoutline` a zřetězení článku `\pdfthread`, `\pdfstartthread`, `\pdfendthread`).
 9. Primitiva zařazující anotace, např. zvukové soubory a videa [26] (`\pdfannot` a `\pdflastannot`).
 10. Primitiva vkládající obecné objekty včetně proudů [26] (`\pdfobj`, `\pdfrefobj`, `\pdflastobj`,) a obecné grafické instrukce (`\pdfliteral`).
- Příklady použití:

- Transformace sazby.

$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} a & b & 0 \\ c & d & 0 \\ e & f & 1 \end{bmatrix}$$

```

R      \pdfliteral{q 0 1 -1 0 0 0 cm}%
      \smash{R}%
      \pdfliteral{Q}%

```

Aby se nám nerozjely body sazby $\text{\TeX}$ u a PDF, musíme dát obsah transformace do boxu nulové velikosti nebo $\text{\TeX}$ ovými primitivami zajistit posun sazby o tolik, o kolik se posune včetně PDF transformace. $\text{\TeX}$ sám totiž o PDF transformaci nemůže nic vědět.

- Průhlednosti (od verze PDF 1.4.

```

\def\HR#1#2{\vrule width.5cm height#1ex depth#2ex}%
\let\ex=\expandafter \let\im=\immediate
\im\pdfobj{<< /Type /ExtGState /ca 1
           /BM /Normal /AIS false >>}%
\edef\NmTrA{\the\pdflastobj\space 0 R }%
\im\pdfobj{<< /Type /ExtGState /ca .8
           /BM /Screen >>}%
\edef\NmTrB{\the\pdflastobj\space 0 R }%
\im\pdfobj{<< /Tr0 \NmTrA /Tr1 \NmTrB >>}%
\edef\NmTr{/ExtGState \the\pdflastobj\space 0 R }%
\ex\global\ex\pdfpageresources\ex{\NmTr}%

\pdfliteral{1 .8 .8 rg /Tr1 gs}%
\HR{2}{1}\hskip-0.25cm\HR{2.5}{.5}%
\pdfliteral{/Tr0 gs 0 g}%

```

- Změna barvy.

Gray Black	<code>\pdfliteral{0.8 g}Gray</code>
	<code>\pdfliteral{0 g}Black</code>
RGB Black	<code>\pdfliteral{0 0 1 rg}RGB</code>
	<code>\pdfliteral{0 g}Black</code>
CMYK Black	<code>\pdfliteral{0 0 1 0 k}CMYK</code>
	<code>\pdfliteral{0 g}Black</code>

11. Podpora UNICODE. Zatím není dokumentováno.
12. Parametry programu pdfTeX (`\pdf texversion`, `\pdf texrevision`).

4.2. Makrobalíky

Velice šikovným nástrojem je balík `thumpdf.sty` od Heiko Oberdieka, který perlovým skriptem ve spolupráci s GhostScriptem vytvoří a vloží do dokumentu *náhledy stránek*. Je napsaný v plain TeXu. Použití je nesmírně jednoduché:

1. Vlož makra do dokumentu: `\input thumpdf.sty`
2. PřeteXuj.
3. Spust skript Perlu: `thumpdf file_without_ext`
4. PřeteXuj.

Uživatelé L^AT_EXu mohou využít balíku Sebastiana Rahtze `hyperref` [13, 10]. Nejenomže zjednodušuje vkládání *hypertextových odkazů* a automaticky generuje odkazy ze standardních L^AT_EXových referenčních mechanismů. Umožňuje dokonce učinit kód nezávislým na způsobech generování PDF popsaných v sekci 3. Použijeme-li parametr `pdf tex`, najdeme podporu v balících `color` od Davida Carlisleho, `graphicx` od Davida Carlisleho a Sebastiana Rahtze a `geometry` pro nastavení rozměrů dokumentu od Hideo Umekiho. Dále jmenujme již zmíněné `pdftricks` a `pdfpages`.

Myslím, že oblast, elektronického publikování, jehož rozvoj teprve nastane, je vyplňování *formulářů*. I zde existuje velmi silný nástroj pro L^AT_EX. Jmenuje se AcroTeX a jeho autorem je D. P. Story [14]. Tento balík obsahuje i podporu Javascriptu.

A snad nejsilnější podporu pro elektronické publikování a prezentace má balík CONTeXt Hanse Hageny. Je to nástroj využívající to nejlepší ze tří různých programů: sazbu pdfTeXu, vektorovou grafiku METAPOSTu a výpočty, textové manipulace a řízení procesu Perlu. Od letošního roku je konečně k dispozici výborná dokumentace v angličtině [5, 6].

5. Prezentace pdfTeXem

V této části budou ukázány možnosti využití PDF při prezentacích včetně odkazů na nejpoužívanější prezentační makrobalíky. O CONTeXtu jsem se již zmínil. Originální varianty Hagenových prezentací přináší dokument [4]. Seznam prezentačních balíků pro L^AT_EX uvádí prezentace [1]. Velmi oblíbený je pdfScreen od C. Radhakrishnana [11]. Umí paralelně vytvářet prezentace a textový dokument. Inkrementální přírůstky textu na stránce zvládá balík TeXPower od Stephana Lehmké [8].

6. Závěr

Formát PDF je standardem pro elektronické publikování a je vhodný i pro data-prezentace. Uživatelé připravující dokumenty $\text{\TeX}$ em si mohou vybrat několik cest, jak připravit velmi kvalitní PDF. Usnadní jim v tom dnes již silná podpora makrobálků. Po prudkém rozvoji v této oblasti koncem devadesátých let se situace stabilizuje. Týká se to i programu $\text{pdf\TeX}$, který přináší několik technologických vylepšení. Největší problémy dnes nenastávají na straně vytváření PDF, ale v jeho zobrazování a tisku. Neexistuje přenositelný nástroj na zpracování méně obvyklých prvků specifikace PDF.

Reference

- [1] David M. Allen. $\text{L\AA T\TeX}$ presentation packages, únor 2002. <http://www.ms.uky.edu/~allen/presentations.pdf>.
- [2] Angus Duggan. Psutils. Free program package.
- [3] Michael C. Grant, David Carlisle. *The PSfrag System, version 3*, 1998.
- [4] Hans Hagen. The $\text{ConT\TeX t}$ presentation styles, 2001. <http://www.tug.org/tug2001/authors/presentations/hagen/hagenIIIsides.pdf>.
- [5] Hans Hagen. *ConT\TeX t the manual*. <http://www.pragma-ade.com/>, listopad 2001.
- [6] Hans Hagen. *META\FUN*. <http://www.pragma-ade.com/>, leden 2002.
- [7] Adobe Systems Incorporated. PostScript language document structuring conventions specification. Technická zpráva 5001, září 1992.
- [8] Stephan Lehmke. $\text{T\TeX}$ Power package. Home page: <http://texpower.sourceforge.net>.
- [9] Andreas Matthias. *The pdfpages Package*, 2002. Email: amat@kabsi.at.
- [10] Heiko Oberdiek. *PDF information and navigation elements with hyperref, pdfT\TeX, and thumbpdf*, 1999. $\text{TeXLive6}:\backslash\text{\$TEXMF/doc/latex/hyperref/slides.pdf}$.
- [11] C. V. Radhakrishnan. *Pdftscreen manual*, 2002. <http://ftp.cstug.cz/pub/tex/CTAN/macros/latex/contrib/supported/pdftscreen/manual-screen.pdf>.
- [12] C. V. Radhakrishnan, C. V. Rajagopal, a Chambert-Loir Antoine. *Trivial experiments with PSTricks manipulation*, září 2001. <http://ftp.agh.edu.pl/pub/tex/macros/latex/contrib/supported/pdftricks/manual.pdf>.
- [13] Sebastian Rahtz. *Hypertext marks in L\AA T\TeX: the hyperref package*, 1998. $\text{TeXLive6}:\backslash\text{\$TEXMF/doc/latex/hyperref/manual.pdf}$.
- [14] D. P. Story. The AcroT\TeX eDucation Bundle. Home page: <http://www.math.uakron.edu/~dpstory/webeq.html>.

- [15] Adobe Systems Incorporated. *PDF reference manual, v. 1.4*, second edition, 2000. <http://partners.adobe.com/asn/developer/acrosdk/DOCS/PDFRef.pdf>.
- [16] Philip Taylor. Computer typesetting or electronic publishing? New trends in scientific publication. V *Zpravodaj Československého sdružení uživatelů T_EXu*, number 1–4, strany 61–89, 1995.
- [17] Hàn Thế Thành. Seznam nových primitiv pdfT_EXu. Soubor: <http://www.fi.muni.cz/~thanh/download/pdftex/syntax.txt>.
- [18] Hàn Thế Thành. Alternativní výstup programu T_EX – PDF. *Zpravodaj Československého sdružení uživatelů T_EXu*, (2):69–85, 1996.
- [19] Hàn Thế Thành. Přenositelný formát dokumentu a sázecí systém T_EX. Diplomová práce, Masarykova univerzita v Brně, Fakulta informatiky, 1996.
- [20] Hàn Thế Thành. *Micro-typographic extensions to the T_EX typesetting system*. Ph.D. thesis, Masarykova univerzita v Brně, Fakulta informatiky, 2000.
- [21] Hàn Thế Thành, Sebastian Rahtz, Hans Hagen. *The pdfT_EX user manual*, 2001. <http://www.tug.org/applications/pdftex/pdftex-s.pdf>.
- [22] Mark A. Wicks. Program dvipdf. Home page: <http://gaspra.kettering.edu/dvipdfm/>.
- [23] Timothy van Zandt. *PSTricks: PostScript macros for Generic T_EX*, 1993. User's guide.
- [24] Vít Zýka. Používáme pdfT_EX: vkládání obrázků. *Zpravodaj Československého sdružení uživatelů T_EXu*, 11(4):181–186, prosinec 2001.
- [25] Vít Zýka. Používáme pdfT_EX II: prezentace fotografií aneb jak na hyper-text. *Zpravodaj Československého sdružení uživatelů T_EXu*, 12(1):13–21, březen 2002.
- [26] Vít Zýka. Používáme pdfT_EX III: video a zvuk v prezentaci. *Zpravodaj Československého sdružení uživatelů T_EXu*, 12(2):47–55, březen 2002.

Summary: T_EX and PDF

T_EX is a system for quality typesetting with portability of source code within platforms and time. Portable Document Format (PDF) is format for electronic publishing. This paper shows possibilities of electronic publishing in PDF for pdfT_EX users.

Vít Zýka
České vysoké učení technické, Fakulta elektrotechnická
Centrum aplikované kybernetiky
Email: zyka@cmp.felk.cvut.cz

Je potřeba i v dnešní době dělat korektury? Při zpracování textu – jak ukazují různé publikace – ani moderní technologie nedokáží zabránit různým chybám. Článek obsahuje základní přehled problematiky korektur: rozdělení korektur, organizační a metodické postupy při provádění korektur, korekturní znaménka. Jsou srovnány klasické postupy se současností, ovlivněnou výpočetní technikou. Na závěr jsou připojeny vybrané prohršky se zaměřením na počítačovou terminologii.

Klíčová slova: korektury, dělení korektur, technické a organizační postupy, anglická terminologie

1. Druhy korektur

Korektury můžeme dělit jednak podle hlediska časové posloupnosti při zpracování, jednak podle hlediska obsahového.

1.1. Časová posloupnost korektur

„Klasické“ rozdělení korektur je ovlivněno postupy používanými v klasické sazbě, kdy autor zhotovoval rukopis v podobě strojopisu, ze kterého sazeč zhotovil opisem sazbu. Toto rozdělení [6] obsahuje korekturu

- domácí sloupcovou,
- autorskou sloupcovou,
- domácí stránkovou,
- autorskou stránkovou.

Domácí korektura se provádí v sazárně, autorskou korekturu provádí autor nebo redaktor.

Současnost: V době „převálcování“ lidstva výpočetní technikou se stalo nepsanou zvyklostí, že autor dodává rukopis v elektronické podobě. Tato elektronická podoba však může vypadat různě, následující čtyři skupiny lze považovat za nejdůležitější:

- a) rukopis v podobě souboru s prostým textem,
- b) rukopis zapsaný v určitém programovém produktu určeném pro zpracování dokumentů (jiným, než ve kterém se provádí sazba),

¹Článek byl poprvé publikován ve sborníku Jan Kasprzak, Petr Sojka: SLT 2002, KONVOJ 2002, ISBN 80-7302-043-2, na stranách 49–55. Publikováno se souhlasem vydavatele a autorů.

- c) rukopis doplněný o strukturní informace (označkováný dokument), strukturní značky bývají zpravidla dohodnuty předem,
- d) rukopis připravený pro sazbu v konkrétním, předem dohodnutém sázecím systému.

Volba způsobu odevzdání rukopisu je podmíněna autorovými znalostmi a také zvládnutím určitých počítačových dovedností.

Široké spektrum možností pořízení rukopisu je současně příčinou existence rozličných způsobů zpracování dokumentů, a tudíž i mnoha různých postupů při korekturách. Tabulka 1 ukazuje, které korektury přetrvávají i v současnosti při zodpovědném zpracování dokumentu.

Tabulka 1: Korektury při rukopisu v elektronické podobě

	a)	b)	c)	d)
domácí sloupcová	ano	ano ¹	ano/ne ²	ne
autorská sloupcová	ano ³	ano ¹	ne	ne
domácí stránková	ano	ano	ano	ano
autorská stránková	ano	ano	ano	ano

¹ – sloupcové korektury u varianty b) jsou nezbytné, protože před vlastní sazbou je třeba konvertovat z dodaného formátu

² – strukturně značkování dokumenty je nutné kontrolovat především z hlediska správného značkování autorem, chyby vznikající při převodu ze strukturního značení by měly být zpravidla zachyceny již při testování převodního programu

³ – lze vynechat u hladkých textů

Z uvedeného je zřejmé, že použití výpočetní techniky odstraňuje z výrobního procesu především sloupcové korektury, zejména autorskou sloupcovou.

Vynechání těchto korektur je však možné jen při správně připraveném dokumentu, neboť jen v těchto případech lze předložení rukopisu v elektronické podobě chápat jako předložení již korigovaného rukopisu.

I přesto, že autorskou sloupcovou korekturu si může autor v některých případech udělat sám, je předložení autorské stránkové korektury nezbytností, neboť autor má na svoji korekturu nárok podle autorského zákona.

1.2. Dělení podle obsahového hlediska

Z obsahové hlediska dělíme korektury na tyto kategorie:

gramatická korektura slouží k odstranění pravopisných chyb, překlepů, vadné interpunkce apod.;

stylistická korektura je do jisté míry totožná s jazykovou úpravou díla;

typografická korektura má za cíl například kontrolu správnosti použitých řezů písma, kontrolu správného vysazení symbolů (matematických, cizojazyčných znaků, mezerování), kontrolu horizontálního i vertikálního odsazení jednotlivých objektů, vysazení tabulek a zalomení obrázků;

formální korektura obsahuje kontrolu správnosti zápisu bibliografických citací, kontrolu počtu obrázků či tabulek a jejich číslování vůči odkazům v textu, kontrolu správného způsobu vyznačování aj., jejím cílem je také sjednocení úpravy těch publikací, které jsou sestavovány různými autory;

grafická korektura se provádí zejména tehdy, jsou-li součástí publikace barevné obrazové přílohy, obsahuje kontrolu barevnosti, kontrastu mezi písmem a pozadím u plnobarevných stran,

technická korektura patří do úkonů prováděných technickým redaktorem, který kontroluje číslování stran, vyřazení na archy, kvalitu podkladů pro tisk (například u předloh pro tisk z laserové tiskárny je třeba kontrolovat nejen kvalitu pokrytí tonerem, ale také případné zašpinění předlohy například špatným válcem) apod.

Ve své podstatě uvedené dělení podle obsahu nepřináší žádný nový prvek do současné korekturní praxe, neboť se používá již od dávných dob. Zcela určitě se však změnil důvod, proč toto dělení využíváme.

Současnost: Rozdělení korektur podle obsahu nabývá na významu právě s masovým rozšířením „domácího“ publikování. Ne každý, kdo v současnosti připravuje publikace do tisku, má takové vzdělání a takovou kvalifikaci, aby mohl sám zaručit, že dílo je po všech stránkách dokonalé.

Jen velká nakladatelství či vydavatelství si mohou dovolit zaměstnávat korektory na plný úvazek. Menší firmy, kterých je v nakladatelském oboru většina, si musejí poradit samy nebo si najímat specializované firmy či jednotlivce. Od externího korektora (například jednotlivce-jazykáře) nelze očekávat provedení korektur technických, mnohdy ani typografických či formálních. Stejně tak specialista na grafické korektury je schopen přehlédnout i pravopisné chyby v textech souvisejících s grafickým zpracováním.

Dokonce ani instituce, jakými jsou například vysoké školy s ediční činností, nemají vždy mezi zaměstnanci osobu odpovědnou a současně i kvalifikovanou k provádění korektur. Do jisté míry lze dokonce hovořit o vymizení profese „klasických“ korektorů.

Proto rozdělení korektur do určitých oblastí je nezbytné k zajištění správného zpracování díla. Jeho nevýhodou je skutečnost, že vede ke specializaci jednotlivých pracovníků a neumožňuje komplexní pohled na celý korekturní postup.

Kromě externistů je možné si v dnešní době zadat provedení korektur specializované firmě. Podle živnostenského zákona jsou korektorské práce součástí nakladatelské činnosti, která patří mezi živnosti volné, k nimž není třeba žádné předchozí kvalifikace či praxe. I proto je úroveň firem poskytujících korektorské

služby velmi rozdílná a – stejně jako u samostatných korektorů – ne všechny firmy jsou schopny provést kvalitně všechny druhy korektur.

2. Postupy při provádění korektur

2.1. Čtení korektur

Čtením korektur nazýváme činnost, při níž srovnává korektor otisk sazby s rukopisem. S ohledem na několikastupňové korektury lze jejich čtení definovat jako srovnání otisku předchozích korektur s otiskem aktuálních korektur.

Pop, Flégr a Pop [6] uvádějí, že srovnání textů je spolehlivější, čte-li korektor nejprve otisk sazby a potom rukopis (obecně: nejprve otisk nových korektur, potom starých).

Současnost: Často kladenou otázkou je, zda lze provádět čtení korektur z monitoru. Odpověď je prostá – možné to je, ale buď za cenu snížení kvality prováděných korektur, nebo s neúměrně vysokou zátěží pro oči. Pro toto tvrzení nejsou k dispozici vědecky ověřené výsledky, je to však zkušenost všech korektorů, se kterými jsem kdy přišel do styku.

Problémem v této souvislosti může být korektura webových stránek, které zpravidla nemívají tištěnou podobu.

Zde se jako schůdné řešení ukázalo rozdělení korektur na dvě části, analogické k sloupcovému a stránkovému korekturám. Při „sloupcové“ korektuře se kontroluje správnost textu (gramatická, stylistická, formální, případně i typografická – pokud se dá vůbec hovořit na webu o typografii), přičemž pro tyto korektury lze text vytisknout buď z prohlížeče, v případě gramatických a stylistických korektur lze dokonce využít i původního zdroje, pokud jsou stránky generovány (ze SGML, z databází aj.).

Druhá fáze – „stránkové“ korektury – slouží ke kontrole celkové grafické úpravy dokumentu, což je možné provádět na monitoru. Součástí této fáze je také kontrola platnosti hypertextových odkazů, přítomnosti všech obrázků, stylů apod., což bez počítače již nejde.

2.2. Použití korekturních znamének

Korekturní znaménka jsou popsána starší normou (ČSN 88 0410).

Dělí se do šesti základních skupin (výměna, vypuštění a vsunutí; změny v sazbě; změny mezer; odstavec a změny písma; oprava technický nedostatků sazby; zrušení korektury). Kromě zmíněné normy jsou vybraná, důležitější znaménka uváděna v celé řadě publikací [6, 1, 4, 5].

Současnost: Je třeba si přiznat, že ani změna technologie sazby nezpůsobila výraznější zásah do množiny chyb či technických nedostatků sazby.

Existuje sice několik značek, které v dnešní době buď nepoužijeme vůbec, nebo jen velmi zřídka (otočení řádku o 180°, odstranění hrotků apod.), většina značek je však i dnes aktuální.

2.3. Vyznačování korektur

Vyznačování provádí korektor dvojmo: jednou přímo v místě, kde se vyskytuje chyba, podruhé na okraji otisku korektur, vždy ve stejné výšce s řádkem s korigovanou chybou.

Znaménka se nesmějí opakovat na jednom řádku, případně v řádcích po sobě jdoucích z důvodu možné nejednoznačnosti výkladu korektury.

Vyznačování se provádí modrou nebo černou barvou.

Současnost: Stávající pravidla pro vyznačování korektur jsou platná i dnes, jen je třeba je rozšířit o jevy, které souvisejí s počítačovým zpracováním textů.

S ohledem na možnost tvorby stylů snad již ve všech sázecích systémech se velmi často vyskytují chyby, které jsou způsobeny nedokonalostí stylu. Tyto chyby se objevují v celé řadě míst připravovaného díla. Není proto nutné tyto chyby vyznačovat v místě každého výskytu, pro sazeče stačí poznámka pouze u výskytu prvního. Je otázkou, do jaké míry dokáže korektor rozpoznat, zda se jedná o chybu stylovou či nikoliv. Pro kvalifikované rozhodnutí (ušetří tím práci sobě i sazečovi) je nezbytné jeho širší vzdělání v oblasti zpracování textu.

Stejným způsobem, tzn. v místě prvního výskytu, je vhodné sazeče upozornit na systematické chyby v textu, které může sazeč opravit pomocí služby hledání a nahrazení.

Vyznačování modrou či černou barvou lze jen doporučit. Červená barva bývá (záleží na vnitřní pokynech) určena pro korektora-revizora, který dohlíží na provádění korektur. Tyto „revizorské“ opravy provedené jsou barevně odlišeny a mohou sloužit jako podklad pro další vzdělávání korektorů, případně také pro upřesňování vnitropodnikových norem.

Červenou barvu lze také doporučit pro označení oprav ve stylu či oprav systematických.

2.4. Korektor

2.4.1. Kompetence korektora

Od korektora se očekává dokonalá znalost českého pravopisu, znalost norem pro zpracování dokumentů nebo jeho částí a taktéž znalost typografie a estetických pravidel. Dříve se vyžadovala také znalost slovenského pravopisu [6].

Současnost: Po rozpadu federace není nezbytná již znalost slovenského jazyka, neboť v České republice se se slovenštinou setkáme nejčastěji jen v odborných či vědeckých publikacích (sbornících, časopisech).

Častější je potřeba angličtiny, kromě odborných publikací obsahujících více-jazyčné abstrakty či klíčová slova se často používají v textech anglické výrazy, názvy, jména, případně bibliografické citace anglicky psaných děl.

2.4.2. „Časová a prostorová složitost“ korektur

Čtení korektur vyžaduje značnou pečlivost a soustředěnost. Je všeobecně známo, že soustředěnost na práci tohoto typu kolísá v čase. Každý korektor by si měl proto stanovit podle svých individuálních schopností množství textu, které je schopen jednorázově zpracovat, odhadnout nutnou dobu relaxace a celkovou denní dobu, po kterou je schopen korektury provádět. Při stanovení uvedených časových údajů je nutné vzít v úvahu, že jednotlivé typy korektur jsou z hlediska soustředění různě časově náročné.

Z těchto údajů lze celkem spolehlivě odhadnout časovou náročnost celého korekturního zpracování díla, což je nezbytné pro plánování celého výrobního procesu i pro stanovení dodacích lhůt v případě zakázkové činnosti.

Vyšší výkonnosti lze dosáhnout optimálním uspořádáním pracovního prostředí. Pokyny pro ergonomické uspořádání pracovního prostředí, hygienické normy a další aspekty jsou zachyceny ve specializované literatuře.

3. Vybrané omyly autorů (i korektorů) v počítačové terminologii

koncovka -ovský × -ový, např. webový × webovský – správný tvar je webový, stejně tak internetový/-á (prohlížeč, adresa), nikoliv internetovský/-á

názvy firem či produktů jsou zpravidla registrovanými ochrannými nebo obchodními známkami, a proto je píšeme zásadně podle registrovaného názvu (Linux, UNIX, PostScript, CorelDRAW, MS-DOS, MS Windows, Windows NT; Hewlett-Packard, KONVOJ). V některých případech se může změnou písmen dokonce jednat i o rozdílný význam (SCO UNIX jako ochranná známka × Unix jako popis vlastností operačního systému).

skloňování názvů firem či produktů způsobuje změnu psaní malých a velkých písmen, název firmy či produktu se mění v tzv. zkratková slova, která se píší s velkým písmenem na začátku (v Dosu, do Linuxu),

přídavná jména odvozená z názvů firem či produktů píšeme výhradně malými písmeny (linuxový, unixový, postscriptový, dosový)

zkratka WWW není přídavným jménem, ale podstatným, a proto ji ve spojení s dalším podstatným jménem používáme v pozici přívlastku neshodného (stránky WWW, prohlížeč WWW aj., nikoliv WWW stránky, což známe například z němčiny – WWW-Seiten)

4. Terminologický slovníček

číst korekturu proofread, correct the proofs
čtení korektur proof reading
imprimatur imprimatur (pass for press)
imprimované stránky press proofs
imprimovaný rukopis hard copy
imprimovat pass for press, pass the proofs for press
korektor reader, proofreader, copy editor
korektorna proofreading room
korektorova značka proofreader's mark
korektorství correcting, proofreader's job
korektura book proof, proof correction, proof reading
korektura (obtah) proof, proof-sheet
korektura domácí (vnitřní) first proof, house correction
korektura autorská author's proofs (author's corrections)
korektura nakladatelská editor's proofs
korektura stránková page proof
korektura tiskárenská first proof, rough proof
korekturní značka, znaménko proof correction mark, correction mark
rukopis manuscript, copy, handwriting

5. Závěr

Problematika provádění korektur je velmi rozsáhlá a její podrobný rozbor přesahuje rámec tohoto příspěvku. Proto jsou zde uvedeny jen podle názoru autora nejdůležitější body, kterými je třeba se zabývat při korektorské činnosti.

Autor rád přivítá jakékoliv další podněty, připomínky či odkazy k tématu na své emailové adrese.

Reference

- [1] Beran, V. *Typografický manuál*. 1. vyd. Manuál: Náchod, 1994. ISBN 80-901824-0-2.
- [2] *Česko-anglický a anglicko-český slovníček nakladatelské a tiskařské terminologie*. 1. vyd. 99 s. International Book Development: Londýn a SČNK: Praha, 1998. ISBN 80-902495-1-5.
- [3] ČSN 88 0410: Korekturní znaménka.
- [4] Hanáček, P. a kol. *Jak publikovat na počítači*. 1. vyd. 216 s. Science: Veletiny, 1996. ISBN 80-901475-7-7.

- [5] Kočíčka, P., Blažek, F. *Praktická typografie*. 1. vyd. 310 s. Computer Press: Praha, 2000.
- [6] Pop, P., Flégr, J., Pop, V. *Sazba I Ruční sazba*. 2. vyd. 188 s. SPN: Praha, 1989.
- [7] Zákon č. 121/2000 Sb., o právu autorském (autorský zákon).
- [8] Zákon č. 356/1999 Sb., o živnostenském podnikání (živnostenský zákon).

Summary: Proofreading and Proofreaders in the New Millenium

Do we need proofreading today? Even leading edge technologies are not error prone. Paper shows basic problems of proofreading: different kinds of proof-reader's work, methodology, proofreader's marks. Classical approaches are compared with the information technology based ones. We sum up by enumeration of most usual errors in Czech computer terminology.

Tomáš Hála
Mendelova zemědělská a lesnická univerzita v Brně
Zemědělská 1, 613 00 Brno
Ústav informatiky, Provozně ekonomická fakulta
Email: thala@mendelu.cz
a
KONVOJ, spol. s r. o., Berkova 22, 612 00 Brno
Email: konvoj@konvoj.cz

Typografické systémy založené na systému $\text{T}_\text{E}\text{X}$ jsou často používány pro realizaci odborných, technických a vědeckých publikací, kde se ve značné míře používají grafy pro znázornění souboru hodnot. Tvorba grafů je dána řadou pravidel, jejichž cílem je vytvoření snadno čitelného, přehledného a srozumitelného výsledku. Jedním ze základních typografických pravidel je použití identických vizuálních prostředků pro objekty identického významu. Článek se zabývá některými základními zásadami tvorby grafů a také automatizovanou podporou pro jejich dávkovou tvorbu.

1. Úvod

V odborných pracích se často data prezentují grafickou formou. Forma grafu bývá vhodným doplňkem dat vyjádřených tabelárně. Hlavním účelem grafu je zvýšení přehlednosti dat a vyjádření charakteru, který není například z tabulky na první pohled patrný. Na druhé straně však graf neposkytuje možnosti tak přesného odečítání hodnot jako tabulka, proto se obě formy vyjádření vzájemně doplňují.

2. Grafické znázornění

Grafické znázornění je vyjadřovací forma používaná k zobrazení zkoumaných jevů. Jejím výsledkem je náčrt, který je sestaven pomocí souboru grafických prostředků. Náčrt může svým charakterem být [2]:

- *schéma* – v hlavních, základních rysech naznačené složení, náčrt procesu, jevu, zákonitosti;
- *diagram* – grafické (geometrické) znázornění průběhu nějakého procesu, vztahu či závislosti, nejčastěji soustavou kartézských os.

Používané grafické prostředky mohou mít význam *ideografický* nebo *geometrický*.

¹Článek byl poprvé publikován ve sborníku Jan Kasprzak, Petr Sojka: SLT 2002, KONVOJ 2002, ISBN 80-7302-043-2, na stranách 57–67. Publikováno se souhlasem vydavatele a autorů.

2.1. Ideografické prostředky

Ideografické prostředky mají kvalitativní význam (číslice, písmena, schematické tvary, čáry, šrafování, barvy apod.). Jsou základním grafickým prostředkem pro kreslení schémat, ale používají se i u diagramů.

Čáry jsou velmi frekventovanou rekvizitou ve všech druzích schémat a diagramů. Musí být od sebe snadno rozeznatelné, aby byl graf čitelný, bez optických klamů. Základním typem čáry je čára plná. Používá se vždy k vyznačení nejvýznamnější skutečnosti v daném grafu. Dalšími typy čar jsou čárkovaná, tečkovaná, čerchovaná a kombinovaná. Používají se ke znázornění dalších skutečností. Některé čáry mají svůj ustálený význam, například čárkovaná čára se používá ke znázornění průměrné hodnoty. Čáry lze dále odlišit tloušťkami, případně i barvami. Tloušťky čar odpovídají typografickým zásadám. Jemné linky mají tloušťky kolem šestnáctiny petitu (0,5 bodu), tlustší linky jsou osminky (1 b). Pro šrafy lze využít i tloušťky čtvrtbodové. Barevné podání lze řešit kombinací černé a pestré barvy. Je-li třeba použít více barev, přidávají se kontrastní barvy z opačné strany spektra.

Různé druhy čar používané v grafech

Různé druhy šrafování používané v grafech

Obrázek 1: Ideografické prostředky grafů

Číslice a písmena tvoří důležitou identifikační složku. Pro kvalitní čtení je nezbytné, aby se stupeň použitého písma pohyboval u nejmenších textů nad hranicí 6 bodů. Popisky os nebo zobrazených hodnot je vhodné použít písmo 8–9 bodů. Není-li na popisky dost místa, je vhodné použít zkratky vysvětlené v textu, nikoliv zmenšovat stupeň použitého písma. S výhodou lze použít některého bezserifového typu písma, které je i v menším stupni zřetelnější.

U všech textů a čísel je nezbytné respektovat všechna typografická a pravopisná pravidla.

2.2. Geometrické prostředky

Jedná se o prvky, které svou velikostí nebo vzdáleností naznačují úroveň sledované hodnoty. Mohou to být geometrické útvary – bod, úsečka, plocha, těleso – a symbolické tvary – například různě velké figurky. V některých případech se ideografický prostředek v různém provedení může stát prostředkem geometrickým, například šrafování o různé hustotě vyjadřující smluvené hodnoty.

2.3. Grafický výklad

K rozlišení kvalitativního a kvantitativního významu grafických prvků slouží *grafický výklad*. Určuje také zásady, podle nichž se graf má číst. Mezi základní prostředky výkladu grafu patří soustava souřadnic, stupnice a moduly, legenda, obsahující přehled všech grafických prvků, dále pak název grafu, podtitul, všechny vysvětlivky, poznámky a podobně.

Soustava souřadnic je soustava čar, jejichž významem je kvalitativní orientace libovolného bodu pomocí nejkratší vzdálenosti od os, určené jejich stupnicemi. Nejčastěji se využívá kartézská soustava pravoúhlých souřadnic, kterou tvoří dvě na sebe kolmé přímky rozdělující rovinu na čtyři kvadranty. Vodorovná přímka je *osa úseček*, značená symbolicky x , svislá přímka se nazývá *osa pořadnic* (označení y). Průsečík se nazývá *počátek* a dělí každou osu na dvě poloosy. Na osu x se vždy vynáší nezávisle proměnná, na osu y pak hodnoty závisle proměnné.

Při okótování bodů na osách souřadné soustavy získáme *stupnici*. Proto se osa také nazývá *nositelka stupnice*. Při tvorbě stupnice je nejdůležitější správně stanovit základní délkovou jednotku úsečky, která odpovídá zobrazovanému číselnému intervalu. Tato délka je jedním z rozhodujících faktorů čitelnosti grafu. Vypočtená délka úsečky také určuje přesnost čtení údajů. Stupnice mohou být rovnoměrné (lineární) a nerovnoměrné (například logaritmické).

Modul stupnice M se určuje u rovnoměrných stupnic v závislosti na ploše, která je grafu určena. Je dán výrazem

$$M = \frac{d}{H - D},$$

kde d je požadovaná délka stupnice, D – dolní mez hodnot zobrazovaného souboru dat, M – horní mez hodnot zobrazovaného souboru dat. Jednotka modulu je dána délkovou jednotkou.

Nejmenší dílek stupnice η by neměl klesnout pod 1 mm. Nejmenší dílek stupnice určuje přesnost čtení hodnot ε , která je dána vztahem

$$\varepsilon = \frac{\eta}{M}.$$

Sestavování stupnice lze také provádět v závislosti na požadované přesnosti čtení. Spočívá ve stanovení přesnosti, z níž se vypočte modul, a ten se dále vynásobí počtem hodnot zkoumaného souboru dat.

Modul na ose y se obvykle stanovuje tak, aby výsledné grafické pole mělo čtvercový tvar, případně tvar vhodného obdélníka.

Mezi nerovnoměrnými stupnicemi zaujímá zvláštní místo stupnice logaritmická. Používá se podle povahy vynášených dat. Její konstrukce je poněkud složitější než u stupnice rovnoměrné. Vynášejí se hodnoty odpovídající logaritmům čísel 1–10. Může mít více cyklů (sad hodnot 1–10), které jsou vůči sobě vynásobeny deseti.

Sdružené stupnice (dvojstupnice) se používá v případě, že je potřebné vynášet absolutní i relativní hodnoty současně. Absolutní hodnoty jsou vynášeny vlevo, relativní vpravo. Pro sdruženou stupnici lze použít jednu čáru, nebo dvě čáry vedle sebe.

Grafická síť se používá pro rychlejší určení přibližné hodnoty z grafu. Je to soustava rovnoběžek s osami souřadné soustavy. Rovnoběžky procházejí jednotlivými body stupnice, nejčastěji právě těmi, které jsou opatřeny kótami. Grafická síť může být jednoduchá, jsou-li rovnoběžky vedeny pouze s jednou osou (jednoduchá grafická síť vodorovná, nebo svislá). Dvojitou grafickou síť tvoří rovnoběžky s oběma osami. Hustota sítě má vliv na přesnost odečtu hodnot. Pro rychlý přehled se používají řídké sítě s velkým grafickým intervalem.

Formát grafu, tj. velikost, na jakou bude kreslen, musí odpovídat složitosti grafu. Je nezbytné, aby byly složitější grafy kresleny na větší formát, a tím byly všechny podstatné prvky dobře viditelné. Dalším důležitým vodítkem pro stanovení formátu grafu je text, do něhož je graf vkládán – stupeň písma, šířka řádku, případně další typografické veličiny textu musí být sladěny s obdobnými prvky grafu. Pro snadné čtení by měl být graf umístěn vodorovně, nevejde-li na šířku sazby, pak otočený o 90° proti směru hodinových ručiček (nadpis grafu je pak vlevo). Stupnice se u širších grafů může opakovat vpravo, u obou stupnic nesmí chybět jednotky. Pro zpřesnění údajů lze vypisovat číselné hodnoty přímo u jednotlivých grafických bodů.

Orientační přehled grafických prvků grafu je uveden na obrázku 3.

3. Rozdělení grafů

Základní rozdělení na schémata a diagramy již bylo uvedeno. Diagramy lze dále rozdělit na prosté (porovnání malého počtu prvků nebo jejich skupin), na

Obrázek 2: Stupnice, modul

diagramy statistických řad, diagramy časových řad (chronodiagramy) a na prostorové diagramy (topodiagramy).

Podle formy grafického obrazu lze klasifikovat rozměrové grafy (kvantita jevy je znázorněna poměrovou velikostí grafického obrazu), souřadnicové grafy (kvantita je vyjádřena pomocí tvarů v souřadné soustavě se stupnicemi, trojrozměrné grafy (zvláštní případ předchozích dvou), statistické mapy (kromě prvků rozměrových a souřadnicových grafů mají ještě kvalitativní znázornění na mapě) a popularizační grafy (jednoduché a názorné typy, piktogramy apod.).

Volba grafu závisí na charakteru dat a účelu zobrazení. Hlavní typy grafů jsou:

Název grafu

Podtitul

Všeobecná poznámka

*) zvláštní poznámka

Obrázek 3: Uspořádání grafických prostředků v grafu (podle [1])

- *Bodový graf* – používá se pro zjištění polohy bodu v souřadnicové soustavě. Body samotné nemají rozměr, hodnota je určena pozicí bodů. Nepravé bodové diagramy naznačují velikostmi bodů četnosti v určité ploše.
- *Spojnicový graf* – vzniká spojením bodů vynesných v ploše. Umožňují jednoduše sledovat více řad v jednom grafu, kde jsou řady rozlišeny ideografickými prvky (barva, tloušťka a provedení čáry). Spojnicový graf se používá velmi často, a to zejména pro vyjádření četností.
- *Plošný graf* – používá jako základní grafický vyjadřovací prostředek plochu čtverce, obdélníka, kruhu apod. Plocha indikuje velikost sledovaného jevu. Vyjádření plochou neumožňuje jednoduše srovnávat zkoumané jevy, ale s výhodou je plošný graf možné využít tam, kde je výsledná hodnota dána součinem dvou ukazatelů – ukazatele tvoří strany obdélníka, výsledkem je plocha obdélníka.
- *Sloupkový graf* – je zvláštním případem plošného grafu, kdy jeden rozměr plochy je konstantní (často šířka sloupku) a druhý rozměr se mění

(vyjadřuje kvantitu sledovaného jevu). Sloupkový graf je nejčastěji užívaným prostředkem pro srovnávání.

- *Kruhový (koláčový) graf* – je druhým speciálním případem plošného grafu. Nejčastěji se využívá pro vyjádření struktury pomocí kruhových výsečí s úhlem reprezentujícím relativní četnosti.
- *Kartogram* – vyjadřuje územní rozložení zkoumaných jevů. Základním grafickým prvkem je mapa, rozdělená na regiony, v nichž jsou kvantitativní veličiny naznačeny různým šrafováním, barevnou výplní nebo odstíny.
- *Kartodiagram* – podobně jako kartogram znázorňuje územní rozložení zkoumaných jevů, avšak v jednotlivých regionech mapy jsou znázorněny jiné grafy vyjadřující příslušné hodnoty.
- *Piktogram* – slouží k popularizačním účelům. Je založen na opakování zvoleného ideografického prvku (například obrázku auta) tolikrát, kolik odpovídá četnosti základních jednotek. Pro zpřehlednění se symboly mohou sdružovat do skupin například po deseti. Graf je málo přesný, ale výrazný.
- *Stereogram (prostorový graf)* – umožňuje sledování tří sledovaných hodnot znaku v jednom grafu. Realizuje se axonometrickým nebo kosoúhlým zobrazením.
- *Pseudoprostorový graf* – znázorňují pouze dvě sledované hodnoty znaku, ale jsou konstruovány v kosoúhlém zobrazení tak, že jeden rozměr je pevně určen. Nejčastějším příkladem je použití sloupkového grafu, kde místo sloupku je kvádr, jehož jeden rozměr (hloubka) je konstantní.

4. Prostředky pro tvorbu grafů

Programové prostředky používané pro tvorbu grafů z dat v tabulkách jsou většinou realizovány jako interaktivní nástroje, kde si uživatel volí z nabídky požadované parametry a systém zobrazuje aktuální grafický tvar. Patří mezi nejrozšířenější aplikace, protože jsou obvyklou součástí kancelářských balíčků (například Open Office, StarOffice, MS Office). Kromě toho existují nástroje integrované do specializovaných statistických balíčků, umožňující vytvářet grafy různých statistických charakteristik (například Statgraphics).

Hlavní výhodou interaktivního nástroje je okamžitá vizuální kontrola dosaženého výsledku. Většinou je k dispozici široká paleta různých typů grafů, možností ovlivňování tvaru všech komponent a okamžité překreslování výsledku při změně dat.

Zásadní nevýhoda interaktivního přístupu se ovšem projeví v okamžiku, kdy je nutné vytvořit sadu grafů s podobnými vlastnostmi. Nemá-li program možnost uložit nastavení grafu a opakovaně je použít, znamená to vždy opakovanou ruční práci u každého grafu zvlášť, přičemž není zaručeno, že výsledek bude skutečně

jednotný. Z typografického hlediska je však nutné, aby významově stejné prvky měly v dokumentu vždy identickou podobu.

Uvedené produkty však ani nemají příliš dobré vlastnosti týkající se dalšího zpracování vykreslených grafů. Například MS Excel nemá vůbec žádnou možnost exportovat graf do některého použitelného grafického formátu, výsledek lze nejvýše vložit jako objekt do jiného programu, ovšem s reálným rizikem, že podoba na otisku nebude zdaleka odpovídat tvaru dosaženému interaktivní volbou příslušných nastavení. Je pochopitelné, že od bezkoncepčního produktu s množstvím principiálních nedostatků a hrubých programátorských chyb nelze čekat nic lepšího, nicméně ostatní produkty nejsou diametrálně odlišné.

5. Automatizovaná tvorba grafů

Problém vytvoření grafu respektujícího základní zásady typografie a správného zobrazování dat je někdy nutné z uvedených důvodů řešit jinými prostředky. Jednou z možností je generování výsledného tvaru grafu z textově vyjádřených dat podle parametrů v konfiguračním souboru. Tuto možnost realizoval ve své diplomové práci [1] Fojtlík. Vytvořil program ChartMaker, jehož výstupem je zdrojový text s příkazy van Zandtova balíku PSTricks.

Tímto postupem se zároveň řeší problém integrace grafů do zdrojového textu $\text{\TeX}$ u ($\text{\LaTeX}$ u).

5.1. Princip činnosti

Program zpracovává dva soubory. Hlavním je datový soubor, který zároveň obsahuje individuální nastavení pro daný graf, druhým zpracovávaným souborem je konfigurace, která platí pro celou sadu grafů. Schematicky lze činnost programu a začlenění grafu znázornit obrázkem 4.

Obrázek 4: Schematické znázornění činnosti programu ChartMaker

5.2. Formáty zdrojových souborů

Vstupní data jsou zapsána ve formátu CSV, na jednotlivých řádcích jsou údaje oddělené středníky, textové položky jsou opatřeny uvozovkami. Tato data jsou předcházena ještě možnými nastaveními, jak ukazuje následující příklad:

```
Title=Zkušební graf
Subtitle=Určen pro článek na SLT
Note=(nejmenší hodnota je 0,5)
XAxisName=Osa $$
YAxisName=Osa $$
ShowXAxisLabels=true
ShowYAxisLabels=1
ShowXAxisPrimaryMarks=1
ShowYAxisPrimaryMarks=1
ShowXAxisSecondaryMarks=1
ShowYAxisSecondaryMarks=1
ShowXPrimaryGrid=1
ShowYPrimaryGrid=1
ShowXSecondaryGrid=1
ShowYSecondaryGrid=1
DataInRows=0
FirstRowLabels=1
FirstColumnLabels=1
Data
;"Sloupec 1";"Sloupec 2";"Sloupec 3";"Sloupec 4"
"Řádek 1";10;8;14;3
"Řádek 2";5.125;1;15;2
"Řádek 3";12;3;4;9
"Řádek 4";12;3;4;9
"Řádek 5";12;3;4;9
"Řádek 6";12;3;4;9
```

Inicializační soubor má formát INI souboru, tj. nastavení mnoha parametrů soustředěných do tématických sekcí. V každé sekci jsou nastavovány parametry různých typů, není-li určitý parametr uveden, uplatní se jeho implicitní hodnoty. Příklad inicializačního souboru:

```
[Main]
Chart.Type=column
Chart.SizeX=130
Chart.SizeY=100
TitleStyle.Style=\bfseries\LARGE
TitleStyle.Color=red
SubtitleStyle.Style=\bfseries\large
SubtitleStyle.Color=black
NoteStyle.Style=\small
NoteStyle.Color=black
ColumnOverlap=90
ColumnSpacing=100
Background.Style.Type=solid
Background.Style.Transparent=false
Background.Style.HatchSep=1.404
Background.Style.HatchWidth=0.1404
Background.Style.HatchAngle=45
Background.Style.HatchColor=gray
Background.Style.FillColor=white
Background.Border.Style.Type=solid
Background.Border.Style.

DashLineLength=1.755
Background.Border.Style.
DashSpaceLength=1.053
Background.Border.Style.DotSpace=1.053
Background.Border.Width=0.1404
Background.Border.Color=black

[Axis]
XAxis.Style.Type=solid
XAxis.Style.DashLineLength=1.755
XAxis.Style.DashSpaceLength=1.053
XAxis.Style.DotSpace=1.053
XAxis.Width=0.1404
XAxis.Color=black
XAxis.Arrows=none
XAxisNameStyle.Style=\bfseries
XAxisNameStyle.Color=black
XAxisNameStyle.Angle=0
XAxisLabelStyle.Style=\small
XAxisLabelStyle.Color=Black
XAxisLabelStyle.Angle=0
```

Zkušební graf

Určen pro článek na SLT

(nejmenší hodnota je 0,5)

Obrázek 5: Příklad vykresleného grafu

```
XAxisLabelSpace=3
YAxisLabelSpace=3
XAxisNameSpace=10
YAxisNameSpace=10
YAxis.Style.Type=solid
YAxis.Style.DashLineLength=1.755
YAxis.Style.DashSpaceLength=1.053
YAxis.Style.DotSpace=1.053
YAxis.Width=0.1404
YAxis.Color=black
YAxis.Arrows=lrarrow
YAxis.NameStyle.Style=\bfseries
YAxis.NameStyle.Color=black
YAxis.NameStyle.Angle=0
YAxis.LabelStyle.Style=\small
```

```
YAxis.LabelStyle.Color=black
YAxis.LabelStyle.Angle=0
PrimaryMarkLength=2
SecondaryMarkLength=1
```

```
[Grid]
PrimaryGrid.Style.Type=solid
PrimaryGrid.Style.DashLineLength=1.755
PrimaryGrid.Style.DashSpaceLength=1.053
PrimaryGrid.Style.DotSpace=1.053
PrimaryGrid.Width=0.1404
PrimaryGrid.Color=black
PrimaryGrid.Arrows=none
SecondaryGrid.Style.Type=solid
```

```

SecondaryGrid.Style.DashLineLength=1.755
SecondaryGrid.Style.DashSpaceLength=1.053
SecondaryGrid.Style.DotSpace=1.053
SecondaryGrid.Width=0.1404
SecondaryGrid.Color=black
SecondaryGrid.Arrows=none

```

[Lines]

```

Lines.Count=2
Line1.Style.Type=solid
Line1.Style.DashLineLength=1.755
Line1.Style.DashSpaceLength=1.053
Line1.Style.DotSpace=1.053
Line1.Width=0.5616
Line1.Color=black
Line1.Arrows=none
Line2.Style.Type=dashed
Line2.Style.DashLineLength=1.755
Line2.Style.DashSpaceLength=1.053
Line2.Style.DotSpace=1.053
Line2.Width=0.5616
Line2.Color=black
Line2.Arrows=none

```

[Areas]

```

Areas.Count=4
Area1.Style.Type=vlines
Area1.Style.Transparent=false
Area1.Style.HatchSep=1.404
Area1.Style.HatchWidth=0.1404
Area1.Style.HatchAngle=45
Area1.Style.HatchColor=black
Area1.Style.FillColor=white
Area1.Border.Style.Type=solid
Area1.Border.Style.DashLineLength=1.755
Area1.Border.Style.DashSpaceLength=1.053
Area1.Border.Style.DotSpace=1.053
Area1.Border.Width=0.5616
Area1.Border.Color=black
Area2.Style.Type=hlines
Area2.Style.Transparent=false
Area2.Style.HatchSep=1.404
Area2.Style.HatchWidth=0.1404
Area2.Style.HatchAngle=45
Area2.Style.HatchColor=black
Area2.Style.FillColor=white

```

```

Area2.Border.Style.Type=solid
Area2.Border.Style.DashLineLength=1.755
Area2.Border.Style.DashSpaceLength=1.053
Area2.Border.Style.DotSpace=1.053
Area2.Border.Width=0.5616
Area2.Border.Color=black
Area3.Style.Type=hlines
Area3.Style.Transparent=false
Area3.Style.HatchSep=1.404
Area3.Style.HatchWidth=0.0404
Area3.Style.HatchAngle=90
Area3.Style.HatchColor=black
Area3.Style.FillColor=white
Area3.Border.Style.Type=solid
Area3.Border.Style.DashLineLength=1.755
Area3.Border.Style.DashSpaceLength=1.053
Area3.Border.Style.DotSpace=1.053
Area3.Border.Width=0.5616
Area3.Border.Color=black
Area4.Style.Type=crosshatch
Area4.Style.Transparent=false
Area4.Style.HatchSep=1.404
Area4.Style.HatchWidth=0.1404
Area4.Style.HatchAngle=45
Area4.Style.HatchColor=black
Area4.Style.FillColor=white
Area4.Border.Style.Type=solid
Area4.Border.Style.DashLineLength=1.755
Area4.Border.Style.DashSpaceLength=1.053
Area4.Border.Style.DotSpace=1.053
Area4.Border.Width=0.5616
Area4.Border.Color=black

```

[Dots]

```

Dots.Count=2
Dot1.Style.Type=fullcircle
Dot1.Style.ScaleX=1
Dot1.Style.ScaleY=1
Dot1.Style.Angle=0
Dot1.Color=black
Dot2.Style.Type=fulltriangle
Dot2.Style.ScaleX=1
Dot2.Style.ScaleY=1
Dot2.Style.Angle=0
Dot2.Color=black

```

Detailní popis všech parametrů lze nalézt v diplomové práci ing. Fojtlíka. Z uvedených příkladů datového a inicializačního souboru byl vytvořen graf na obrázku 5. Pro vložení grafu do zdrojového souboru je nutné připojit styly `multido` a `pstricks`, příkladem minimálního zdrojového souboru může být tento text:

```

\documentclass{article}
\usepackage{czech,multido,pstricks}
\begin{document}
\input graf.pst
\end{document}

```


6. Závěr

Systém automatizované tvorby grafů prezentuje určitou myšlenku, která má za cíl usnadnit tvorbu správně vytvořených grafických reprezentací dat vyjádřených většinou tabelárně. Konkrétní aplikace nevyčerpává všechny potenciální možnosti, obsahuje jisté dětské nemoci, ale ukazuje cestu, kterou je možné dále rozšířit a zkvalitnit.

Reference

- [1] Fojtlík, V. *Automatizovaná grafická prezentace dat*. Diplomová práce. Brno: MZLU, 2002.
- [2] Klimeš, L. *Slovník cizích slov*. Praha: SPN, 1987.
- [3] Kubíček, J. *Statistika (vyjadřovací prostředky a formy ve statistice)*. Brno: VŠZ, 1988.
- [4] van Zandt, T. *PSTricks. User's Guide*. [Elektronický soubor]. 1993.

Summary: Batch Graph Generation in the T_EX-Based Systems

Typesetting systems based on T_EX are often used for the realization of technical and scientific publications, where graphs are used for visualization to the large extent. There is set of rules that should be used for the creation of readable, clearly organized and arranged results. One of basic typographical rules is using the same visual appearance for the objects of the same kind. The paper deals with the basic principles of graph creation and batch typesetting.

Jiří Rybicka
Ústav informatiky
Provozně ekonomická fakulta
Mendelova zemědělská a lesnická univerzita v Brně
Zemědělská 5, 613 00 Brno
Email: rybicka@mendelu.cz

OFS (Olšákův fontový systém) je balíček TeXových maker, který umožní získat větší přehled nad rozsáhlými kolekcemi fontů a umožní s nimi poměrně snadnou manipulaci. Balík byl vyvinut pro získání přehledu nad fonty z Typokatalogu Střešovické písmolijny [1]. Na základě poptávky L^AT_EXových uživatelů byl balík napsán ještě jednou pro L^AT_EX, kde využívá NFSS a pokouší se je trošičku vylepšit. Základní uživatelské příkazy OFS jsou pak v obou prostředích (plain i L^AT_EX) stejné.

Klíčová slova: OFS, fonty, NFSS, L^AT_EX, C_Splain, C_SL^AT_EX, C_Sfonty

1. Úvodem

Makro OFS jsem si napsal hlavně proto, abych se vyznal ve stovkách fontů, které pocházely ze Střešovické písmolijny a pro které jsem před rokem udělal T_EXovou podporu. Protože jsem plainista, šlo mi hlavně o to udělat makro co nejvíce srozumitelné uživateli plainu, který potřebuje přesně vědět, co to makro dělá. Tento požadavek například L^AT_EXové NFSS nesplňuje.

Později se začali o OFS zajímat též někteří uživatelé L^AT_EXu. Tož jsem se přemohl a pokusil OFS napsat ještě jednou, tentokrát pod L^AT_EXem s využitím NFSS. Cílem tohoto přepsání bylo hlavně to, aby se uživatelské příkazy pro vyhledávání a přepínání rodin fontů zcela shodovaly v L^AT_EXové verzi s verzí plainovou. L^AT_EXová verze přesto umí podstatně méně věcí než plainová, protože pokud by měla umět vše, musel bych NFSS zcela odmítnout a fontový modul L^AT_EXu si napsat po svém. Tím bych ale popřel L^AT_EX jako takový, takže jsem zůstal u respektování principů NFSS. Věci, které se v NFSS dají dělat velmi těžko, jsem raději nedělal.

Z těchto důvodů jsem se v L^AT_EXu nepouštěl ani do podpory matematických fontů. Koncepce matematiky v NFSS mi připadá jako plainistovi poněkud nerosrozumitelná. Navíc velké kolekce fontů, které byly hlavní motivací OFS, jsou většinou textové. Pokud chce někdo použít matematickou sadu fontů v L^AT_EXu, použije `\usepackage{styl}` a ten styl je k té matematické sadě většinou dodáván. Nepovažoval jsem tedy řešení matematiky v L^AT_EXu za prioritní. Na druhou stranu matematické rodiny v plainu se pomocí OFS doplňují a zavádějí ve všech

¹Článek byl poprvé publikován ve sborníku Jan Kasprzak, Petr Sojka: SLT 2002, KONVOJ 2002, ISBN 80-7302-043-2, na stranách 79-92. Publikováno se souhlasem vydavatele a autorů.

velikostech velmi elegantně. Věřím, že alespoň nějakému plainistovi (kromě mě) se toto řešení bude hodit.

Makro OFS jsem zveřejnil na [2] pod T_EX-like licencí, tj. je volně k mání, ale změny pod stejným názvem nesmí dále šířit nikdo jiný, než autor. Do T_EXových distribucí jsem makro zatím neprosadil, protože bohužel chybí anglická dokumentace. Domnívám se, že kdyby anglická dokumentace byla, nebyl by se zařazením do T_EXových distribucí problém a makro by mohlo využít daleko více lidí. Bohužel, nejsem v angličtině natolik zdatný, abych tento problém rychle překonal. Pro letošní letní prázdniny jsem sice plánoval, že se pokusím anglickou dokumentaci napsat, jenže přišla voda. . .

K OFS existuje samozřejmě česká dokumentace [3] podrobně popisující chování makra. Navíc jsem se o OFS zmínil už v článku o T_EXové podpoře Štormových fontů [4]. Dovolil jsem si zde přesto znovu k tomuto tématu vrátit. Nechci být nyní tak technicky exaktní jako v manuálu [3], ale zase mám zde více místa k rozepsání možností makra, než jsem měl ve článku [4], který byl především o Štormových fontech.

2. Základy uživatelského rozhraní

Uživatelské rozhraní je shodné v L^AT_EXové verzi OFS i v plainové snad s výjimkou zavedení makra OFS, které v plainu provedeme jednoduše pomocí `\input ofs [kolekce, fontů]`, zatímco v záhlaví L^AT_EXového dokumentu píšeme `\usepackage [kolekce, fontů] {ofs}`.

Abych v tomto textu nemusel každou chvíli větvit svůj výklad na situaci vhodnou pro plain a pro L^AT_EX, rozhodl jsem se zde předpokládat, že pracujeme pouze s plainem (například s C_Splainem). L^AT_EXový uživatel si bude muset dohledat specifika verze OFS pro svůj formát v dokumentaci [3].

Příkazem `\fontusage` dostaneme na terminál a do logu základní informace o uživatelských příkazech:

```
$ tex ofs \fontusage
This is TeX, Version 3.14159 (Web2C 7.3beta5)
(/usr/local/share/texmf/tex/csplain/ofs.tex
OFS (Olsak's Font System) based on plain initialized. <Oct. 2002>
(/usr/local/share/texmf/tex/csplain/ofsdef.tex)
\fontusage: ===== Olsak's Font System, usage: =====
\input ofs [sjannon, sdynamo, a35] ... for example
\showfonts ... shows all loaded font families (by previous \input)
\setfonts [Family/] ... local switch to the new family, after this, the
  \rm, \bf, \it, bi will switch to the variants. The current size is used.
\setfonts [/size] ... local switch to the new size of fonts, the family is
  not changed. The "size" has the following possible formats:
```

```

at<dimen>          ... the same as \font\something=file at<dimen>
<dimen>           ... the same as at<dimen>
<number>          ... the same as at<number>pt
scaled<number>    ... the same as \font\something=file scaled<number>
mag<decimal-number> fonts will be magnified by given coefficient
                    depend on current size of the fonts.
\setfonts [Family/size] ... switch to the new family at given size
\setfonts [Family-vr/] ... switch to the specified font, the current size
                    is used. The "vr" is acronym for variant (bf for example).
\setfonts [Family-vr/size] ... switch to the specified font.
\fontdef\name [Family/size] ... same as \gdef\name{\setfonts[Family/size]}
The "Family" or "size" parameter may be empty.
\fontdef\name [Family-vr/size] ... \name is fixed-font switch iff:
    "size" is no empty and no mag<dec-number>.
Fixed-font switch "\name" is implemented as \global\font\name=file.
\setmath [size/size/size] ... set math it/rm as current it/rm + use PS Symbol
\nofontmessages, \logfontmessages, \displayfontmessages, \detailfontmessages
... the levels of log.
*
```

Vidíme, že příkaz `\showfonts` nám ukáže kolekce fontů. Pokud jsme nepoužili ‘previous `\input`’, dopadne základní kolekce (implementovaná přímo uvnitř OFS) takto:

```

*\showfonts
OFS (1.0): The list of known font families:
defaults:
[CMRoman/]          \rm, \bf, \it, \bi, \sl
[CMSans/]           \rm, \bf, \it, -
[CMTypewriter/]     \rm, - , \it, - , \sl
[Times/]            \rm, \bf, \it, \bi
[Helvetica/]        \rm, \bf, \it, \bi, \nrm, \nbf, \nit, \nbi
[Courier/]          \rm, \bf, \it, \bi
```

Názvy fontových rodin jsou zde uvedeny v hranatých závorkách a vedle jsou uvedeny přepínače variant, které pro danou rodinu je možné použít.

Vidíme, že nejběžnější textové fonty Computer Modern jsou zaneseny v OFS do tří rodin `CMRoman`, `CMSans` a `CMTypewriter`. Myslím, že netřeba dodávat, co to znamená. V rodinách `CMSans` a `CMTypewriter` není k dispozici jinak obvyklá varianta `BoldItalic`, pro kterou je vyhrazen přepínač `\bi`. V rodině `CMTypewriter` nenajdeme ani přepínač `\bf` pro tučnou variantu, ale zato můžeme použít „nadstandardní“ variantu `\sl` (slanted), která je k dispozici i v rodině `CMRoman`.

Deklarace dalších písmových rodin jsou zanašeny do deklaračních souborů s příponou `tex`². V každém distribuci `TEXu` doporučuji udržovat soubor volající

²V `LATEX`ovém OFS se jedná o příponu `sty`.

všechny deklarační soubory `allfonts.tex`. Například na mém počítači vypadá tento soubor takto:

```
$ cat 'kpsewhich allfonts.tex'
%%% All OFS families declared on this TeX
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%%% Petr Olsak
\input a35      % PostScript 35
\input ffonts  % Another free fonts
\input btfonts % Bitstream fonts
\input skatalog % Stromtype foundry, 89 families
```

Když tedy napíšu `$ tex allfonts \showfonts \end | less`, dostanu na svém počítači výpis zhruba tří set písmových rodin, každá obvykle ve čtyřech variantách. Důležité je, že se v tom vyznám, a nemusím vzpomínat, jak se která metrika jmenuje. Následuje jen část výpisu.

```
OFS (1.0): The list of known font families:
defaults:
  [CMRoman/]      \rm, \bf, \it, \bi, \sl
  [CMSans/]       \rm, \bf, \it, -
  [CMTypewriter/] \rm, - , \it, - , \sl
  [Times/]        \rm, \bf, \it, \bi
  [Helvetica/]    \rm, \bf, \it, \bi, \nrm, \nbf, \nit, \nbi
  [Courier/]      \rm, \bf, \it, \bi
a35.tex:
  [AvantGarde/]   \rm, \bf, \it, \bi
  [Bookman/]      \rm, \bf, \it, \bi
  [NewCentury/]   \rm, \bf, \it, \bi
  [Palatino/]     \rm, \bf, \it, \bi
  [ZapfChancery/] \rm, - , \it, -
  [ZapfDingbats/] \rm, - , - , -
  [Symbol/]       \rm, - , \it, -
ffonts.tex:
  [Charter/]      \rm, \bf, \it, \bi
...
sjannon.tex:
  [JannonAntikva/] \rm, \bf, \it, \bi, \mr, \mi
  [JannonText/]    \rm, \bf, \it, \bi, \mr, \mi
  [JannonCaps/]    \rm, \bf, \it, \bi
...
sdynamo.tex:
...
  [DynaGroteskLE/] \rm, \bf, \it, \bi
  [DynaGroteskD/]  \rm, \bf, \it, \bi
  [DynaGroteskR/]  \rm, \bf, \it, \bi
...
stitul.tex:
  [Alcoholica/]    \rm, \bf, \it, -
  [Monarchia/]     \rm, \bf, - , -
```

```

[MonarchiaText/]      \rm, \bf, - , -
[Clichee/]            \rm, \bf, \it, \bi
[Regula/]             \rm, - , \it, -
[Splendid/]          \rm, \bf, \it, - , \script, \sans
[Cobra/]             \rm, \bf, - , -
[ExcelScript/]        \rm, - , - , - , \ext
[ExcelScriptText/]    \rm, - , - , - , \ext
[Zeppelin/]          \rm, \bf, - , - , \lr, \coll
[Negro/]             \rm, - , - , -
[Farao/]             \rm, \bf, - , - , \kr, \coll
...
slido.tex:
[Lido/]              \rm, \bf, \it, \bi, \crm, \cbf

```

Ve výpisu je uveden i T_EXový soubor, kde se deklarace odpovídajících rodin skutečně nachází. Vidíme tedy, že `a35.tex` už obsahuje deklarace rodin, ale například soubor `skatalog.tex` zřejmě obsahuje jen další `\input sjannon`, `\input sdynamo`, `\input stitul` atd. Když si pro svůj dokument vyberu třeba rodiny `JannonText`, `DynaGroteskR` a `Farao`, pak místo toho, abych psal

```
\input ofs [allfonts]
```

si vystačím s

```
\input ofs [sjannon, sdynamo, stitul]
```

což způsobí načítání podstatně méně souborů s deklaracemi fontových rodin.

Pokud potřebuji vědět, jak ty fonty vypadají, napíšu například:

```
$ pdfcspain ofscatal [sjannon, sdynamo, stitul] ; acroread ofscatal.pdf
```

Soubor `ofscatal.tex`, který vytvoří katalog, byl zařazen do balíku OFS od verze Oct 2002.

3. Přepínač rodin a velikostí

Hlavním příkazem OFS na uživatelské úrovni je `\setfonts`. Jedná se především o přepínač rodin, viz výpis `\fontusage` v předchozí sekci. Přepínač má dva parametry v hranaté závorce oddělené lomítkem. Prvním parametrem je název rodiny a druhým požadovaná velikost fontů. Pokud některý parametr chybí, přepínač nebude jeho nastavení měnit. Přepínač samozřejmě funguje lokálně ve skupinách, jako bývá u fontových přepínačů obvyklé. Po přepnutí rodiny pracují přepínače variant (obvykle `\rm`, `\bf`, `\it`, `\bi`) pro novou rodinu. Varianta se po přepnutí rodiny inicializuje stejná, jako byla před přepnutím, tj. jsme-li zrovna v kurzívě rodiny Times a přepneme do Helveticy, zůstáváme ve variantě `\it` rodiny Helvetica, tj. skloněné písmo. Pokud ale při přepnutí rodiny aktuální varianta v nové rodině neexistuje, inicializuje se varianta `\rm`, kterou musí obsahovat každá rodina.

Příklady

```
\setfonts [JannonText/10.5] % nastavím na začátku dokumentu
\setfonts [/14]\bf          % například pro nadpisy
\setfonts [/8]\rm           % pro poznámky pod čarou
\setfonts [DynagroteskR/]\it % třeba pro citáty
\setfonts [CMTypewriter/]   % pro strojopis.
% tento přepínač např. v kurzívě nastaví automaticky
% variantu kurzíva-strojopis.
```

Výhodou tedy je, že si nemusím pamatovat názvy matrik (v L^AT_EXu pak nesrozumitelné zkratky rodin používané v NFSS), ale píšu název rodiny do dokumentu stejným způsobem, jak jej vidím v písmovém katalogu. Pokud udělám v názvu rodiny překlep (třeba nedodržím velká a malá písmena), příkaz `\setfonts` spustí `\showfonts`, tj. na obrazovce a v logu vidím seznam všech rodin, které mohu použít.

Velikost fontů mohu nastavit přímo udáním v jednotce **pt**, nebo připsáním jiné jednotky (např. **mm**), nebo jako u primitivu `\font` pomocí slova **scaled** (koeficient zvětšení se váže k základní velikosti fontu) a konečně i pomocí zcela nového prefixu **mag**, který udává zvětšení vzhledem ke zrovna použité velikosti fontů, například:

```
\def\maly{\setfonts [/mag0.8]}
stanu se {\maly menším a {\maly menším a {\maly ještě menším}}}
a už jsem se z toho dostal.
```

dopadne takto:

stanu se menším a menším a ještě menším a už jsem se z toho dostal.

Prefix **mag** pro velikost využijeme například v logu L^AT_EX, které jsem si v plainu definoval jako

```
\def\LaTeX{L\kern-.2em\raise.45ex\hbox{\setfonts[/mag.7] A}\kern-.05em\TeX}
```

a funguje to v nadpisech (je tam tučně vyvýšené A odpovídající velikosti) i v poznámkách pod čarou. V L^AT_EXu samotném mají toto logo definováno tak, že se pro vyvýšené A použije font v indexové velikosti. No jo, zde jim to náhodou prochází, ale co kdyby se (například v jiném v logu) hodilo pro některé písmeno použít velikost **mag.8**, která není jako indexová velikost použita? Bez OFS bychom to pak dělali dost obtížně, protože NFSS nám řešení nenabízí.

Dalším využitím prefixu **mag** je možnost korigovat nestejně střední výšky písma. Například v tomto článku je pro strojopis použit **CMTypewriter**. Pokud by střední výška základního písma byla jiná, stačí pro přepínač `\tt` použít příkaz `\setfonts[CMTypewriter/mag1.1]`, kde 1.1 je korekční faktor.

Příkaz `\setfonts` může obsahovat i specifikaci varianty (viz výpis příkazu `\fontusage`). Pak se už nejedná o přepínač rodiny, ale fontu samotného. V takovém případě přepínač `\setfonts` neovlivní přepínače variant `\rm`, `\bf` a dalších, ale nastaví jen požadovaný jediný font. Například pro nadpisy by šlo psát

`\setfonts[-bf/14]` místo `\setfonts[/14]\bf`, ale museli bychom mít jistotu, že v nadpisu nepoužijeme přepínač varianty, protože ten by vrátil font do velikosti aktuální rodiny. Tou velikostí je třeba 10pt.

Abychom v nadpisu mohli použít přepínač `\it`, a přitom se dostali do varianty `\bi` (což je žádoucí), je potřeba přepínač fontu v makru pro nadpis naprogramovat zhruba takto:

```
\def\nadpis #1{{\setfonts[/14]\bf \let\it=\bi #1}}
```

Kvůli tomuto jednoduchému obratu, který zvládne snad každý plainista, jsem se rozhodl nekomplikovat jádro OFS podobným způsobem jako NFSS a nezavádět tedy další „nezávislou souřadnici“ popisující duktus fontu. Pro speciální rodiny fontů (jako je třeba rodina **DynaGrotesk**) jsem ale snadno pomocí doplňujících maker vytvořil přepínač, který respektuje ještě o jednu „nezávislou souřadnici“ více, než zvládá NFSS. V tomto případě to nebyl jen duktus, ale i stupeň zúžení písma. Navíc tento přepínač dokáže „poskakovat“ po jednotkách vpřed a vzad podél zvolené souřadnice. To také NFSS nezvládá.

Často se hodí vytvářet zkratky pro přepínače rodin. K tomu slouží makro `\fontdef\přepínač[Rodina/velikost]`, které funguje podobně, jako použití konstrukce `\gdef\přepínač{\setfonts[Rodina/velikost]}`. V záhlaví dokumentu může být například uvedeno:

```
\input ofs [ffonts]           % Charter je ve skupině free fonts
\setfonts [Charter/10pt]       % výchozí rodina
\fontdef\tt [CMTypewriter/mag1.1] % strojopis, korekce střední výšky
\fontdef\verbtt [CMTypewriter/8]  % strojopis pro display ukázky
\fontdef\small [Charter/9]       % zmenšení pro abstrakt a záhlaví
```

Možná čtenáře napadne, že jsem nemusel psát podruhé slovo **Charter** v deklaraci přepínače `\small`. Při kompletní změně základní rodiny dokumentu na jinou bych pak mohl změnit slovo **Charter** jen na jediném místě (u příkazu `\setfonts`) a byl bych hotov. Bohužel, vynechání rodiny v příkazu `\small` vede k problémům, protože tento příkaz je použit v záhlaví dokumentu, tj. v `\output` rutině. Pokud zde nevedeme explicitně rodinu, dědí se rodina aktuálně použitá, která ovšem může být jakákoli, protože `\output` rutina je vyvolávána z různých míst při zpracování dokumentu. Mít záhlaví jednou strojopisem a podruhé Charterem podle toho, odkud byla `\output` rutina zavolána, je samozřejmě nežádoucí.

Tento problém se dá řešit „vykřičníkovou“ konvencí, kterou umí zpracovat příkaz `\fontdef`:

```
\fontdef\small [!/9] \addcmd\small {\rm} % pro abstrakt a záhlaví
```

Vykřičník je nahrazen aktuální rodinou už v době činnosti příkazu `\fontdef`, nikoli tedy až v době provádění přepínače. V `\output` rutině se musíme ještě postarat o potlačení dědičnosti aktuální varianty. Proto jsem použil ještě příkaz `\addcmd` (zaveden nově od verze OFS Oct 2002), který k existujícímu makru

přidá další příkazy na jeho konec. Celá deklarace fontů v dokumentu tedy může vypadat zhruba takto:

```
\input ofs [ffonts] % Charter je ve skupině free fonts
\setfonts [Charter/10pt] % výchozí rodina
\fontdef\tt [CMTypeewriter/mag1.1] % strojopis, korekce střední výšky
\fontdef\verbtt [CMTypeewriter-rm/8] % strojopis pro display ukázky
\fontdef\small [!/9]
\addcmd \small {\baselineskip11pt\rm} % zmenšení pro abstrakt a záhlaví
\fontdef\fontsekcce [!/12]
\addcmd \fontsekcce {\bf \let\it=\bi} % pro nadpisy sekcí
\fontdef\fonttitul [!-bf/14.4] % v titulu nebudeme přepínat varianty
```

Povedlo se tedy veškeré aktivity související s výběrem fontů soustředit na jedno místo dokumentu bez nutnosti používat názvy metrik.

4. Kódování fontů

OFS pro plain implicitně pracuje v kódování $\mathcal{S}$ fontů, ale můžete jej přepnout do jiného kódování.³

Základem informace o kódování v OFS pro plain je makro `\fotenc`, které má implicitně hodnotu `8z`. To znamená, že rodiny `CMRoman`, `CMTypeewriter` a `CMSans` budou pracovat s $\mathcal{S}$ fonty. Kdyby uživatel nastavil `\def\fotenc{8t}`, začalo by OFS pracovat v případě těchto rodin s DC nebo EC fonty. Jaké je technické pozadí vysvětlíme v následující sekci.

Většina metrik PostScriptových fontů má varianty `*8t` (pro kódování podle Corku) a `*8z` (pro kódování podle $\mathcal{S}$ fontů). Makro `\fotenc` tedy musí obsahovat koncovku metrik, které chceme použít. Tím volíme kódování. Příklad:

```
\input ofs
\setfonts [Times/] text 1 % metrika: ptmr8z, tj. kódování  $\mathcal{S}$ fontů
\def\fotenc{8t}
\setfonts [/] text 2 % metrika: ptmr8t tj. kódování podle Corku
```

Máte-li své fonty v nějakém dalším kódování a koncovky metrik máte s názvy třeba `*8x`, pak není problém napsat `\def\fotenc{8x}` a můžete začít používat své vlastní metriky.

Pokud používáte kódování podle Corku, doporučuji definovat makro `\fotenc` jako `8t` před zavedením makra OFS. Při použití složitější deklarovaných rodin se to hodí.

V balíku OFS jsou soubory `ofs-8z.tex` a `ofs-8t.tex`, ve kterých je deklarace akcentů a některých na kódování závislých maker, jako například `\promile`. Implicitně není načten ani jeden z těchto souborů, tj. jsou respektována makra

³OFS pro L^AT_EX přenechává starost o kódování zcela na NFSS.

pro akcenty z originálního plainu, která expandují na `\accent` podle kódování CMfontů.

Pokud je toto chování nevyhovující, můžete načíst pomocí `\input` jeden nebo oba dva výše zmíněné soubory (v libovolném pořadí – jejich definice se nehádají). Pokud načtete oba soubory, pak makra pro akcenty expandují na znaky podle kódování, které je aktuálně nastaveno v makru `\fotenc` (8z nebo 8t).

Podívejme se do souborů `ofs-8z.tex` a `ofs-8t.tex` (výrazně kráceno):

```
%%% Default accents in CM
\accentdef \' * 8z {\accent 18 } % grave
\accentdef \' * 8z {\accent 19 } % acute
\accentdef \v * 8z {\accent 20 } % caron
\accentdef \u * 8z {\accent 21 } % breve
...
%%% Standard characters in plain (redefined here)
\def\aa{\r a}
\def\AA{\r A}
\characterdef \i 8z 16
\characterdef \j 8z 17
\characterdef \SS 8z {\SS}
\characterdef \AE 8z 29
...
%%% Extra characters from CS-fonts
\characterdef \promile 8z 141
\characterdef \varhyphen 8z 156
\characterdef \flqq 8z 158
\characterdef \frqq 8z 159
\characterdef \clqq 8z 254
\characterdef \crqq 8z 255
...
%%% Accented letters from CS fonts
\accentdef \' A 8z 152
\accentdef \' A 8z 193
\accentdef \" A 8z 196
\accentdef \' a 8z 184
\accentdef \' a 8z 225
...
%%% Default accents in Cork
\accentdef \' * 8t {\accent 0 }
\accentdef \' * 8t {\accent 1 }
\accentdef \^ * 8t {\accent 2 }
\accentdef \ * 8t {\accent 3 }
\accentdef \" * 8t {\accent 4 }
...
%%% Standard characters in plain (redefined here)
\def\aa{\r a}
\def\AA{\r A}
\characterdef \i 8t 25
```

```

\characterdef \j      8t  26
\characterdef \SS     8t 223
\characterdef \AE     8t 198
...
\characterdef \promile      8t {\%\char 24 }
\characterdef \textpertenthousand 8t {\%\char 24\char 24 }
...
%% Accented letters from Cork encoding
\accentdef \. i      8t 'i
\accentdef \u A      8t 128
\accentdef \k A      8t 129
\accentdef \' C      8t 130
\accentdef \v C      8t 131
...

```

Myslím, že syntaxe a význam příkazů `\characterdef` a `\accentdef` je v této ukázce samovysvětlující. Pokud ne, odkazují čtenáře do dokumentace [3], která do posledního detailu (včetně popisu jednotlivých fází expanze) vysvětluje činnost těchto příkazů.

L^AT_EXovému uživateli to může připomínat příkazy `\DeclareTextSymbol` a `\DeclareTextComposite` a jim podobné, které dělají zhruba totéž (jen poněkud komplikovaněji, nepřehledněji a těžkopádněji).

OFS také počítá s možností, že některé znaky jsou přidány do extra fontu, který s původním fontem vytváří uspořádanou dvojici. Takové dvojice metrik jsou použity například pro fonty Štormovy písmolijny, protože obsahují více než 256 znaků. Přitom by bylo škoda některé znaky nevyužít. Ke každé metrice těchto fontů (`*8z` i `*8t`) je přiřazena extra metrika `*6s`, obsahující zbylé znaky. Pomocí `\characterdef` a `\accentdef` můžeme deklarovat přítomnost těchto znaků v kódování `6s` (viz soubor `stormenc.tex`). Pokud aktuální metrika má k sobě extra font v kódování `6s` a je požadován znak z tohoto fontu, OFS jej automaticky „vyloví“ pomocí přechodného přepnutí na extra metriku.

5. Deklarace rodin

Podívejme se nyní do souboru `a35.tex`, který deklaruje rodiny pro základních 35 PostScriptových fontů (kráceno):

```

%% Times, Helvetica, Courier is in OFS defaults
\ofsdeclarefamily [AvantGarde] {% ----- AvantGarde
  \loadtextfam (Book)      pagk\fontenc;%
                        (Demi)      pagd\fontenc;%
                        (BookOblique) pagko\fontenc;%
                        (DemiOblique) pagdo\fontenc;;%
  \def\TeX{T\kern-.08em\lower.3333ex\hbox{E}\kern-0.09emX}}%
}

```

```

\ofsdeclarefamily [NewCentury] {% ----- NewCenturySchlbk
  \loadtextfam (Roman)      pncr\fotenc;%
                           pncb\fotenc;%
                           pncri\fotenc;%
                           pncbi\fotenc;%;
  \def\TeX{T\kern-.1667em\lower.5ex\hbox{E}\kern-.125emX}%
}
...

```

Zde je tedy deklarováno zobrazení mezi názvy rodin a metrikami.⁴ Každá rodina má čtyři povinné metriky (pro běžné čtyři varianty). Metriky jsou zapisány pomocí `\fotenc`, aby fungovalo přepínání kódování. Mezi předposledním a posledním středníkem může být uvedeno extra kódování, pokud je font rozložen do dvou metrik (v tomto příkladě není). Příkazy uvedené v parametru `\ofsdeclarefamily` se provedou při každém přepnutí rodiny příkazem `\setfonts`. V ukázce tam máme alternativní definice loga `\TeX`, aby toto logo pěkně vypadalo v každém fontu.

Od verze OFS Oct 2002 je přidána možnost použít nepovinné parametry v argumentech příkazu `\loadtextfam`. Tyto parametry píšeme do kulaté závorky a značí název varianty, pokud je odlišný od běžného názvu. Tato informace se použije v logu a při tisku katalogu.

Podívejme se ještě do souboru `sjannon.tex` z podpory Štormových fontů (výpis je zde neúplný):

```

\ofsdeclarefamily [JannonAntikva] {% ----- Jannon Antikva
  \loadtextfam  sjnr\fotenc;%
                sjnb\fotenc;%
                sjnri\fotenc;%
                sjnbi\fotenc;6s;%
  \newvariant2 \mr (Medium)      sjnm\fotenc;6s;%
  \newvariant3 \mi (MediumItalic) sjnmi\fotenc;6s;%
}
\ofsdeclarefamily [JannonText] {% ----- Jannon Text
  \loadtextfam  sjnrg\fotenc;%
                sjnbg\fotenc;%
                sjnrig\fotenc;%
                sjnbig\fotenc;6s;%
  \newvariant2 \mr (Medium)      sjnmg\fotenc;6s;%
  \newvariant3 \mi (MediumItalic) sjnmig\fotenc;6s;%
}
...

```

Vidíme, že navíc je zde deklarována extra metrika v kódování `6s` a že se zde deklarují doplňující přepínače variant `\mr` a `\mi` mimo základní čtyři varianty. Podrobnější vysvětlení použitých příkazů čtenář opět najde v dokumentaci [3].

⁴V OFS pro L^AT_EX takové soubory neexistují; tam se pouze deklaruje zobrazení mezi (dlouhými) názvy rodin v OFS a zkratkami rodin v NFSS.

Jak jsou pomocí expanze `\fotenc` mapovány metriky $\mathcal{C}$ Sfontů resp. DC fontů, které v názvech koncovku `8z` ani `8t` nemají? To souvisí rovněž s tím, že tyto fonty mají pro různé velikosti různé metriky. V deklaračních souborech je pak tento problém vyřešen způsobem „dvě mouchy jednou ranou“ (viz soubor `ofsdef.tex`):

```
\registertfm cmr8z      -      csr10  % metrika pro všechny velikosti
\registertfm cmr8z      0pt-6pt  csr5
\registertfm cmr8z      6pt-7pt  csr6
\registertfm cmr8z      7pt-8pt  csr7
\registertfm cmr8z      8pt-9pt  csr8
\registertfm cmr8z      9pt-10pt  csr9
\registertfm cmr8z      10pt-12pt  csr10
\registertfm cmr8z      12pt-17pt  csr12
\registertfm cmr8z      17pt-*    csr17
...
\registertfm cmr8t      -      dcr10   % metrika pro všechny velikosti
...
\ofsdeclarefamily [CMRoman] {% ----- Computer Modern Roman
  \loadtextfam cmr\fotenc;%
      cmbx\fotenc;%
      cmti\fotenc;%
      cmbxti\fotenc;%
  \newvariant8 \sl (Slanted) cmsl\fotenc;%
}
```

Centrálním příkazem je zde makro `\registertfm`, které mapuje neexistující metriky formálně vytvořené pomocí expanze `\fotenc` do skutečných metrik. Toto makro také umí registrovat různé metriky pro různé velikosti. Pokud tedy budeme chtít font daný pomocí `\setfonts[CMRoman-rm/12.5]`, vyvolá se při `\fotenc` s hodnotou `8z` metrika `csr12 at12.5pt`. „Metrika pro všechny velikosti“ se použije v případě, že je velikost fontu deklarovaná s prefixem `scaled`, takže `\setfonts[CMRoman-rm/scaled1250]` vyvolá metriku `csr10 scaled1250`. Ztrácíte-li přehled o tom, jaká metrika se ve skutečnosti použila, můžete zapnout logovací přepínač `\detailfontmessages`.

Podíváte-li se do souboru `ofsdef.tex` podrobněji, můžete si všimnout, že jsem metriky rodiny `CMRoman` a dalších CM rodin pro kódování `8t` poněkud odbyl. Registroval jsem je jako metriky DC fontů bez střídání metrik pro různé velikosti. Nechtělo se mi to totiž vypisovat a nevěděl jsem, zda uživatel tohoto kódování raději nepoužije EC fonty místo DC fontů. Ve zmíněném souboru jsou příklady, jak by se ta deklarace měla provést. Pokud ji někdo dopíše (například pro DC i EC fonty do zvláštních souborů), rád ji do balíku OFS zařadím. Osobně ale fonty kódované v `8t` nepoužívám, takže mě zatím nic nemotivuje to udělat.

6. Matematické fonty

Každý plainista ví, že fonty pro matematiku se sdružují do matematických rodin obsahující font pro základní, indexovou a index-indexovou velikost. Rovněž ví, že první čtyři matematické rodiny (s číslem 0 až 3) jsou s T_EXem jistým způsobem pevně významově svázány a další rodiny se dají deklarovat.

Aby plainista při deklarování matematických rodin nemusel psát třikrát za sebou `\font` a potom ještě `\textfont\rodina=...`, `\scriptfont...` atd., je v OFS k tomu vytvořena zkratka `\loadmathfam`, jak ukážu za chvíli.

Inicializaci matematických fontů v OFS pro plain provedeme pomocí příkazu `\setmath`. Dokud tento příkaz nepoužijeme, jsou matematické fonty ve stavu, jak je inicializoval plain. Ve třech parametrech příkazu `\setmath` (oddělených lomítky) dáváme najevo, jakou chceme základní velikost, jakou pro indexy a jakou pro indexy indexů. Jsou-li tyto parametry prázdné, je použito relativní zmenšení podle aktuální velikosti textového fontu pomocí prefixu `mag` takto:

```
\setmath [//] je totéž jako \setmath [mag1.0/mag.7/mag.5]
```

Příkaz `\setmath` vypočítá požadované velikosti a startuje makro pro zavedení fontů `\mathfonts`. Dále tento příkaz startuje makro pro inicializaci matematického kódování `\mathchars`. Tato makra si může plainista definovat jak chce, nicméně většinou využije již připravených maker `\defaultmathfonts` a `\defaultmathchars`, která jsou udělána tak, že spouštějí různé varianty kódu podle hodnoty maker `\fomenc` a `\mathversion`.

Při `\def\fomenc{PS}` (implicitní hodnota: PostScriptové fonty) se zavedou fonty tak, že matematická kurzíva se ztotožní s aktuální textovou kurzívou a podobně rodina 0 pro číslice a textové symboly zůstává nastavena podle aktuální textové rodiny varianty `\rm`. Matematické symboly se berou (pokud to je možné) z běžně dostupného PostScriptového fontu `Symbol`. Zbytek (např. natahovací závorky) pak zůstává v `Computer Modern`. Matematické kódování je pro tuto situaci výrazně pozměněno obvyklými primitivami `\mathchardef` a podobnými, aby byly dosažitelné všechny znaky plainu. Například pro řecká písmena musela být zavedena nová rodina se skloněným fontem `Symbol` a příkazy typu `\alpha` jsou překódovány z původní matematické rodiny 1 na tuto novou rodinu.

Nastavíte-li `\def\fomenc{CM}`, pak příkaz `\setmath` zavede stejné matematické fonty z rodiny `Computer Modern`, jako v plainu. Příkaz `\setmath[//]` pak pouze aktualizuje velikosti těchto fontů podle velikosti aktuálního textového fontu.

Nakoupíte-li fonty `MathTimes`, dále načtete soubor `ofsmtdf.tex` (součásti OFS od verze Jun. 2002) a definujete `\def\fomenc{MT}`, pak příkaz `\setmath` zavede kurzívu a rodinu 0 stejně jako při `\def\fomenc{PS}`, ale navíc použije pro *všechny* symboly včetně natahovacích závorek fonty `MathTimes`.

Kromě hodnoty `\fomenc` se příkaz `\setmath` větví i vzhledem k hodnotě makra `\mathversion`. Implicitně OFS počítá se dvěma hodnotami tohoto makra: `\def\mathversion{normal}` nebo `bold`. Při verzi `bold` jsou načteny do matematických rodin tučné alternativy fontů, pokud to jde. Vše názorně vysvětlí pohled do souboru `ofsdef.tex`:

```

\def\defaultmathfonts{\csname load\fomenc\mathversion math\endcsname}
\def\defaultmathchars{\csname set\fomenc mathchars\endcsname}
\def\mathfonts{\defaultmathfonts}
\def\mathchars{\defaultmathchars
  \let\mathchars=\relax % to protect the twice math-setting
}
\def\loadPSnormalmath{%
  \loadmathfam 0[tenrm]/%           Actual Roman font
  \loadmathfam 1[tenit]/%           Actual Italic font
  \loadmathfam 2[cmsy]/%            Standard symbols from CM
  \noindexsize\loadmathfam 3[tenex]/% Standard extra symbols from CM
  \chardef\itfam=1 \chardef\bifam=5
  \loadmathfam \bffam [tenbf]/%      Actual Bold font
  \loadmathfam \bifam [tenbi]/%      Actual Bold Italic
  \newmathfam\symbfam
  \loadmathfam \symbfam [/psyr]/%    PostScript Symbol
  \newmathfam\symbfam
  \loadmathfam \symbfam [/psyro]/%   PostScript Symbol Oblique
}
\def\loadPSboldmath{%
  \loadmathfam 0[tenbf]/%           Actual Bold font
  \loadmathfam 1[tenbi]/%           Actual Bold-Italic font
  ...
}
\def\loadCMnormalmath{%
  \loadmathfam 0[cmr8z]/%           Roman font
  \loadmathfam 1[cmmi]/%            Math Italic font
  \loadmathfam 2[cmsy]/%            Standard symbols from CM
  \noindexsize\loadmathfam 3[cmex10]/% Standard extra symbols from CM
  ...
}
\def\loadCMboldmath{%
  \loadmathfam 0[cmbr8z]/%           Roman font
  \loadmathfam 1[cmmb10]/%           Math Italic font
  ...
}
\def\setPSmathchars{%
  \fontmessage{\ofsmessageheader Math codes are set for PS encoding}%
  \mathcode'\,="602C
  \mathcode'\,="002E
  \delcode'\<="\hex\symbfam E130A
  \delcode'\>="\hex\symbfam F130B
  \delcode'\|="\hex\symbfam 7C30C

```

```

\edef\langle{\delimiter"4\hex\sybofam E130A }%
\edef\rangle{\delimiter"5\hex\sybofam F130B }%
\mathchardef\alpha "0\hex\sybofam 61
\mathchardef\beta "0\hex\sybofam 62
\mathchardef\gamma "0\hex\sybofam 67
...
}

```

Chceme-li přidat nějaké další matematické rodiny (v terminologii NFSS matematické abecedy), pak můžeme postupovat třeba takto:

```

\def\mathfonts{\defaultmathfonts
\newmathfam\bbfam
\loadmathfam \bbfam [/bbold12]% Dvojitá vertikální kresba
\def\bb{\fam\bbfam}%
}
\def\mathchars{\defaultmathchars
\mathchardef\balph "0\hex\bbfam 0B
\mathchardef\bbeta "0\hex\bbfam 0C
...
}

```

Příklad z konce sekce 3 nyní obohatíme o možnost práce s matematikou v libovolných velikostech:

```

\input ofs [ffonts] % Charter je ve skupině free fonts
\setfonts [Charter/10pt] % výchozí rodina
\setmath[/] % inicializace matematiky
\fontdef\tt [CMTypewriter/mag1.1] % strojopis, korekce střední výšky
\fontdef\verbtt [CMTypewriter-rm/8] % strojopis pro display ukázky
\fontdef\small [!/9] % zmenšení pro abstrakt a záhlaví
\addcmd \small {\baselineskip11pt \rm \def\mathversion{normal}\setmath[/]}
\fontdef\fontsekce [!/12] % pro nadpisy sekcí
\addcmd \fontsekce {\bf \let\it=\bi \def\mathversion{bold}\setmath[/]}
\fontdef\fonttitul [!-bf/14.4] % titul

```

Nyní i v abstraktech, poznámkách pod čarou a záhlavích je matematika ve všech velikostech všech možných indexů zmenšená odpovídajícím způsobem. Pokud by se matematika měla použít v názvu sekce, pak bude také tučná a správně veliká.

Reference

- [1] <http://www.pismolijna.cz>, <http://www.cstug.cz/stormtype>.
- [2] <ftp://math.feld.cvut.cz/pub/olsak/ofs>.
- [3] Petr Olšák. *OFS: Olšákův fontový systém*. 2001. Dokumentace k balíku je v souborech `ofsdoc.tex`, `ofsdoc.pdf`.

- [4] Petr Olšák. *Jak T_EX k fontům ze Střešovic přišel*. Zpravodaj Československého sdružení uživatelů T_EXu, 4/2001, strany 153–180.

Summary: Font Management Macros OFS by Olšák

OFS is the T_EX macropackage that is able to manage big font installation and includes support for handling and manipulation of font collections. Fonts of Typokatalog 3.5 by Střešovická písmolijna foundry were used as motivation and testbed for package development. First version was designed for plainT_EX only. At L^AT_EX users' requests, the same functionality was implemented for L^AT_EX too. Basic user OFS commands are same in both plainT_EX and L^AT_EX versions of OFS (L^AT_EX version is implemented as layer over NFSS with totally different macro code).

Petr Olšák
Elektrotechnická fakulta ČVUT, Praha
Email: olsak@math.feld.cvut.cz

Odborné knihy mají obvykle pevnější strukturu než beletrie. Tato struktura nebývá samoúčelná, ale často je důležitou nositelkou informace. Ač L^AT_EX jistým způsobem vede zkušené uživatele k užívání strukturního značkování, není zachování informace vždy triviální a vyžaduje programátorský způsob ke zpracování dokumentu. Jisté potíže vzniknou i při tvorbě neobvyklých rejstříků, kde se naráží na omezení programu MakeIndex. XML naproti tomu definuje strukturu dokumentu. Navíc přináší hotové nástroje pro různé typy zpracování. Pro znalce L^AT_EXu je pak přirozenou cestou využití XSLT a transformovaný text zpracovat L^AT_EXem jako sázecím strojem. Je demonstrována případová studie počínaje návrhem DTD přes transformační styl až po L^AT_EXová makra užitá k sazbě. Je provedeno srovnání se zpracováním obdobné knihy, kdy nebylo využito výhod XML.

Klíčová slova: XML, L^AT_EX, XSLT, DTD, validace, schéma

1. Úvod

Při zpracování dokumentů zjišťujeme, že jejich text je určitým způsobem členěn. Toto členění obvykle plní určitou funkci. Při tisku takového dokumentu pak musíme členění reprezentovat graficky.

Finální grafickou úpravu nemusíme vždy znát na počátku práce s dokumentem. S výhodou tedy využijeme nějaký sázecí systém, který dokáže separovat obsah a formu. Možností máme několik.

L^AT_EX (a vlastně i plain T_EX) umožňuje definovat kontextové značky, jimž až dodatečně přidělíme vizuální podobu. Pokud se během zpracování změní požadavky na grafickou úpravu, nemusíme zasahovat do textu, ale upravujeme pouze balík podpůrných maker.

Na zachycení obsahu dokumentu nezávisle od formy jsou však primárně určeny SGML a XML. Výhodou je, že k těmto formátům existuje řada volně šiřitelných i komerčních nástrojů. Navíc lze dokument snadno přeformátovat do několika různých podob. V tomto příspěvku ukážeme, že použití XML je výhodné i v případě, kdy ze vstupního souboru generujeme pouze jediný výstup.

¹Článek byl poprvé publikován ve sborníku Jan Kasprzak, Petr Sojka: SLT 2002, KONVOJ 2002, ISBN 80-7302-043-2, na stranách 27–48. Publikováno se souhlasem vydavatele a autorů.

2. Sazba XML souboru

XML definuje strukturu dokumentu, ale neurčuje vizuální vzhled. Ten lze také určit pomocí nástrojů XML, tzv. formátovacích objektů. Přestože již existují nezávislé nástroje, které provádějí sazbu přímo na základě formátovacích objektů, nejsou v typografické kvalitě schopny konkurovat $\text{\TeX}$.

V $\text{\TeX}$ ovém světě již existují nástroje pro přímou sazbu XML, např. $\text{\PassiveTeX}$ [8] a $\text{\ConTeXt}$. Jejich využitelnost však není univerzální. Může se stát, že z daného XML dokumentu chceme tisknout jen určité části, případně se mají tisknout v jiném pořadí, než v jakém se v původním souboru vyskytují. Tyto operace zvládne snadno XSLT. Zkušený uživatel $\text{\TeX}$ u či $\text{\LaTeX}$ u si nyní zákonitě položí otázku, proč by po transformaci mělo následovat zpracování s využitím některého z výše uvedených nástrojů, když výstupem transformace může být stejně tak $\text{\TeX}$ ový soubor a o jeho formátování se mohou postarat $\text{\TeX}$ ová makra.

Při transformaci z XML do $\text{\TeX}$ u se můžeme rozhodnout, jakou formátovací práci provede transformační styl a co svěříme až $\text{\TeX}$ ovým makrům. Prvním extrémem je, že při transformaci pomocí XSLT převedeme vše až na $\text{\TeX}$ ové příkazy nízké úrovně. Opačným extrémem je, že veškerou práci ponecháme $\text{\LaTeX}$ u. XSLT zpracuje pouze základní elementy a vše ostatní se bude formátovat s využitím $\text{\LaTeX}$ ového balíku `KEYVAL`, přičemž atributy jsou konvertovány na nepovinné parametry prostředí. Transformační styl pak může vypadat např. takto:

```
<?xml version="1.0" encoding='cp852'?>

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                version="1.0"
                xmlns:saxon="http://icl.com/saxon"
                extension-element-prefixes="saxon">
<xsl:output indent="no" method="text" encoding="cp852"
            saxon:character-representation="native"/>

<xsl:strip-space elements="*" />

<!-- Root -->
<xsl:template match="/">
  \documentclass{book}
  \usepackage{keyval}
  \usepackage{mujstyl}
  \begin{document}
    <xsl:apply-templates/>
  \end{document}
</xsl:template>
```

```

<!-- Texty -->
<xsl:template match="text()">
  <xsl:value-of select="."/>
</xsl:template>

<!-- Odstavec -->
<xsl:template match='p'>
  <xsl:apply-templates/>
  <xsl:text>&#10;&#10;</xsl:text>
</xsl:template>

<!-- Tučně -->
<xsl:template match='b'>
  <xsl:text>\textbf{</xsl:text>
  <xsl:apply-templates/>
  <xsl:text>}</xsl:text>
</xsl:template>

<!-- Kurzíva -->
<xsl:template match='i'>
  <xsl:text>\textit{</xsl:text>
  <xsl:apply-templates/>
  <xsl:text>}</xsl:text>
</xsl:template>

<!-- Ostatní -->
<xsl:template match='*'>
  <xsl:value-of select='concat("\begin{", name(), "}")' />
  <xsl:if test='@*'>
    <xsl:text>[</xsl:text>
    <xsl:apply-templates select='@*' />
    <xsl:text>]</xsl:text>
  </xsl:if>
  <xsl:apply-templates/>
  <xsl:value-of select='concat("\end{", name(), "}")' />
</xsl:template>

<!-- Atributy -->
<xsl:template match='@*'>
  <xsl:if test='position()>1'>
    <xsl:text>,</xsl:text>
  </xsl:if>
  <xsl:text>{</xsl:text>
  <xsl:apply-templates/>
  <xsl:text>}</xsl:text>
</xsl:template>

```

```

</xsl:if>
<xsl:value-of select='concat(name(), "=", .)'/>
</xsl:template>

</xsl:stylesheet>

```

Skutečně použitý styl pak bude někde mezi těmito dvěma extrémy. Co provede XSLT a co se ponechá L^AT_EXovým makrům, bude záležet především na schopnostech a zkušenostech každého programátora. Můžeme se ovšem také rozhodnout pro některý z běžně užívaných formátů XML a využít transformačních stylů, které jsou již hotové.

3. XML, pro a proti

Viděli jsme, že oproti přímému vytvoření dokumentu v L^AT_EXu máme při použití XML jednu operaci navíc, a to transformaci. Měli bychom se tedy zamyslet, zda nám tato práce navíc přinese nějaký užitek.

Obvyklým argumentem pro použití XML je snadnost převedení dokumentu do několika různých tvarů (PDF, HTML, tištěná podoba...). XML má však jisté výhody i v případě, že požadujeme jediný výstup.

Jedna z výhod se projeví při sazbě odborných knih, které mají pevně danou strukturu. Struktura je zde též nositelkou informace a může sloužit pro následné zpracování. Předpokládáme totiž, že životnost knihy nekončí jejím vydáním, ale její obsah může v budoucnu sloužit k účelům, které v okamžiku sazby ještě neznáme. Můžeme namítnout, že i z L^AT_EXového souboru lze získat dodatečné informace, pokud byl vhodně kontextově značkován, ale vždy je to mnohem náročnější než využití XSLT.

V učebnici matematiky lze věty a důkazy značkovat způsobem:

```

<věta číslo='1'>
  <text>
    Sem přijde text věty.
  </text>
  <důkaz>
    Zde bude důkaz
  </důkaz>
</věta>

```

V budoucnu lze pak z knihy vytáhnout např. jen věty a jejich důkazy. Podobně můžeme využít strukturu:

```

<cvičení číslo='1'>
  <příklad>
    Zadání příkladu.
  </příklad>
</cvičení>

```

```
</příklad>
<řešení>
  Řešení, které bude uvedeno jinde.
</řešení>
</cvičení>
```

Tak máme příklad i jeho řešení v dokumentu na stejném místě, což sníží pravděpodobnost zavlečení chyb. Pomocí XSLT snadno v hlavní části knihy vytiskneme jen zadání příkladu a v závěrečné kapitole jeho řešení.

Jinou aplikací je vytažení hypertextových odkazů z knihy, aby mohly být zveřejněny na WWW stránkách. Příklad najdete na WWW stránkách věnovaných knize *XML pro každého* [4].

Další výhoda se projeví, pokud v dokumentu potřebujeme velmi nestandardní rejstříky. Program MakeIndex má svá omezení. Někdy bychom potřebovali buď netriviální zásah do jeho zdrojového kódu, nebo pomocný preprocesor. Ukázka bude uvedena později v případové studii. Vzhledem k tomu, že XSLT umí abecední řazení, může být využití XSLT při tvorbě rejstříků snadnější.

XML narozdíl od L^AT_EXu umožňuje definovat strukturu dokumentu a ověřit jeho správnost. O metodách validace se podrobněji zmíníme později.

Při psaní českých textů v XML můžeme narazit na problém s nezlomitelnou mezerou za jednopísmennými neslabičnými předložkami. Řešení však není složité. Lze použít program VLNA či jeho ekvivalent a následně nahradit tildu entitou ` `.

4. Volba typu dokumentu

Na začátku návrhu dokumentu v XML musíme zvolit způsob jeho značkování. Velmi rozšířeným standardem, používaným v mnoha dokumentech, je DocBook [1]. Pro jeho použití v konkrétním případě se však nerozhodujeme podle toho, zda je dobrý či špatný, zda je používán zřídka či často, ale podle toho, zda podporuje struktury, které náš dokument obsahuje, případně, zda se dá snadno modifikovat, aby vyhověl našim značkovacím požadavkům. DocBook obsahuje základní elementy, které se používají ve všech dokumentech, a dále řadu dalších elementů potřebných pro různé technické dokumenty. DTD je navrženo tak, aby jej bylo možno poměrně snadno doplňovat a modifikovat. Přesto se v praxi setkáme s případy, kdy DocBook nabízí příliš mnoho elementů, které vůbec nepotřebujeme, a zcela chybí elementy, které jsou pro danou aplikaci naprosto nepostradatelné. Pak je vytvoření nového DTD nejpřirozenějším řešením.

5. DTD ano či ne?

Výhodou XML oproti SGML je, že pro zpracování dokumentu často nepotřebujeme DTD. Obvykle si vystačíme s definicemi různých entit, které můžeme vložit přímo do dokumentu. Ač je deklarace entit součástí DTD, není to takové DTD, jaké známe z SGML. Navrhujeme-li tedy strukturu dokumentu pro jedno použití, nemusíme se s vytvářením DTD zdržovat. Praxe však ukazuje, že DTD je užitečné i v těchto případech.

V první řadě si musíme uvědomit, že člověk při psaní dokumentu chybuje. Při zapisování tagu lze udělat překlep. Nemáme-li textový editor, který doplňuje ukončovací tagy, můžeme je zapomenout nebo v nich udělat překlep. Editor, který doplňuje uzavírací tagy párově k počátečním, je dobrý sluha, ale může být i zlým pánem. Předpokládejme, že místo `<emphasis>` namíšeme omylem `<emhpasis>`. „Chytrý“ editor nám pak na konec zvýrazněné fráze doplní `</emhpasis>`. Parser, který neprovádí validaci, žádnou chybu nezaznamená. Chyba se projeví až při zpracování dokumentu. XSLT procesor má obvykle defaultní pravidlo pro případ, že žádná explicitní šablona nebyla nalezena. Takto chybně zadaný element se tedy zpracuje „nějak“ a zvláště v případech rozsáhlých dokumentů a komplikovaných stylů nemusí být snadné odhalit příčinu, proč je část textu zpracována zcela jinak, než bylo zamýšleno. Diagnostické zprávy validujících parserů jsou sice někdy poněkud kryptické, ale mnohem rychleji vedou k určení zdroje chyby.

Druhou výhodou validace oceníme v okamžiku, kdy začneme programovat styl, jenž s dokumentem provádí složitější transformaci. DTD nám totiž naprosto jasně říká, jaké situace mohou nastat. Víme tudíž naprosto přesně, jaké případy musíme v transformačním stylu ošetřit, a čím se zabývat nemusíme, protože to validující parser nepřipustí. Autor ze své zkušenosti ví, že návrh transformačního stylu je snadnější, když si vezmeme DTD a postupně podle něj vytváříme jednotlivé šablony.

6. Alternativní metody validace

Je obecně známo, že DTD určuje pouze základní pravidla pro strukturu dokumentu. V praxi často potřebujeme podstatně přesnější vymezení typu a obsahu elementů i jejich atributů. Tento problém se snaží řešit *schéma*. Bohužel nástroje, které využívají schéma pro validaci, nejsou ještě dostatečně rozšířené. Navíc ani tak nejsou ošetřeny všechny možnosti, s nimiž se můžeme setkat. Nezbyvá tedy jiná možnost než přistoupit k validaci vlastními prostředky. V principu je možné využít některou z knihoven pro práci s XML soubory a napsat si speciální validátor. Často je však vhodnější vytvořit validátor pomocí XSLT. Tento přístup je

využíván i v CML (Chemical Markup Language) v projektu SELFML (Standard Electronic File Format Markup Language) [5].

Vezměme si jako příklad XML soubor, který obsahuje výsledky matematického zpracování jistých fyzikálně chemických měření. Soubor odpovídá následujícímu DTD:

```
<!ELEMENT rset (result+)>

<!ELEMENT result (system,concentration,T,parameters,table,sigma)>

<!ELEMENT table (row+)>

<!ELEMENT row (cell+)>

<!ELEMENT cell (#PCDATA)>

<!ELEMENT system (#PCDATA)>

<!ELEMENT concentration (#PCDATA)>

<!ELEMENT T (#PCDATA)>
<!ATTLIST T unit CDATA 'K'>

<!ELEMENT sigma (#PCDATA)>

<!ELEMENT parameters EMPTY>
<!ATTLIST parameters C CDATA #REQUIRED
                      D CDATA #REQUIRED>
```

Z DTD je patrné, že každý `<result>` obsahuje právě jednu tabulku. V DTD však nelze zapsat jednu podstatnou podmínku, kterou nepodporuje ani schéma. Tabulka totiž obsahuje obvykle jeden řádek, který má druhý sloupec prázdný. Takový řádek nás při dalším zpracování nebude zajímat. Podmínkou však je, aby všechny tabulky, po vynechání řádku s prázdným druhým sloupcem, měly identický počet řádků. Druhou podmínkou je, že hodnoty v prvním sloupci všech tabulek musí být shodné.

Čtenář by mohl namítnout, že jsme říkali, že soubor byl vytvořen počítačem. Validace by tudíž neměla být nutná, protože generované soubory jsou správné. Ve skutečnosti je však nepřehledná řada příčin, které vedou k tomu, že programově vytvořený soubor je vadný. Dokud není program dokonale otestován, není jisté, že v něm není nějaká chyba. Validace výsledného XML souboru je jedním z užitečných kroků při testování programu. A nesmíme ani zapomenout na to, že původní vstupní data vytvářel člověk. V hodnotách prvního sloupce tedy

mohou být překlipy² a také mohlo dojít k tomu, že se jeden řádek v některé tabulce ztratil. Podmínky tohoto typu neřeší ani schéma, ale k validaci lze použít následující styl:

```
<?xml version="1.0" encoding='cp852'?>

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                version="1.0"
                xmlns:saxon="http://icl.com/saxon"
                extension-element-prefixes="saxon">
<xsl:output indent="no" method="text" encoding="cp852"
            saxon:character-representation="native"/>

<xsl:strip-space elements="*" />

<!-- All elements -->
<xsl:template match="/*">
  <xsl:apply-templates/>
</xsl:template>

<!-- Text contents of all elements -->
<xsl:template match="text()"/>

<!-- Result -->
<xsl:template match='result[position()>1] '>
  <xsl:apply-templates>
    <xsl:with-param name='pos'>
      <xsl:value-of select='position()' />
    </xsl:with-param>
  </xsl:apply-templates>
</xsl:template>

<xsl:template match='result' />

<!-- Table -->
<xsl:template match='table'>
  <xsl:param name='pos' />
  <xsl:if test=
    'not(count(row[not(cell[position()=2]="")])=
```

²Zdálo by se, že by bylo jednodušší, kdyby data pro první sloupec byla zadána pouze jednou. Z povahy měření a následného zpracování, jejichž vysvětlení je nad rámec tohoto příspěvku, však plyne, že by to způsobilo laborantce mnoho těžkostí a spíše by to vedlo k zavlečení dalších chyb.

```

        count(/rset/result[position()=$pos]/table/row
        [not(cell[position()=2]="")]))'>
<xsl:text>Nesouhlasí počet řádků: </xsl:text>
<xsl:value-of
    select='count(row[not(cell[position()=2]="")])' />
<xsl:text> != </xsl:text>
<xsl:value-of
    select='count(/rset/result[position()=$pos]/table/row
    [not(cell[position()=2]="")])' />
<xsl:text>&#10;</xsl:text>
</xsl:if>
<xsl:apply-templates>
    <xsl:with-param name='pos'>
        <xsl:value-of select='$pos' />
    </xsl:with-param>
</xsl:apply-templates>
</xsl:template>

<!-- Row -->
<xsl:template match='row'>
    <xsl:param name='pos' />
    <xsl:variable name='rpos'>
        <xsl:value-of select='position()' />
    </xsl:variable>
    <xsl:if test=
        'not(cell[1]=/rset/result[$pos]/table/row[$rpos]/cell[1])'>
        <xsl:text>Nesouhlasí hodnota v 1. sloupci: </xsl:text>
        <xsl:value-of select='cell[1]' />
        <xsl:text> != </xsl:text>
        <xsl:value-of
            select='/rset/result[$pos]/table/row[$rpos]/cell[1]' />
        <xsl:text>&#10;</xsl:text>
    </xsl:if>
</xsl:template>

</xsl:stylesheet>

```

Ve stylu jsou zajímavé dvě věci. První z nich je potlačení výstupu textových uzlů. Druhou zvláštností jsou dvě šablony pro element **<result>**. Předpokládáme totiž, že tabulka v prvním výsledkovém bloku je správná, proto ji nekontrolujeme. Všechny ostatní musí souhlasit s první tabulkou.

Tím ovšem nejsou vyčerpány všechny možnosti, s nimiž se můžeme setkat. Některé chyby v XML souborech nelze odhalit žádným strojově vyjádřitelným algoritmem. K jejich nalezení nám ovšem zase pomůže XSLT. Kromě transformací se dá XSLT využít i jako dotazovací jazyk. Můžeme si tedy vyhledat a vypsat podezřelá místa a pak rozhodnout, zda je obsah správný či špatný. Konkrétní příklad bude uveden později.

7. Případová studie

Možnosti využití XML a L^AT_EXu jsou demonstrovány na knize *Vinohradský hřbitov včera & dnes* [6]. Její styl pro transformaci do L^AT_EXu má však více než 900 řádků a L^AT_EXová makra mají přes 250 řádků. Jejich plný kód zde proto nebude uveden a zaměříme se pouze na vyzdvížení hlavních výhod, které použití XML přineslo.

7.1. Pracovní cyklus

Kniha tohoto typu vzniká poměrně dlouho. Autor, který vzhledem ke svému věku počítač nepoužívá, zapisuje text obvykle na kartičky. Z této kartotéky písárka text knihy opíše. Navíc každý ze spoluautorů pracoval na jiné části (jež se však prolínají). Písárka musí tyto části vhodně spojit. Z toho pak plyne, že při prvních korekturách se provádí poměrně mnoho změn. Kniha je sázena dvousloupcově a do textu jsou vsazovány fotografie. Pravidla pro jejich umístění jsou dosti přísná a mechanismus plovoucích objektů nelze využít. Fotografie jsou proto vkládány až při poslední korektuře. Při jednotlivých fázích zpracování tedy dochází k posunu materiálu na jiné stránky. Odkazy uvnitř knihy musíme proto řešit nějakým automatem.

Odkazy na čísla stran se týkaly zejména rejstříků, jež byly dost nestandardní. Autoři je sice zpracovali ručně, ale opakované dopisování správných čísel stran by jistě vedlo k mnoha chybám. Jak se později ukáže, právě při tvorbě rejstříků se projevila jedna z výhod XML.

Text knihy přepisovala externí písárka a posílala po částech elektronickou poštou. Protože písárka nemá žádný nástroj pro práci s XML (text psala ve Wordu), nemá T_EX ani Acrobat Reader, byl nakonec vytvořen i soukromý styl pro transformaci do HTML. Pokud bylo nutno, aby písárka něco rychle ověřila v původní kartotéce, mohla se podívat na privátní WWW stránky se zpracovanou částí knihy. Klikací verze v HTML také významně pomohla při korekturách.

7.2. Tvoříme DTD

K potřebě vytvoření DTD jsem dospěl hned na počátku práce, kdy písárka (která se při přepisování tohoto rukopisu teprve začínala učit XML) udělala

první chybu. Základní struktura byla zřejmá, ale postupem času se DTD mírně měnilo a rostlo.

Zmíněna kniha má dva úvody, hlavní část, dva rejstříky, seznam literatury a tiráž. Pouze hlavní část a rejstříky byly generovány z XML, zbývající části byly napsány přímo v L^AT_EXu. DTD bylo proto děláno jen pro část dokumentu.

V úvodu bychom se měli zmínit, že parser sice komentáře ignoruje, ale musí je přečíst, aby našel jejich konec. I když pro tagy použijeme pouze písmena anglické abecedy, musíme definovat použité kódování kvůli česky psaným komentářům. DTD je také XML soubor a pokud v něm píšeme česky, musíme v něm použít XML deklaraci. Protože DTD je poměrně krátké, uvedeme je celé:

```
<?xml version='1.0' encoding='cp852'?>
<!-- DTD pro hřbitov -->

<!-- Znakové entity -->
<!ENTITY otilde "&#245;">
<!ENTITY egrave "&#232;">

<!-- Parametrické entity -->
<!-- ===== -->

<!-- Obsah hrobů -->
<!ENTITY % attr
    "jmeno    CDATA #REQUIRED
    krestni   CDATA #IMPLIED
    narozen   CDATA #IMPLIED
    smrt      CDATA #IMPLIED
    titul     CDATA #IMPLIED
    prefix    CDATA #IMPLIED
    hodnost   CDATA #IMPLIED">

<!-- Ano/Ne -->
<!ENTITY % ano "(ano|ne) 'ano'">

<!-- Struktura dokumentu,
    hřbitov má nejméně jedno oddělení -->
<!ELEMENT hrbitor (oddeleni|tex)+>

<!-- Oddělení jsou číslována,
    obvykle začínají plánkem a mají hroby -->
<!ELEMENT oddeleni (hrob|img|tex)+>
<!ATTLIST oddeleni cislo CDATA #REQUIRED
```

```

        planek CDATA #IMPLIED
        extra CDATA #IMPLIED
        popis (vedle|pres|sloupec) 'vedle'
        space CDATA #IMPLIED>

<!-- Obrázek, většinou fotografie -->
<!ELEMENT img (#PCDATA)>
<!ATTLIST img height CDATA #IMPLIED
              width CDATA #IMPLIED
              scale CDATA #IMPLIED>

<!-- Oddělení má hroby, většinou číslované,
      výjimečně obsahuje obrázky -->
<!ELEMENT hrob (#PCDATA|clovek|rodina|umelec|p|tex|em|img)*>
<!ATTLIST hrob cislo CDATA #IMPLIED>

<!-- Rodina má jenom lidi -->
<!ELEMENT rodina (clovek+)>
<!ATTLIST rodina odstavec %ano;>

<!-- Označíme, zda člověk patří do seznamu
      a zda se začíná nový odstavec -->
<!ELEMENT clovek (#PCDATA|hide)*>
<!ATTLIST clovek %attr;
              tecka %ano;
              seznam %ano;
              odstavec %ano;>

<!ELEMENT hide (#PCDATA)>

<!ELEMENT umelec (#PCDATA)>
<!ATTLIST umelec %attr;>

<!-- Odstavec -->
<!ELEMENT p EMPTY>

<!-- Specifické příkazy pro TeX -->
<!ELEMENT tex (#PCDATA)>

<!-- Text zvýrazněný proložením -->
<!ELEMENT em (#PCDATA)>

```

Na začátku DTD si definujeme entity pro znaky, které nemáme v české abecedě, a parametrické entity, jež budeme dále v DTD využívat. Zbytek DTD je komentován, ale přesto potřebuje několik poznámek.

Členění do odstavců, jak je běžné v HTML i v DocBooku, by zde přidělovalo práci a nebylo by účelné vzhledem k tomu, že odstavec zde není podstatnou stavební jednotkou textu. Proto používáme značku `<p/>` s významem *předěl odstavců*.

Element `<oddeleni>` má několik atributů. Text každého oddělení začíná plánkem a z prostorových důvodů se plánek sází jako perex s popisem buď umístěným vedle nebo přes plánek (když není obdélníkový a je v něm volné místo), jindy se sází přímo do sloupce. Umístění popisu je specifikováno atributem `popis`. Text oddělení nezačíná vždy na nové stránce. Prostředí MULTICOL může mít problémy, když perex je příliš vysoký. Řeší se to nepovinným parametrem, kdy tomuto prostředí sdělíme, že má přejít na novou stránku, je-li na stránce aktuální méně místa než zadaná hodnota. Tuto hodnotu zadáváme v atributu `space`. Atribut `extra` obsahuje volitelný L^AT_EXový kód, který ve výjimečných případech potřebujeme vložit.

Komentář si zaslouží elementy `<clovek>` a `<umelec>`. První z nich obsahuje údaje o pohřbené osobě. Protože jsou někteří pohřbení jen zmíněni, vyskytují se i uprostřed odstavce a nemají být uvedeni v rejstříku, máme pro tyto účely vytvořeny atributy. Obsahem elementu `<clovek>` je nějaká charakterizace, obvykle profesní (lékař, továrník, hudební skladatel apod.) příslušné osoby. Element `<umelec>` slouží k vyznačení autorů funerálního umění. Téměř vždy se v textu vyskytují ve standardní podobě, kterou vytvoříme programově, ale v některých případech je jméno uvedeno uprostřed věty v jiném než prvním pádě. Pak je požadovaný tvar uveden v obsahu tohoto elementu, ale ve většině případů je element `<umelec>` prázdný.

7.3. Pomocné nástroje

V části 6 jsme se zmínili o tom, že k validaci souboru lze využít XSLT. V předchozí části jsme uvedli, že element `<umelec>` je obvykle prázdný. Protože písárka občas omylem do tohoto elementu uzavřela text, který tam nepatří, bylo nutno najít nějaký systém, který by správnost elementu `<umelec>` ověřil. Proto byl vytvořen následující styl:

```
<?xml version="1.0" encoding="cp852"?>
```

```
<!--
```

```
Tento styl vypisuje umělce, kteří mají neprázdný obsah.
```

```
To je většinou špatně.
```

```
-->
```

```

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                version="1.0"
                xmlns:saxon="http://icl.com/saxon"
                extension-element-prefixes="saxon">

<xsl:output indent="no" method="text" encoding="cp852"
            saxon:character-representation="native"/>

<xsl:strip-space elements='*'/>

<!-- Použij šablony na předky elementu <umelec> -->
<xsl:template match='||hrbitov|oddeleni|hrob'>
    <xsl:apply-templates/>
</xsl:template>

<!-- Pokud je obsah elementu neprázdný,
      vypiš jej a uveď informace pro jeho nalezení -->
<xsl:template match='umelec'>
    <xsl:if test=".!='' ">
        <xsl:value-of select='saxon:systemId()' />
        <xsl:text> line </xsl:text>
        <xsl:value-of select='saxon:lineNumber()' />
        <xsl:text>&#10;</xsl:text>
        <xsl:value-of select='@jmeno' /><xsl:text>, </xsl:text>
        <xsl:value-of select='@krestni' /><xsl:text>: </xsl:text>
        <xsl:value-of select='../@cislo' /><xsl:text></xsl:text>
        <xsl:choose>
            <xsl:when test='../@cislo">
                <xsl:value-of select='../@cislo' />
            </xsl:when>
            <xsl:otherwise>
                <xsl:text>bez čísla --- </xsl:text>
                <xsl:value-of select='saxon:path()' />
            </xsl:otherwise>
        </xsl:choose>
        <xsl:text>&#10;</xsl:text>
        <xsl:value-of select='normalize-space(.)' />
        <xsl:text>&#10;&#10;</xsl:text>
    </xsl:if>
</xsl:template>

<!-- Ignoruj všechno ostatní -->

```

```
<xsl:template match='*|@*|text()' />
```

```
</xsl:stylesheet>
```

Styl využívá některá rozšíření XSLT procesoru Saxon [3]. Díky tomu nám sdělí přesné místo neprázdného elementu `<umelec>` a my se rozhodneme, zda je to správně, nebo zda to vyžaduje opravu.

Jiným typem chyby byl zapomenutý element `<clovek>`. Za určitých okolností totiž nemusel být prvním uzlem v elementu `<hrob>` a validace pomocí DTD pak takovou chybu neodhalí. Lze to však řešit pomocí XSLT:

```
<?xml version="1.0" encoding="cp852"?>
```

```
<!--
```

Tento styl vypisuje hroby, kde není žádný člověk.

```
-->
```

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                version="1.0"
                xmlns:saxon="http://icl.com/saxon"
                extension-element-prefixes="saxon">
```

```
<xsl:output indent="no" method="text" encoding="cp852"
            saxon:character-representation="native"/>
```

```
<xsl:strip-space elements='*' />
```

```
<xsl:template match='/*'>
```

```
  <xsl:apply-templates/>
```

```
</xsl:template>
```

```
<!-- Všechny elementy <hrob>, v nichž není
```

```
      žádný element <clovek> -->
```

```
<xsl:template match='hrob[count(./clovek)=0]'>
```

```
  <xsl:value-of select='saxon:systemId()' />
```

```
  <xsl:text> line </xsl:text>
```

```
  <xsl:value-of select='saxon:lineNumber()' />
```

```
  <xsl:text>: </xsl:text>
```

```
  <xsl:value-of select='../@cislo' /><xsl:text>/</xsl:text>
```

```
  <xsl:choose>
```

```
    <xsl:when test="@cislo">
```

```
      <xsl:value-of select='@cislo' />
```

```
    </xsl:when>
```



```

    <xsl:otherwise>
      <xsl:text>bez čísla</xsl:text>
    </xsl:otherwise>
  </xsl:choose>
  <xsl:text>&#10;</xsl:text>
</xsl:template>

```

```

<!-- Ignoruj atributy a text -->
<xsl:template match='@*|text()' />

```

```

</xsl:stylesheet>

```

Podobný nástroj byl vytvořen pro nalezení místa, kde se vyskytuje konkrétní zeměřelý či konkrétní umělec. Vzhledem k tomu, že vstupních souborů bylo 22, ukázal se tento nástroj jako velmi užitečný.

XSLT našlo uplatnění i při fakturaci. Bylo nutno zjistit počet snímků, které byly nově fotografovány (byly v adresářích „1“ a „2“), počet fotografií z autorova archívu, které byly pouze skenovány (byly v adresáři „3“) a počet plánek. Tyto údaje zjistil následující styl:

```

<?xml version="1.0" encoding="cp852"?>

```

```

<!--

```

```

Tento styl zjišťuje počet fotek v jednotlivých adresářích
a~počet plánek
-->

```

```

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="1.0"
  xmlns:saxon="http://icl.com/saxon"
  extension-element-prefixes="saxon">

```

```

  <xsl:output indent="no" method="text" encoding="cp852"
    saxon:character-representation="native"/>

```

```

  <xsl:strip-space elements='*' />

```

```

  <!-- Celý výpočet dělá tato šablona -->

```

```

  <xsl:template match='/'>
    <xsl:text>Adresář 1:    </xsl:text>
    <xsl:value-of select=
      'count(//img[substring-before(normalize-space(.),"/")="1"])' />
    <xsl:text>&#10;</xsl:text>
  </xsl:template>

```

```

<xsl:text>Adresář 2:    </xsl:text>
  <xsl:value-of select=
    'count(//img[substring-before(normalize-space(.),"/")="2"])'>
<xsl:text>&#10;</xsl:text>
<xsl:text>Adresář 3:    </xsl:text>
  <xsl:value-of select=
    'count(//img[substring-before(normalize-space(.),"/")="3"])'>
<xsl:text>&#10;</xsl:text>
<xsl:text>Fotky celkem: </xsl:text>
  <xsl:value-of select='count(//img)'>
<xsl:text>&#10;</xsl:text>
<xsl:text>Počet plánek: </xsl:text>
  <xsl:value-of select='count(//oddeleni[@planek])'>
<xsl:text>&#10;</xsl:text>
</xsl:template>

<!-- Ignoruj všechno ostatní -->
<xsl:template match='*|@*|text()'>

</xsl:stylesheet>

```

7.4. Tvorba rejstříků

Kniha obsahuje jmenný rejstřík pohřbených osob a rejstřík tvůrců funerálního umění. Ve jmenném rejstříku se uvádí jméno člověka, jeho profesní charakterizace, datum narození a smrti a číslo stránky. Je to více údajů, než se v jiných knihách zpracovává programem MakeIndex, ale v zásadě by tento problém byl standardními L^AT_EXovými prostředky řešitelný. Náročnější je rejstřík tvůrců funerálního umění. Ten obsahuje příjmení a jméno umělce, jeho datum narození a případně úmrtí a číslo stránky. Potíž je v tom, že v textu se nevyskytují vždy všechny údaje. Písařka jména umělců nezná a není schopna všechny údaje do elementů <umelec> doplňovat. Navíc by tím stoupl počet neprázdných elementů <umelec> v dokumentu a validace by pak byla obtížnější. Při sazbě knihy *Olšanské umění, jeho tvůrci a doba* [9] to bylo řešeno pomocným perlowským skriptem a následným tříděním programem CSR [7]. Nyní bylo doplňování údajů řešeno výhradně v XSLT. Navíc bylo třídění podle normy implementováno do procesoru Saxon [10]. Uvádíme jen základní šablonu, která provádí třídění. Šablony pro standardizovaný tisk některých částí uvedeny nejsou.

```

<!-- Šablona pro výpis rejstříku umělců -->
<xsl:template name='umelci'>
  <xsl:variable name='jmeno' saxon:assignable='yes'>
  <xsl:variable name='krestni' saxon:assignable='yes'>

```

```

<xsl:variable name='narozen' saxon:assignable='yes'/>
<xsl:variable name='smrt' saxon:assignable='yes'/>
<xsl:variable name='odkaz' saxon:assignable='yes'/>
<xsl:variable name='novy' saxon:assignable='yes'/>
<xsl:variable name='fn'>
  <xsl:value-of select='$dir'/>
  <xsl:text>umelci.tex</xsl:text>
</xsl:variable>
<xsl:text>\input{</xsl:text>
<xsl:value-of select='$fn'/>
<xsl:text>}&#10;</xsl:text>
<xsl:document indent="no" method="text"
  encoding="cp852" href="{ $fn }"
  saxon:character-representation="native">
  <xsl:text>\typeout{Tvůrci funerálního umění</xsl:text>
  <xsl:text> File: </xsl:text>
  <xsl:value-of select='saxon:systemId()'>
  <xsl:text> line </xsl:text>
  <xsl:value-of select='saxon:lineNumber()'>
  <xsl:text>}&#10;</xsl:text>
  <xsl:text>\begin{umelci}&#10;</xsl:text>
  <!-- Setřídění všech umělců -->
  <xsl:for-each select='//umelec'>
    <xsl:sort select='@jmeno' lang='cs'/>
    <xsl:sort select='@krestni' lang='cs'/>
    <!-- Zjišťujeme, zda je to nový umělec,
      data jsou dána jen někdy -->
    <xsl:if test="$jmeno!='' ">
      <xsl:if test='@jmeno!=$jmeno'>
        <saxon:assign name='novy' select='1'>
      </xsl:if>
      <xsl:if test='$krestni'>
        <xsl:if test='$krestni!=@krestni'>
          <saxon:assign name='novy' select='1'>
        </xsl:if>
      </xsl:if>
      <xsl:if test='$narozen'>
        <xsl:if test='$narozen!=@narozen'>
          <saxon:assign name='novy' select='1'>
        </xsl:if>
      </xsl:if>
      <xsl:if test='$smrt'>

```

```

    <xsl:if test='$smrt!=@smrt'>
      <saxon:assign name='novy' select='1' />
    </xsl:if>
  </xsl:if>
  <xsl:if test='$novy'>
    <!-- Vypsání jména a vymazání proměnných -->
    <xsl:call-template name='umelec-v-rejstriku'>
      <xsl:with-param name='jmeno' select='$jmeno' />
      <xsl:with-param name='krestni' select='$krestni' />
      <xsl:with-param name='narozen' select='$narozen' />
      <xsl:with-param name='smrt' select='$smrt' />
      <xsl:with-param name='odkaz' select='$odkaz' />
    </xsl:call-template>
    <saxon:assign name='krestni' />
    <saxon:assign name='narozen' />
    <saxon:assign name='smrt' />
    <saxon:assign name='odkaz' />
    <saxon:assign name='novy' />
  </xsl:if>
</xsl:if>
<!-- Definice hodnot proměnných -->
<saxon:assign name='jmeno'>
  <xsl:value-of select='@jmeno' />
</saxon:assign>
<xsl:if test='@krestni'>
  <saxon:assign name='krestni'>
    <xsl:value-of select='@krestni' />
  </saxon:assign>
</xsl:if>
<xsl:if test='@narozen'>
  <saxon:assign name='narozen'>
    <xsl:value-of select='@narozen' />
  </saxon:assign>
</xsl:if>
<xsl:if test='@smrt'>
  <saxon:assign name='smrt'>
    <xsl:value-of select='@smrt' />
  </saxon:assign>
</xsl:if>
<!-- Vytvoření odkazu: id -->
<saxon:assign name='odkaz'>
  <xsl:value-of select='$odkaz' /><xsl:text> </xsl:text>

```

```

        <xsl:value-of select='generate-id()' />
    </saxon:assign>
</xsl:for-each>
<!-- Konec cyklu, takže musíme vypsát posledního umělce -->
<xsl:if test="$jmeno!=''">
    <xsl:call-template name='umelec-v-rejstříku'>
        <xsl:with-param name='jmeno' select='$jmeno' />
        <xsl:with-param name='krestni' select='$krestni' />
        <xsl:with-param name='narozen' select='$narozen' />
        <xsl:with-param name='smrt' select='$smrt' />
        <xsl:with-param name='odkaz' select='$odkaz' />
    </xsl:call-template>
    <saxon:assign name='krestni' />
    <saxon:assign name='narozen' />
    <saxon:assign name='smrt' />
    <saxon:assign name='odkaz' />
</xsl:if>
    <xsl:text>\end{umelci}&#10;</xsl:text>
</xsl:document>
</xsl:template>

<!-- Výpis umělce v rejstříku -->
<xsl:template name='umelec-v-rejstříku'>
    <xsl:param name='jmeno' />
    <xsl:param name='krestni' />
    <xsl:param name='narozen' />
    <xsl:param name='smrt' />
    <xsl:param name='odkaz' />
    <!-- Proměnná pro tisk iniciál -->
    <xsl:variable name='init'>
        <xsl:value-of select='ibs:cs-initial($jmeno)' />
    </xsl:variable>
    <xsl:if test='$init!=$initial'>
        <xsl:text>\uminic </xsl:text>
        <xsl:value-of select='$init' />
        <xsl:text>&#10;</xsl:text>
        <saxon:assign name='initial'>
            <xsl:value-of select='$init' />
        </saxon:assign>
    </xsl:if>
    <!-- Tisk jména a dat -->
    <xsl:text>\umelec{</xsl:text>

```

```

<xsl:call-template name='str-rep'>
  <xsl:with-param name='string'>
    <xsl:value-of select='$jmeno' />
    <xsl:if test="$krestni!=''">
      <xsl:text> </xsl:text><xsl:value-of select='$krestni' />
    </xsl:if>
    <xsl:call-template name='roky'>
      <xsl:with-param name='narozen'>
        <xsl:value-of select='$narozen' />
      </xsl:with-param>
      <xsl:with-param name='smrt'>
        <xsl:value-of select='$smrt' />
      </xsl:with-param>
    </xsl:call-template>
  </xsl:with-param>
</xsl:call-template>
<xsl:text>}\umpage </xsl:text>
<!-- Rozeber odkazy, oddělovač je definován
      v LaTeXovém stylu -->
<xsl:for-each select='saxon:tokenize($odkaz)''>
  <!-- Třídění není použité! -->
  <xsl:text>{</xsl:text>
  <xsl:value-of select='.' />
  <xsl:text>}</xsl:text>
</xsl:for-each>
<!-- Konec řádku -->
<xsl:text>\umpage&#10;</xsl:text>
</xsl:template>

```

Zbývá ještě vyřešit odkaz na správnou stránku. Používáme-li MakeIndex, máme v neseřazeném souboru čísla stránek uvedena. Pomocí XSLT však rejstřík vytváříme ještě před zahájením sazby a čísla stránek tudíž neznáme. Vytvoříme tedy jednoznačný identifikátor funkcí `generate-id{}`. Při sazbě textu jej použijeme jako jméno návěští v makru `\label`, v rejstříku jej použijeme v makru `\ref`. L^AT_EXovou část rejstříku tvůrců funerálního umění pak vysázíme těmito makry:

```
% Rejstřík umělců
```

```

\newenvironment{umelci}{\clearpage \let\Mezera\space
  \begin{multicols}{2}[\NadpisOddeleni
    {Tvůrci funerálního umění}]}%
  {\end{multicols}}

```

```

\def\uminic #1 {\par\vb\textbf{#1}\par\nobreak}

\def\umelec #1{\par
  \umpagetoks{}\def\delim{~\dots~\ignorespaces}\noindent
  \hangafter 1 \hangindent\parindent
  \rightskip 0mm plus 2em \relax #1}

\def\umprint #1.{\let\next\umprint
  \ifx \relax #1\let\next\par
  \else
    \delim #1\def\delim{, \ignorespaces}\fi \next}

\newtoks\umpagetoks

\def\umpage #1{\let\next\umpage
  \ifx \umpage #1\def\next{\expandafter
    \umprint\the\umpagetoks\relax.}\else
    \expandafter\ifx\curname r@#1\endcsname\relax
      \@latex@warning{Reference ‘#1’ on page \thepage
        \space undefined}\else
        \expandafter\expandafter\expandafter \edef
        \expandafter\expandafter\expandafter \test
        \expandafter\expandafter\expandafter
        {\expandafter\expandafter\expandafter
          \@secondoftwo\curname r@#1\endcsname}%
        \ifcat$\the\umpagetoks$\umpagetoks\expandafter{\test.}\else
          \def\next{\expandafter \expandafter \expandafter
            \umsortstart \expandafter \test
            \expandafter .\the\umpagetoks \relax.}\fi \fi \fi
  \next}

\def\umsortstart{\umpagetoks{}\umsort}
\def\umsort #1.#2.{\ifx \relax #2%
  \umpagetoks\expandafter
  {\the\umpagetoks #1.}\let\next\umpage\else
  \ifnum #1>#2
    \umpagetoks\expandafter{\the\umpagetoks #2.}%
    \def\next{\umsort #1.}\else
    \let\next\umsortend
  \ifnum #1=#2
    \umpagetoks\expandafter{\the\umpagetoks #2.}\else

```

```

\umpagetoks\expandafter{\the\umpagetoks #1.#2.}%
\fi
\fi
\fi \next}

\def\umsortend #1\relax.{\umpagetoks
\expandafter{\the\umpagetoks #1}\umpage}

```

V původní verzi procesoru Saxon bylo zajištěno, že abecedně seřazené identifikátory získané funkcí `generate-id()` odpovídaly jejich pořadí v dokumentu. Později však z důvodu zvýšení rychlosti zpracování od toho bylo upuštěno [2]. V rejstříku ovšem chceme čísla stránek setříděná. Může se též stát, že je některý umělec na téže stránce uveden dvakrát. V rejstříku však nechceme duplikovaná čísla. Dosahujeme toho cviky s token registrem `\umpagetoks`. Analýzu příslušných maker ponecháváme čtenářům za domácí cvičení.

8. Závěr

Při sazbě knihy, použité v případové studii, se prokázalo, že XML může významně pomoci i tehdy, požadujeme-li pouze jediný výstupní formát. Navíc došlo k nasazení XML v čistě komerční oblasti, kde za špatné rozhodnutí platíme skutečnými penězi. Zde však XML pomohlo k časové, a tím i finanční úspoře. Současně byl vyvrácen mýtus, že běžná sekretářka není schopna psát dokumenty v L^AT_EXu, natož v XML. Nelze pochopitelně předpokládat, že si sekretářka naprogramuje L^AT_EXový makrobalík, transformační styl pro XSLT či formátovací objekty, ale pokud jí někdo potřebné nástroje dodá, pak se je naučí používat stejně, jako se naučila naklikat typ a velikost písma, zarovnání odstavce a jiné volby v nejrůznějších dialogových oknech kancelářského textového editoru. V tomto případě stačila patnáctiminutová instruktáž o základech XML a když si odmyslíme běžné drobné překlepy, napsala písarka celý text bez chyb.

9. Poděkování

Chtěl bych poděkovat Jiřímu Koskovi za přehledně psanou knihu *XML pro každého*, bez jejíhož přečtení bych asi podrobnější knihy o XML nepochopil, i za jeho cenné rady, které mi při práci na knize *Olšanské umění, jeho tvůrci a doba*, jež byla mým prvním větším projektem v XML, ochotně poskytoval. Dále děkuji Michaele Knapové, která, ač není odbornicí v informačních technologiích a nemá žádný specializovaný editor pro práci s XML, přepsala celý text z rukopisu v podstatě bezchybně.

Reference

- [1] **DocBook web page**, <http://www.oasis-open.org/docbook/>
- [2] Kay M. H.: Soukromé sdělení.
- [3] Kay M. H.: **XSLT procesor Saxon**,
<http://users.iclway.co.uk/mhkay/>
- [4] Kosek J.: **XML pro každého**. Grada Publishing, 2000.
<http://www.kosek.cz/xml/>
- [5] Murray-Rust P., Rzepa H., Jiráť J., Nic M., Wagner Z.: **Chemical Markup Language**, <http://sourceforge.net/projects/cml/>
- [6] Neckářová L., Vanoušek A., Wagner J.: **Vinohradský hřbitov včera & dnes**. Správa pražských hřbitovů, Praha 2002. (Publikace je určena pro vnitřní potřebu členů Klubu Za starou Prahu a spolku Svatobor.)
- [7] Olšák P.: **Program csr (Czech Sort) – abecední řazení podle normy**. Zpravodaj Československého sdružení uživatelů T_EXu, 4 (3), 126–139 (1994).
<http://math.feld.cvut.cz/olsak/>
- [8] Rahtz S.: **PassiveT_EX**, <http://users.ox.ac.uk/~rahtz/passivetex/>
- [9] Vanoušek A.: **Olšanské umění, jeho tvůrci a doba**. Správa pražských hřbitovů, Praha 2000. (Publikace je určena pro vnitřní potřebu členů Klubu Za starou Prahu a spolku Svatobor.)
- [10] Wagner Z.: **Nástroje pro práci s XML/XSLT**,
<http://www.icebearsoft.cz/icebearsoft.euweb.cz/xml/>

Summary: Exploitation of XML and L^AT_EX in Scientific Book Publishing

Scientific books usually have rigorous structure than texts in humanities. This structure conveys information. Even though L^AT_EX enforces some kind of structured markup, nontrivial macroprogramming is needed in bigger applications. Some problems are caused during creation of indices with MakeIndex. Usage of XML strictly enforces document structure; one can take advantage of many existing tools for XML processing. Using L^AT_EX for the typesetting of XML transformed by XSLT is a natural choice. The approach is described on a case study that illustrates the whole publishing process.

Zdeněk Wagner

Ice Bear Soft

Email: wagner@cesnet.cz

Web: [icebearsoft.euweb.cz](http://www.icebearsoft.euweb.cz)

V tomto krátkom príspevku by som sa chcel s čitateľmi podeliť so svojimi skúsenosťami s použitím štýlu `pdfscreen.sty`. V Zpravodaji 1–2/97 (teda už pred takmer šiestimi rokmi!) sme si mohli prečítať zaujímavý príspevok [1] Petra Olšáka: „Jak sem dělal knihu klikací“. Musím sa priznať, že by som vtedy rád dokázal „stvorit“ nejaký pekný dokument vo formáte `pdf`, avšak nebol som do toho ničím tlačенý a navyše sa mi to zdalo príliš zložitú. Zhodou okolností je v tom istom čísle uverejnený aj príspevok „Hierarchie $\text{T}_{\text{E}}\text{X}$ istů“ [2]. V súčasnosti sa môžem podľa tejto škály zaradiť niekde medzi `Advanced` a `T_{\text{E}}\text{X}nician`. Tomu budú zodpovedať aj mnou nižšie popísané makrá a prosím $\text{T}_{\text{E}}\text{X}$ istov vyšších kategórií, aby sa nad ich kvalitou povzniesli. Mojim cieľom je tiež ukázať, že sa treba odvážne púšťať do riešenia $\text{T}_{\text{E}}\text{X}$ nických problémov a je možné dosiahnuť „slušné“ výsledky aj na začiatkovej etape chápania krás a možností $\text{T}_{\text{E}}\text{X}$ u.

1. Trochu histórie

Ako šiel čas, tak som sa viacmenej náhodne stretával s rôznymi ukážkami `pdf`-dokumentov. Jednou z nich bolo CD „Matematická analýza s programom Maple“ od autorov Z. Došlej, R. Plcha a P. Sojku, vydané Masarykovou univerzitou v Brne v roku 1999 [3]. Použitý formát dokumentu, využívajúci celú obrazovku bez potreby neustáleho potahovania lišty myšou, sa mi hneď zapáčil. Netušil som však, akým záhadným spôsobom sa dá niečo také urobiť (a nezapadlo ma otvoriť knižočku k CD, kde to je samozrejme napísané). Neskôr, asi začiatkom roku 2001, som na <http://www.math.uakron.edu/~dpstory/e-calculus.html> naďabil na „e-Calculus by D. P. Story“ [4], ktorého Copyright © 1995–1998 mi naznačil, koľko „meškám“. Znova som sa presvedčil, že na pohodlné čítanie textu z obrazovky je potrebné prispôbiť layout dokumentu tvaru obrazovky, zvoliť vhodné (dostatočne veľké) písmo a umožniť pohodlné stránkovanie (teda napr. pomocou klávesov `PageUp` a `PageDown`, resp. kliknutím myšou). Neskôr som sa ešte dostal ku dokumentu, ktorý D. P. Story nazval „Using $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ to Create Quality PDF Documents for the World Wide Web“ [5]. Rozhodol som sa tento text preložiť do slovenčiny, profesor D. P. Story mi láskavo na požiadanie poslal jeho zdrojový súbor. Keď som mal už takmer celý preklad hotový, stroskotal som na nejakej otázke, týkajúcej sa pozadia textu, čo mi nefungovalo presne podľa návodu, pretože nepoužívam `Y & Y` inštaláciu $\text{T}_{\text{E}}\text{X}$ u (súviselo to s použitím `PostScriptu`). To bol už koniec roka 2001. Vtedy som „dočasne“ od-

Obrázek 1: Ukážka obrazovky

ložil preklad bokom a stále sa nachádza v nezmenenom nedokončenom stave. Používal som však `web.sty` pri tvorbe učebníc.

V priebehu roka 2002 som sa nakoniec (ani si nepamätám za akých okolností) dostal ku štýlu `pdfscreen.sty`. Ukázalo sa, že výsledok je veľmi podobný na zobrazenie spomínaného brnenského CD [3] (prečo, to som už vysvetlil vyššie). Na obrázku 1 je znázornená typická výsledná obrazovka. Tento štýl môžete nájsť na CD `TEXlive 6` v adresári `drive:\texmf\tex\latex\pdfscreen`. Ak otvoríte pdf-súbor s názvom `drive:\texmf\doc\latex\pdfscreen>manual-screen.pdf`, môžete si prečítať podrobný návod [6], ktorý napísal C. V. Radhakrishnan, zrejme autor štýlu. Tam sa tiež dozviete, že sa vlastne jedná o rozšírenie možností balíka `hyperref.sty` od S. Rahtza. Priznám sa, že som tento návod nedočítal do konca, len som sa uspokojil s použitím samotného štýlu `pdfscreen.sty`. Tým chcem tiež naznačiť, že v návode nájdete mnohé ďalšie podrobnosti (napríklad ako pridávať tlačidlá), ktoré ja vôbec nespomínam.

2. Hlavičkové súbory

V tomto oddieli len uvediem použité štýly, ktoré som načítaval v preambule dokumentu.

```
01 \documentclass[pdftex,a4size,12pt]{article}
02 \usepackage{slovak}
03 \usepackage{amsfonts,amsmath,amssymb}
04 \usepackage[pdftex]{graphicx}
05 \usepackage{array}
06 \DeclareGraphicsExtensions{.pdf,.png,.jpg,.mps}
07 \usepackage{xspace,colortbl}
08 \usepackage[screen,panelright]{pdfscreen}
09      %panelleft,paneltoc,print
10 \begin{screen}
11     \margins{.65in}{.65in}{.65in}{.65in}
12     \screensize{6.25in}{8in}
13 \end{screen}
```

Komentár je zrejme zbytočný, len si všimnite, že zmenou voľby **screen** na **print** v riadku 08 získame namiesto súboru **pdf**, vhodného na prezeranie na obrazovke, súbor vhodný na vytlačenie.

3. Slovenské nastavenia

Nasledujúce slovenské nastavenie, použité tiež v preambule, vysvetľuje zároveň, aké tlačidlá uvidíme na obrazovke (pozri obrázok 1). Riadky 15–22 odpovedajú tlačidlám v uvedenom poradí, s výnimkou šípiek, označujúcich prechod na koniec dokumentu, resp. na nasledujúcu stranu. Tieto tlačidlá neobsahujú text.

```
14 %%%%%%%%%% Slovenské nastavenia %%%%
15 \panelhomepagename{Domovská stránka}
16 \paneltitlepagename{Titulná strana}
17 \panelcontentsname{Obsah}
18 \panelpagename{Strana}
19 \panelgobackname{Späť}
20 \panelfullscreenname{Celá strana}
21 \panelclosename{Zatvoriť}
22 \panelquitname{Koniec}
23 %%%%%%%%%%
```

Rozdielny je účinok stlačenia posledných dvoch uvedených tlačidiel. Stlačenie tlačidla v riadku 21 len uzavrie **pdf**-súbor, stlačenie tlačidla **Koniec** spôsobí uzavretie Acrobat Readera.

4. Nastavenie farieb

Nasledujúce riadky ukazujú, akým spôsobom je možné zadefinovať nové farby. Treba povedať, že niektoré z uvedených sa mi zapáčili na prvý pohľad. Farbu na riadku 32 (light blue background) som namiešal ja, ostatné sú prevzaté.

```
24 \definecolor{beigebg}{rgb}{1,.9,.8}
25 \definecolor{olivebg}{rgb}{.8,.9,.8}
26 \definecolor{offwhitebg}{rgb}{1,1,.9}
27 \definecolor{marshbg}{rgb}{.9,.9,.8}
28 \definecolor{lavenderbg}{rgb}{1,.75,.75}
29 \definecolor{bluebg}{rgb}{.9,1,1}
30 \definecolor{mustardbg}{rgb}{1,.8,.6}
31 \definecolor{redbg}{rgb}{1,0.4,0.4}

32 \definecolor{lbluebg}{rgb}{.6,.85,1}

33 \definecolor{redtxt}{rgb}{1,0,0}
34 \definecolor{greentxt}{rgb}{0,.5,0}
35 \definecolor{bluetxt}{rgb}{0,0,.6}
```

5. Definícia užitočných makier

Prichádzame ku časti, kde sa ukáže moja odvaha, keďže uvediem mnou definované makrá. Treba pripomenúť, že klikania na vzorce, literatúru a pod. už sú automaticky zahrnuté v popisovaných balíkoch, takže už na tejto úrovni je výsledok veľmi elegantný, umožňujúci príjemný pohyb po stránkach.

5.1. Definície, príklady, vety, ...

Nasledujú definície makier, neuvádzam všetky, ktoré vytvárajú očíslované objekty typu definícia, príklad a podobne. Text je zobrazovaný na farebnom pozadí, ktoré (podľa mňa) uľahčuje orientáciu v texte. Pri tlačovej verzii treba makrá samozrejme vhodne predefinovať, pri čítaní knižky by farby pôsobili rušivo.

```
36 \usepackage{mathpple}
37 \def\mbf#1{\textcolor{bluetxt}{\textbf{#1}}}
38 \def\mbbf#1{\textcolor{bluetxt}{\mbox{\boldmath$#1$}}}
39 \def\mbbfs#1{\textcolor{bluetxt}{\%
40 \mbox{\scriptsize\boldmath$#1$}}}
41 \def\Md{\mathrm{d}} % označenie diferenciálu
42 \def\theequation{\arabic{section}.\arabic{equation}}
43 \def\thefigure{\arabic{section}.\arabic{figure}}
```

V riadku 36 inštalujem písmo Palatino, ktoré sa mi zdá elegantné. Riadky 37–40 sú definície farebných polotučných písem.

```
44 \setlength\fbboxsep{0.01\textwidth}
45 \newlength\odstavec \setlength\odstavec{1\parindent}
46 \def\odsek{\hspace*{\odstavec}}
47 \newbox\kniha
48 \newcounter{definicia}\setcounter{definicia}{0}
49 \def\definicia{\addtocounter{definicia}{1}%
50   \setbox\kniha\hbox\bgroup%
51   \begin{minipage}{0.98\textwidth}%
52   \mbf{Definícia~%
53   \arabic{section}.\arabic{definicia}.}\quad}
54 \def\edefinicia{\end{minipage}\egroup\smallskip\noindent%
55   \colorbox{bluebg}{\usebox\kniha}\smallskip}
56 \def\bkniha{\setbox\kniha\hbox\bgroup%
57   \begin{minipage}{0.98\textwidth}} %22pc
58 \def\ekniha{\end{minipage}\egroup\smallskip\noindent%
59   \colorbox{olivebg}{\usebox\kniha}\bigskip}
60 \newcounter{priklad}\setcounter{priklad}{0}
61 \def\priklad{\addtocounter{priklad}{1}%
62   \setbox\kniha\hbox\bgroup%
63   \begin{minipage}{0.98\textwidth}\mbf{Príklad~%
64   \arabic{section}.\arabic{priklad}.}\quad}
65 \def\epriklad{\end{minipage}\egroup\smallskip\noindent%
66   \colorbox{lavenderbg}{\usebox\kniha}\smallskip}
67 \newcounter{veta}\setcounter{veta}{0}
68 \def\veta#1{\addtocounter{veta}{1}%
69   \setbox\kniha\hbox\bgroup%
70   \begin{minipage}{0.98\textwidth}\mbf{Veta~%
71   \arabic{section}.\arabic{veta}. #1}\quad}
72 \def\eveta{\end{minipage}\egroup\smallskip\noindent%
73   \colorbox{beigebg}{\usebox\kniha}\smallskip}
```

Vzhľadom na to, že zafarbiť príkazom `\colorbox` môžeme len boxy, texty definícií, príkladov, viet a pod. umiestňujeme do boxu prostredia `minipage`. Metódou pokusov som nastavil dĺžku v riadku 44, s ňou súvisí neskoršie použitie šírky `minipage`, pozri napr. riadok 51. V prostredí `minipage` je parameter `parindent` nastavený na nulu, preto ak chceme použiť odstavec odsadený tak ako v základnom texte, použijeme príkaz `\odsek` definovaný v riadkoch 45–46.

Na zapamätanie si textu definícií a pod. som použil uloženie obsahu boxu, vytvoreného prostredím `minipage`, do boxu `\kniha`, definovaného v riadku 47.

Jednotlivé definície, príklady, vety a pod. vytvárame v dokumente príkazmi typu:

```
74 \definicia Funkcia $f(x_1,x_2,\dots,x_n)$ \dots
75 \edefinicia
```

alebo

```
76 \veta{ (\mbf{Frobeniova})} Sústava lineárnych rovníc
77 \dots \eveta
```

Uspokojil som sa s uvedenými definíciami, ktoré síce automaticky očísľujú uvažované objekty, ale nevieme sa na nich odkazovať dvojicou príkazov `\label` a `\ref`. Tuším, že pri poriadnom štúdiu literatúry by sa problém dal zvládnuť, nebolo to však až také akútne, hoci priznávam, že by mi to možno ušetrilo čas, ktorý som v niektorých prípadoch musel spotrebovať na „ručné“ vytvorenie počítadiel na potrebné odkazy. Navyše som v skutočnosti pri dlhom zadaní príkladu použil ešte príkaz na pokračovanie príkladu.

5.2. Prepojenie úloh a ich riešení

Na záver uvediem definíciu makier, ktoré automaticky prepoja úlohy s ich riešeniami.

```
78 \newcounter{uloha}\setcounter{uloha}{0}
79 \def\uloha#1{\addtocounter{uloha}{1}%
80   \setbox\kniha\hbox\bgroup%
81   \begin{minipage}{0.98\textwidth}%
82   \hypertarget{U#1}{\mbf{Úloha~%
83     \arabic{section}.\arabic{uloha}.}}\quad}
84 \def\euloha#1{\hfill\hyperlink{RU#1}{${\bullet}$}%
85   \end{minipage}\egroup\smallskip\noindent%
86   \colorbox{mustardbg}{\usebox\kniha}\smallskip}
87 \newcounter{ruloha}\setcounter{ruloha}{0}
88 \def\ruloha#1{\addtocounter{ruloha}{1}%
89   \setbox\kniha\hbox\bgroup%
90   \begin{minipage}{0.98\textwidth}\footnotesize%
91   \hypertarget{RU#1}{\mbf{%
92     \arabic{section}.\arabic{ruloha}.}}\quad}
93 \def\eruloha#1{\hfill\hyperlink{U#1}{${\bullet}$}%
94   \end{minipage}\egroup\smallskip\noindent%
95   \colorbox{mustardbg}{\usebox\kniha}\smallskip}
```

Zamyslite sa nad uvedenými definíciami. Ich použitie je nasledovné. V základnom súbore zadáme ďalšiu úlohu príkazmi v riadkoch 96–97, obsahujúcimi parameter – prezývku danej úlohy.

```
96 \uloha{LPA} Určte \dots
97 \euloha{LPA}
```

Na inom mieste, napríklad v samostatnom súbore riešení potom použijeme príkazy, uvedené v riadkoch 98–99, s tými istými prezývkami!

```
98 \ruloha{LPA} Lahko sa presvedčíme \dots
99 \eruloha{LPA}
```

Získané prepojenie bude fungovať v prípade, ak ku každej úlohe napíšeme aj odpovedajúcu odpoveď. Riešenia úloh musia byť usporiadané v rovnakom poradí ako samotné úlohy.

6. Záver

Uvedený postup len naznačuje možnosti, poskytované uvedeným štýlom `pdfscreen.sty`. Samotné písanie a $\text{\TeX}$ ovanie boli veľmi príjemné. Pri kompilovaní `pdf\TeX`om som použil formát `pdfcslatex`. Je možné, že makrá nie sú dobre napísané, pretože som mal problémy pri kompilácii. Z neznámych príčin mal `pdf\LaTeX` problémy s prezývkami, nepodarilo sa mi skompilovať veľký súbor (to môže súvisieť s verziou `pdf\TeXu`), rozdelením učebnej pomôcky som však získal o publikáciu viac :-)

Reference

- [1] Olšák, P.: *Jak sem dělal knihu klikací*, Zpravodaj 1–2/97, 1–17.
- [2] *Hierarchie $\text{\TeX}$ istů*, Zpravodaj 1–2/97, 30–33.
- [3] Došlá, Z. – Plch, R. – Sojka, P.: *Matematická analýza s programem Maple I: Diferenciální počet funkcí více proměnných*, Brno, Masarykova univerzita, 1999, 424 s., elektronická publikácia, <http://www.math.muni.cz/~plch/difer/difer.html>.
- [4] Story, D. P.: *e-Calculus*, Univerzita v Akronu, 1998, elektronická publikácia.
- [5] Story, D. P.: *Using $\text{\LaTeX}$ to Create Quality PDF Documents for the World Wide Web*, Univerzita v Akronu, 1998, 55 s., elektronická publikácia.
- [6] Radhakrishnan, C. V.: *Pdftscreen*, 2000, 18 s., elektronická publikácia.

Summary: Experiments with Macropackage `pdfscreen.sty`

An experience with `pdfscreen.sty` $\text{\LaTeX}$ style is discussed, and examples of usage are given.

Ján Buša
Jan.Busa@tuke.sk

Rejstřík

Acrobat, 143
Antoš, D., 219
Aubrecht, P., 219
automatizace testů, 123

Buša, J., 212

Csfonty, 172
CSL^AT_EX, 172
Csplain, 172

DSC, 141
DTD, 188

editorské práce, 152

fonty, 172
formuláře, 123

Hàn Thê Thành, 144
Hrad, M., 123
Hála, T., 152

korektury, 152

L^AT_EX, 172, 188

Maple, 212

NFSS, 172

OFS, 172

Olšák, P., 171

PDF, 140, 212
pdfscreen.sty, 212
pdfT_EX, 140
pdftricks, 141
prezentace, 140

redakční práce, 152
RIP, 141
rozpoznávání, 123
Rybička, J., 160

SCAT, 123
schéma, 188
skenování, 123
Sojka, P., 123

validace, 188

Wagner, Z., 188

XML, 188
xpdf, 144
XSLT, 188

Zýka, V., 140

Škarvada, L., 219

čárové kódy, 123

Hlavní část tohoto dvojčísla Zpravodaje, jehož jsem hostujícím editorem, vznikla přesazbou mírně upravených vybraných článků prvně publikovaných ve sborníku třetího ročníku semináře o Linuxu a $\text{\TeX}$ u $\text{\SLT}$ 2002 pořádaného 7.–10. listopadu 2002 v Seči. Oproti sborníku $\text{\SLT}$ byly do dvojčísla zařazeny aktuální informace $\text{\CS}$ TUGu a článek o balíku `pdfscreen.sty` z pera kolegy Buši.

Sazba byla provedena písmem Computer Modern místo v původní publikaci použitém Charteru. Byl převzat design Zpravodaje a sloučeny tematický a autorský rejstřík. Závěrečný překlad byl proveden `pdf $\text{\TeX}$` em verze 3.14159-1.00b-pretest-20020211 (Web2C 7.3.7x), s využitím OFS. Dále byly využity programy `dvips(k)` 5.90a, `ghostscript` 8.00, `CSindex`, `pstops`, `make` a další software s veřejným zdrojovým kódem.

Závěrem mi nezbyvá než poděkovat Davidovi Antošovi za pomoc při sazbě tohoto dvojčísla, Liborovi Škarvadovi a Petrovi Aubrechtovi za korekturu některých článků tohoto Zpravodaje a čtenářům za trpělivost při čekání na něj. I když se při přípravě čísla do tisku nepodařilo zrealizovat všechna doporučení z článku o korekturách v 21. století, doufám, že četba dvojčísla vám přinese potěšení nerušené množstvím chyb.

Zpravodaj Československého sdružení uživatelů $\text{T}_{\text{E}}\text{X}$

ISSN 1211-6661 (tištěná verze), ISSN 1213-8185 (online verze)

Vydalo: Československé sdružení uživatelů $\text{T}_{\text{E}}\text{X}$
vlastním nákladem jako interní publikaci

Obálka: Antonín Strejc
Na obálce použito logo SLT,
jehož autorkou je Petra Rychlá

Počet výtisků: 700

Uzávěrka: 15. prosince 2002

Odpovědný redaktor: Zdeněk Wagner

Redakční rada: Petr Aubrecht, Matěj Cepl, Jiří Demel,
Jana Chlebíková, Jiří Kosek, Jaromír Kuben,
Petr Sojka, Martin Tkadlčík

Tisk a distribuce: KONVOJ, spol. s r. o., Berkova 22, 612 00 Brno,
tel. +420 549 240 233

Adresa: $\mathcal{C}\mathcal{S}\text{TUG}$, c/o FEL ČVUT, Technická 2, 166 27 Praha 6

Tel: +420 224 353 611

Fax: +420 233 332 938

Email: cstug@cstug.cz

Zřízené poštovní aliasy sdružení $\mathcal{C}\mathcal{S}\text{TUG}$:

bulletin@cstug.cz, zpravodaj@cstug.cz

korespondence ohledně Zpravodaje sdružení

board@cstug.cz

korespondence členům výboru

cstug@cstug.cz, president@cstug.cz

korespondence předsedovi sdružení

gacstug@cstug.cz

grantová agentura $\mathcal{C}\mathcal{S}\text{TUGu}$

secretary@cstug.cz, orders@cstug.cz

korespondence administrativní síle sdružení, objednávky CD-ROM

cstug-members@cstug.cz

korespondence členům sdružení

cstug-faq@cstug.cz

řešené otázky s odpověďmi navrhované k zařazení do dokumentu $\mathcal{C}\mathcal{S}\text{FAQ}$

bookorders@cstug.cz

objednávky tištěné $\text{T}_{\text{E}}\text{X}$ ové literatury na dobírku

ftp server sdružení:

<ftp://ftp.cstug.cz/>

www server sdružení:

<http://www.cstug.cz/>

Uzávěrka příštího čísla: 31. března 2003