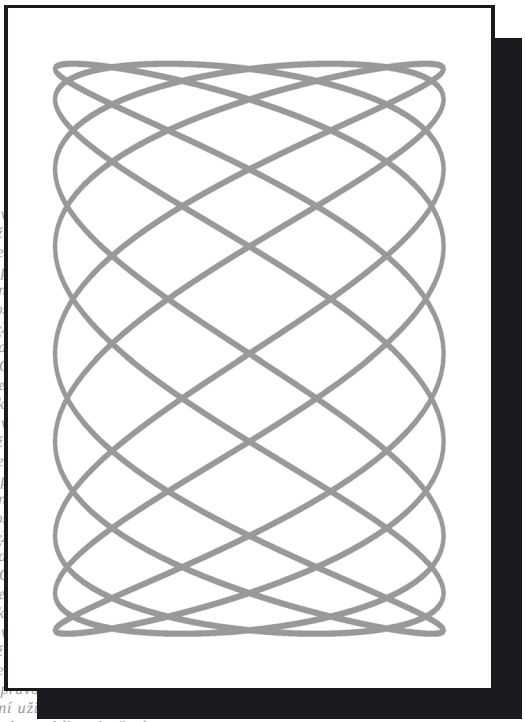
CSTUG



ZPRAVODAJ

Československého sdružení uživatelů TEXu

1

2002

ISSN 1211-6661 Tištěná verze

ISSN 1213-8185 Online verze

Ročník 12

OBSAH

Zpravodaj Československého sdružení uživatelů TeXu 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 anonymním ftp na `ftp.icpf.cas.cz` do adresáře `/wagner/incoming/`, nejlépe jako jeden archivní soubor (`.zip, .arj, .tar.gz`). Současně zašlete elektronickou poštou upozornění na `mailto:bulletin@cstug.cz`. Uvedený adresář je přístupný pouze pro zápis. 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 CSTeXu), zejména v případě, kdy vás nelze kontaktovat e-mailem.

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

Úvodníček

Petr Olšák

Na prosincovém valném shromáždění došlo z mého pohledu k nejhoršímu :-). Ačkoli jsem se snažil dát všemožně najevo, že spojení funkce předsedy s mou osobou není nejvhodnější nápad, a vymýšlel jsem útlumové programy, se kterými jsem předstupoval do voleb výboru, nakonec jsem byl do toho výboru přeci jen zvolen. I když s asi nejméně hlasy. A nově zvolený výbor si mě paradoxně zvolil předsedou.

Nicméně nedošlo k nejhoršímu z pohledu řadového člena, který nevolil nulové varianty a podobné útlumové programy. Nový výbor tuto cestu neschválil, takže i nadále by měl CSTUG fungovat zhruba tak, jak fungoval dosud. Členské příspěvky pro tento rok jsou stejné, jako byly v roce minulém a jejich výše byla uvedena ve Zpravodaji 4/2001 a je uvedena na našem webu `www.cstug.cz`.

Dilema, které z toho pro mne osobně plyne, jsem se rozhodl řešit přístupem, který by se dal shrnout do slova *nepřekážet*. Pokud chtějí členové nového výboru zachovat, nebo dokonce rozšířit, činnosti CSTUGu, mohou. Já jim v tom samozřejmě z titulu své funkce nebudu bránit a budu jim v tom pomáhat v rozsahu, jaký jsem schopen pokrýt vlastními silami. Nebudu ale sám nikoho do ničeho tlačit, přesvědčovat a dělat z vlastní iniciativy nějaké nové akce. Navíc veřejně prohlašuji, že jsem totální managerský antitalent. Přivítám, pokud ostatní členové výboru nějaké nové aktivity vymyslí a staré udrží aspoň na stejné úrovni. Ostatně zapojit se do aktivní činnosti mohou i ostatní členové CSTUGu, nejen členové výboru.

U příležitosti volby nového výboru a předsedy bych chtěl na tomto místě poděkovat odstupujícímu výboru a předsedovi za velký kus práce, který pro naše sdružení často nezištně udělali. Zvláště odstupující předseda Petr Sojka posunul fungování CSTUGu velmi výrazným krokem kupředu. Já se jen obávám, aby za mého působení nedošlo zase ke kroku zpět. Pokusím se ovšem udělat vše pro to, aby tomu tak nebylo. Zdaleka ne v poslední řadě chci poděkovat i těm lidem, kteří se rozhodli kandidovat do nového výboru a vyjádřili tím ochotu se aktivně podílet na činnosti CSTUGu i do budoucna. Toto poděkování platí jednak novým tvářím, ale též těm, kteří se rozhodli uplatnit své zkušenosti s prací ve starém výboru i nadále.

Mým původním záměrem, který se promítl do mého volebního programu, bylo převést veškerou agendu CSTUGu včetně sídla organizace k 1. 1. 2002 do Prahy. Měl jsem dojem, že to bude v mnohém ohledu operativnější. V listopadu, ještě před valným shromážděním jsem měl osobní rozhovor s panem Ondřejem Váchou, který nám už mnoho let dělá hospodáře sdružení a zprostředkovává

napojení na účetní. Dospěli jsme tehdy k závěru, že unáhlené stěhování všech věcí do Prahy, když vlastně není komu předat funkci hospodáře, by nebylo moc vhodné. Domluvili jsme se, že se pokusíme potřebné náležitosti převést do Prahy až v roce 2002 a postupně. Bohužel, mezi pražskými členy výboru se zatím nepodařilo najít člověka, který by funkci Ondřeje Váchy kompletně převzal. Nicméně Karel Horák nám zprostředkoval aspoň seznámení s paní Mgr. Holovskou z Matematického ústavu AV ČR, která se nabídla dělat některé práce sekretářky. Jaké práce by mohla převzít, je zatím v jednání. Je zřejmé, že asi nebude moci dělat všechno, oč se dosud staral Ondřej Vácha, takže prostor pro dalšího dobrovolníka je zřejmě stále otevřen.

Z předchozího odstavce plyne, proč jsme zatím na valném shromáždění nepožádali o změnu stanov ve věci sídla našeho sdružení. Adresa pro tento rok tedy zůstává stejná: CSTUG, Botanická 68a (FI MU), 602 00, Brno. Teprve na dalším zimním shromáždění se pokusíme změnit stanovy. Chtěl bych v této souvislosti oslovit všechny členy, aby se zamysleli nad textem stávajících stanov. Pokud někdo má pocit, že by bylo dobré něco formulovat lépe, může se ozvat na výborový list `board@cstug.cz`. Může se stát členem komise pro změnu stanov, která vypracuje návrh nového textu, jejž s předstihem předložíme všem členům sdružení k diskusi. Pak ke konci roku na valném shromáždění navrhneme konečnou verzi změny ke schválení.

Otvíráte první číslo našeho časopisu v roce 2002, který jsme se rozhodli vydat ještě stále stejnými cestami jako čísla minulá. Mám tím na mysli sazbu, tiskárnu, a distribuci. Pokud zrovna čtete tyto řádky, je to neklamným znamením, že staré cesty pořád ještě fungují. Otázka, zda v souvislosti s převedením činnosti CSTUGu do Prahy se mají tyto cesty měnit, je zatím otevřena a ve stadiu diskuse na výborovém e-mailovém listu.

Kolem pana Wagnera se seskupilo několik dobrovolníků, kteří vytvořili tzv. „redakční radu“, která bude pomáhat s vybíráním a korigováním článků do časopisu. Předpokládám, že jmenný seznam členů redakční rady najdete v tomto časopise na jiném místě. Je samozřejmé, že množství zájemců o práci v redakční radě není ještě zárukou toho, že časopis bude obsahovat vůbec nějaké články. Vyzýváme tedy hlavně autory článků, kteří jsou ochotni se podělit o své zkušenosti s TEXem s ostatními a napsat o tom do časopisu. Zde je prostor pro každého člena CSTUGu. Určitě nebude redakční rada příspěvek odmítat jen proto, že nepochází od dostatečně známého autora, nebo proto, že se jedná o nějaké „začátečnické“ téma. Právě naopak, podle ohlasů soudíme, že by plno čtenářů takové téma velmi přivítalo.

S přáním příjemně strávených chvil s časopisem i s TEXem předávám slovo autorům jednotlivých příspěvků.

Generování vzorů pomocí knihovny PatLib a programu OPatGen

David Antoš, Petr Sojka

Článek[1] popisuje techniku generování vzorů jako prostředek pro získávání informace z rozsáhlých dat. Typickou aplikací této techniky je vytvoření časově i prostorově velmi efektivního algoritmu dělení slov ze seznamu již rozdělených slov. Doposud chyběl generátor vzorů dělení pro UNICODE (pro systém Ω) a rozšíření dosud užívaného programu PatGen, omezeného osmibitovým ASCII, nebylo již nadále únosné. Proto vyvíjíme knihovnu PatLib pro obecnou manipulaci se vzory a na ní postavený generátor vzorů dělení slov OPatGen. Popíšeme architekturu tohoto systému. Vzory lze použít i pro rozpoznávání hranic složených slov, proto zmíníme návrhy na rozšíření následníků TEXu o klasifikované dělení s více typy dělicích bodů a o automatické potlačování ligatur na švech složených slov.

Klíčová slova: PatLib, OPatGen, soutěživé vzory, rozpoznávání vzorů, generování vzorů, dělení (složených) slov, PatGen, Ω.

Úvod

"Go forth and make masterpieces of hyphenation patterns . . . "
—Yannis Haralambous [5]

Téměř vše lze považovat za symbol a ze symbolů můžeme kombinovat *vzory*. Vzory jsou často zjevením a popisem vyšších zákonitostí. Hofstadter [9] považuje rozpoznávání vzorů za centrální pojem inteligence a v tomto smyslu jsou zaměřeny i mnohé inteligenční testy.

Technika vzorů je efektivním prostředkem pro extrakci informací a rozpoznávání v datech. Na vzorech je v TEXu [12] postaveno elegantní a na jazyku nezávislé řešení pro kvalitní dělení slov. Tento efektivní algoritmus pak našel cestu i do téměř všech následně vznikajících sázecích systémů včetně těch komerčních a dle našeho mínění si zaslouží ještě mnohem větší pozornost pro své široké možnosti použití.

Generování vzorů pro dělení pomocí programu PatGen [14] nedostačuje plně současným potřebám a pro jeho širší použití je třeba po dvaceti letech od

[1]Předběžná verze tohoto článku byla publikována v [3]. Výzkum byl prováděn v rámci výzkumného záměru CEZ:J07/98:143300003.

jeho vzniku provést mnohá zobecnění. Vznikl systém Ω [6] mj. s cílem umožnit přímou práci se slovy všech jazyků v UNICODE (universální vzory dělení, kontextové substituce, překladové procesy Ω – ΩTP). Tyto nové výzvy a v dalším textu naznačené analýzy nás dovedly k závěru, že nejlepším řešením bude úplná reimplementace PatGENu.

V článku popíšeme základní principy techniky vzorů, jakým způsobem jsou vzory generovány a jak je potom TEX používá. Dále se budeme věnovat architektuře systému OPatGen a knihovny PatLib a jejímu použití jako universální knihovny pro manipulaci se vzory. Závěrem zmíníme možné aplikace technologie založené na zpracování vzorů.

Vzory

> *Middle English* patron *'something serving as a model', from Old French. The change in sense is from the idea of a patron giving an example to be copied. Metathesis in the second syllable occured in the 16th century. By 1700* patron *ceased to be used on things, and the two forms became differentiated in sense.*
> — Origin of word pattern: [4]

Vzory slouží pro rozpoznávání „zajímavých míst" v datech. Zajímavým místem může být hranice znaků, kde je v daném slovu povoleno dělit, hranice voda/les na leteckém snímku krajiny a podobně. Zajímavé místo lze rozpoznat pomocí jeho kontextu. Vzory jsou podslova dané množiny slov, mezi jejichž symboly je vyznačena informace o zajímavých místech.

Informace o těchto bodech je v principu dvojího druhu: jedna říká, že dané místo *je* místem zájmu, druhá, že *není*. Typickou reprezentací bývají přirozená čísla, lichá pro ano, sudá pro ne. Tedy vzory máme *pokrývací* a *zabraňující*. Lze také použít další speciální symboly, jako například tečku značící začátek nebo konec slova. Například české dělení obsahuje vzory `i1h`, `i3hl.` a `i2hl.`. Použití vzorů je toto: pro všechna podslova daného slova se najdou všechny odpovídající vzory. Tedy slovu `cihla` odpovídají vzory `i1h`, `i2hl` z naší množiny, takže máme `ci2hla`. Vzory se při použití *přebíjejí*, můžeme říci, že *soutěží*, a výsledkem je maximum z hodnot odpovídajících pozici mezi znaky. Za chvíli si vysvětlíme, jak se vzory generují. To také lépe objasní, proč se používají zrovna takto.

Podrobný popis použití vzorů lze nalézt v [12, příloha H]. Laskavého čtenáře se zájmem o podrobnější a formálnější informace o vzorech odkazujeme na články [19, 10, 1], o nástroje na práci s konečnými automaty pak na [15, 11, 16, 13, 2].

Generování vzorů

"An important feature of a learning machine is that its teacher will often be very largely ignorant of quite what is going on inside, although he may still be able to some extent to predict his pupil's behaviour."
—Alan Turing, [21]

Požadujeme, aby vzory pokrývaly co nejvíce míst zájmu, tedy aby měly *maximální úplnost*. Zároveň chceme, aby chybovaly co možná nejméně (*maximální přesnost*) a přitom byly *prostorově minimální*, tedy tak malé, jak je to jen možné. Z těchto protichůdných požadavků je nejméně bolestný požadavek minimality, ze stoprocentní přesnosti však slevovat nechceme. Iterativní metodou lze dosáhnout překvapivě dobrých výsledků a komprimovat informace ze vstupních dat do množiny vzorů. Popišme, jak takové generování probíhá.

Potřebujeme rozsáhlou množinu vstupních dat, ve které jsou označena místa zájmu, v našem případě se bude jednat o slova přirozeného jazyka a v nich označená místa, kde je dovoleno dělit.

Nyní budeme opakovaně procházet soubor vstupních dat. Jednotlivé průchody nazvěme *úrovněmi*. V lichých úrovních generujeme pokrývací vzory, v sudých zabraňující.

V každé úrovni zvolíme *pravidlo*, pomocí něhož vybíráme *kandidáty* na vzory. V našem případě pravidlo může mít tvar „k-znakový podřetězec slova obsahující dělicí bod". Vybereme kandidáty na vzory a uložíme je do vhodné datové struktury. Ne každý kandidát je ovšem dobrý vzor, proto potřebujeme *pravidlo pro výběr vzorů*. Obvykle je rozumné projít všechny kandidáty na vzory a vyzkoušet jejich funkci (přitom se používají i vzory z předchozích úrovní, je o práci všech dosud vytvořených vzorů) na slovech ze vstupních dat. Přitom spočteme, kolikrát kandidát pracoval dobře a kolikrát chyboval. Pravidlem pak může být funkce nad těmito hodnotami porovnávaná se zadanou hodnotou, tedy klasické lineární prahování.

Kandidáty, které jsme v předchozím procesu označili za dobré, zařadíme mezi vzory. Tyto vzory ovšem mohou (a většinou také budou) stále chybovat. Takže budeme pokračovat další úrovní, tentokrát sudou, v níž budeme vytvářet zabraňující vzory. Dobrým vzorem určité úrovně je kandidát, který opravuje chyby vzorů z nižších úrovní. Podrobněji řečeno, v pokrývací (liché) úrovni je dobrý kandidát, který najde správný dělící bod. V zabraňující úrovni je dobrý ten, který opraví bod označený chybně předchozí množinou vzorů jako místo dělení.

Tímto způsobem v principu pracuje program PatGen, s poměrně pevně danými pravidly pro výběr vzorů (lineární prahování). Vzory dělení slov pro TeX existují pro několik desítek jazyků, většina generována PatGenem ze slovníku rozdělených slov. Musíme poznamenat, že se často také postupuje tak, že se vzory vytvářejí částečně ručně (buď pro bootstrapping, nebo ručně).

Jaká je úspěšnost této techniky? Ze slovníku velikosti několika MB lze vytvořit vzory velikosti řádově desítek KB pokrývající nad 98 % dělicích bodů a s chybovostí pod 0,1 %. Četné experimenty ukázaly, že se vystačí často se čtyřmi úrovněmi [20]. Pomocí vhodných technik (bootstrapping, stratifikace) a strategií nastavení parametrů pro lineární prahování bylo ukázáno [20, 17, 18], jak se dají generované vzory optimalizovat. Příklady statistik z generování variant českých vzorů dělení jsou v tabulkách 1, 2 a 3. Pro nastavování parametrů nemáme žádné teoretické podklady, generování vzorů je zatím do značné míry závislé na umění a zkušenostech.

Tabulka 1: Standardní generování českých vzorů s parametry Lianga

úroveň	délka	param	% dobré	% špatné	# vzorů	velikost
1	2–3	1 2 20	96,95	14,97	+ 855	
2	3–4	2 1 8	94,33	0,47	+1706	
3	4–5	1 4 7	98,28	0,56	+1033	
4	5–6	3 2 1	98,22	0,01	+2028	32 kB

Tabulka 2: Standardní generování českých vzorů s optimalizací na velikost vzorů

úroveň	délka	param	% dobré	% špatné	# vzorů	velikost
1	1–3	1 2 20	97,41	23,23	+ 605	
2	2–4	2 1 8	85,98	0,31	+ 904	
3	3–5	1 4 7	98,40	0,78	+1267	
4	4–6	3 2 1	98,26	0,01	+1665	23 kB

Tabulka 3: Standardní generování českých vzorů s optimalizací pokrytí dělicích bodů

úroveň	délka	param	% dobré	% špatné	# vzorů	velikost
1	1–3	1 5 1	95,43	6,84	+2261	
2	1–3	1 5 1	95,84	1,17	+1051	
3	2–5	1 3 1	99,69	1,24	+3255	
4	2–5	1 3 1	99,63	0,09	+1672	40 kB

Proč PatGen nestačí?

"The road to wisdom?
Well it's plain and simple to express:
Err and err and err again
but less and less and less."
—Piet Hein [8]

Program PatGen má několik vážných omezení. Je to monolit strukturovaného kódu, který, ačkoli je velmi dobře dokumentován (WEB, tj. dokumentovaný Pascal), není snadné upravovat. Obsahuje radikální optimalizace, které umožnily, že se proces generování vzorů dělení vešel do paměti PDP-10. Tyto optimalizace značně snižují srozumitelnost kódu, ztěžujíce tak možnosti úprav.

Datové struktury PatGenu jsou postaveny na osmibitový ASCII kód, přitom rozšíření na UNICODE prakticky nepřichází v úvahu. Maximální počet úrovní je devět. V průběhu výběru kandidátů na vzory lze současně vybírat pouze kandidáty shodných délek. Data jsou interně ukládána do statických struktur, pokud během generování paměť dojde, je nutno zasáhnout do zdrojového kódu a rekompilovat.

Samozřejmě lze PatGen využít pro generování vzorů pro jiné jevy než dělení slov, ovšem pouze tak, že se daný problém na dělení slov převede. To může být značně netriviální a navíc lze takto řešit pouze problémy s dostatečně malou abecedou, přibližně pod 240 symbolů. PatGen totiž některé ASCII znaky používá jako výstupní symboly.

PatLib a OPatGen

"My library was dukedom large enough."
—Shakespeare, The Tempest (1611), act 1, sc. 2 l. 109

Rozhodli jsme se tedy PatGen zobecnit. Implementovali jsme knihovnu PatLib (PATtern LIbrary) pro práci se vzory. Generátor vzorů dělení v UNICODE, který využívá služeb knihovny, jsme nazvali OPatGen.

Pro implementaci jsme zvolili méně obvyklou kombinaci v CWEBu psaného C++ z důvodů portability a efektivity a udržování kvalitní dokumentace. Navíc šablony v C++ umožňují odložit přesnou specifikaci typu na co možná nejpozději, což se během analýzy ukázalo jako velká výhoda.

Knihovna PatLib se skládá z manipulátoru s konečným jazykem a generátoru vzorů. Vrstva pro uložení konečných slov je implementována pomocí komprimované varianty datové struktury trie.

Manipulátor jako svou interní abecedu používá čísla pro zachování rozumné efektivity, je však schopen pracovat i s jinými objekty. Manipulátor se vzory je v principu konečným automatem s výstupem omezeným na konečný jazyk.

Poskytuje základní služby typu „vlož vzor“, „dej výstup vzoru“, „odstraň vzor“ a dále umí vydat po jednotlivých vzorech celý uložený jazyk. Výstupní informací vzoru může být také libovolný objekt.

Protože mezi nejčastěji zjišťované charakteristiky kandidátů na vzory patří dvojice čísel udávající, kolikrát kandidát pracuje správně a kolikrát způsobuje chybu, připravili jsme i službu zajišťující pohodlnou práci s těmito údaji.

Generátor implementuje výše popsanou (s PatGENem kompatibilní) strategii generování, tedy opakovaně prochází vstupní data a sbírá statistiky o kandidátech na vzory. Po jednotlivých průchodech vybírá technikou lineárního prahování vzory z kandidátů.

Tím jsme oddělili sémantiku ukládaných informací od jejich reprezentace. Nezáleží tedy na tom, s jakými daty aplikace pracuje. Pokud pro aplikaci vyhovuje námi implementovaná strategie generování, pak stačí doplnit vrstvu rozhraní na soubory. Pokud chceme provést ve strategii generování menší změny, lze předefinovat příslušné metody tříd generátoru a upravit parametry šablon. Pro vytvoření od základu jiné strategie by mohla posloužit alespoň vrstva pro uložení jazyka.

Samozřejmě za větší obecnost a flexibilitu platíme snížením výkonu. Zatímco třeba výstupem vzoru PatGENu je odkaz do hashovací tabulky obsahující dvojici ⟨číslo úrovně, pozice⟩, my musíme mít jako výstupní abecedu třídu s kopírovacím konstruktorem. Naše první zkušenosti ukazují, že na reálných datech (slovníku 170 tisíc slov) je reimplementace PatGENu postavená na knihovně PatLib nejvýše šestkrát pomalejší. Nutno podotknout, že kód současné verze ještě není optimalizován, v této fázi jsme se snažili zachovat „konceptuální čistotu“.

Na OPatGen samotný už zbývá jen realizovat konkrétní rozhraní na soubory. Situaci nám poněkud komplikuje požadavek co možná největší zpětné kompatibility s PatGENem. I v OPatGENu tedy najdeme možnost použití „translate“ souboru. OPatGen může pracovat ve dvou režimech, osmibitovém ASCII nebo plném UTF-8 UNICODE. Rozhodli jsme se manipulaci s UNICODE zvládat ve vlastní režii. Tím dosáhneme větší nezávislosti na systému a možnosti implementace plného UTF-8. Většina běžných operačních systémů dnes totiž podporuje UTF-8 jen omezeně, často pouze do 16 bitů.

Součástí distribuce produktu je také uživatelský manuál, ve kterém jsou zvláště vyznačeny (drobné) rozdíly mezi PatGENem a OPatGENem, aby byl přechod mezi těmito programy co nejsnazší.

Aplikace techniky vzorů u následníků TEXu

"But at least I can point out a minor weakness of TEX's algorithm: all possible hyphenations have the same penalty. This might be ok for english, but for languages like German that have a lot of composite words there should be the ability to assign lower penalties between parts of a composite i.e. Um-brechen should be favored against Umbre-chen."
—Florian Hars [7]

Dělení slov

Připomeňme si, kdy TEX slova dělí. V následujícím popisu zanedbáváme detaily, které nejsou podstatné pro následující úvahy. Pro přesný popis odkazujeme na [12] nebo články [17, 18].

Algoritmus zlomu odstavce má nejvýše tři průchody. V prvním průchodu se slova nedělí a hledá se řešení, při kterém mají všechny řádky hodnotu badness menší nebo rovnu `\pretolerance`. Pokud takové řešení neexistuje, nastupuje druhý průchod.

Ve druhém průchodu se do slov odstavce přidají symboly pro možné dělicí body. Pak se vyhodnocují všechny zlomy, pro něž platí, že všechny řádky mají badness nejvýše `\tolerance`. Pokud neexistuje přijatelné řešení, ve třetím průchodu se navíc povolí roztažitelné mezislovní mezery.

Tento algoritmus má však nepříjemné omezení pro jazyky, v nich jsou častá složená slova. Švy složených slov jsou typografy považována za místa pro dělení vhodnější, ostatní místa jsou vhodná méně. Bohužel TEX nedovoluje klasifikovat dělicí body, pro libovolné místo vybrané jako možný dělicí bod ve druhém průchodu algoritmu zlomu odstavce má pouze jedinou možnou `\hyphenpenalty`.

Možným řešením by bylo vytvoření dvojice vzorů, jedna sada vzorů pro švy složených slov, druhá pro všechny dělicí body. Navrhujeme zavedení penalty za dělení slova na švu, `\compoundhyphenpenalty`. Ta by byla menší než `\hyphenpenalty` a tím by bylo preferováno dělení na švech slov. To samozřejmě vyžaduje zásah do algoritmu zlomu odstavce, tedy se může týkat pouze některého z následníků TEXu.

Rozšíření o tuto klasifikaci dělení přináší otázku, kdy a jak je během zlomu odstavce provádět. Jednou možností je nahradit současný průchod, kdy se dělení provádí, průchodem, ve kterém se zjistí dělicí místa na švech slov a nastaví se jim `\compoundhyphenpenalty`. Dále se zjistí ostatní vhodné dělicí body a těm, které ještě nemají nastavenou penaltu (tj. nejsou na švu slova), se nastaví hodnota `\hyphenpenalty`.

Jinou možností je místo současného průchodu s dělením slov implementovat průchody dva. V prvním by se provedlo pouze dělení na švech slov a pokud by zlom odstavce byl dostatečně kvalitní, ponechal by se. Pokud ne, provedlo by se i dělení v dalších možných místech (s `\hyphenpenalty`) a odstavec by se lámal znovu.

Navíc znalost švů složených slov je výhodná i z dalšího důvodu. Typografická pravidla požadují, aby se na švech nevyskytovaly ligatury. Tedy například slovo šéflékař má být správně vysázeno šéflékař. Bohužel to je nutno TEXu sdělit ručně, proto jsem poslední slovo předchozí věty musel psát `šéf\hskip0ptlékař`.

Bylo by vhodné, aby se všechna slova ze vstupního proudu otestovala na švy složených slov a vstupní proud se příslušně upravil. To samozřejmě přináší další otázky, jako například, zda nevypustit první průchod zlomu odstavce a nezačít rovnou s povoleným dělením na švech slov.

Překladové procesy Ω
Systém Ω umožňuje téměř v jakémkoliv okamžiku zpracování textu ve svém zažívacím traktu aplikovat *překladový proces* (ΩTP, Omega Translation Process). Ten je dosud implementován pomocí, zjednodušeně řečeno, substitucí regulárních výrazů. Tato realizace dostačuje u jednoduchých zobrazení, pro složitější (například dělení slov) není dostatečně efektivní. Proto se pro složitá mapování (dělení slov, aplikace ligatur v daném jazyce, spelling či dokonce grammar checker, kontextové zpracování znaků v arabštině) nabízí pro generování a uložení těchto informací použití techniky vzorů. Knihovna PatLib by pak hrála v efektivní implementaci těchto nástrojů klíčovou roli.

Shrnutí

'I když se neprosadím, chtěl bych věřit, že bude někdo pokračovat v tom, co jsem započal. Ne bezprostředně, ale člověk není sám ve víře v opatrnost.'
—Vincent van Gogh

V článku jsme popsali techniku vzorů, možnosti jejího využití a návrh knihovny pro práci se vzory.

Zdrojové kódy knihovny PatLib a generátoru OPatGen v jejich aktuálních vývojových verzích spolu s dokumentací a dalšími materiály naleznete na adrese `http://www.fi.muni.cz/~xantos/PATLIB`. Doufáme ve využití PatLibu v širokém spektru aplikací, od implementace OPatGenu, přes implementaci překladových procesů sázecího systému Ω, po aplikace v oblastech zpracování přirozeného jazyka či grafiky.

Odkazy

[1] Cezar Câmpeanu, Nicolae Sânteau, and Sheng Yu. Minimal cover-automata for finite languages. In Champarnaud et al. [2], pages 43–56.

[2] Jean-Marc Champarnaud, Denis Maurel, and Djelloul Ziadi, editors. *Automata Implementation, Third International Workshop on Implementing Automata, WIA '98*, Berlin, Heidelberg, 1999. Springer-Verlag.

[3] David Antoš and Petr Sojka. Generování vzorů dělení slov v UNICODE. Str. 23–32, Brno, Czech Republic, Feb 2001. Konvoj.

[4] Patrick Hanks, editor. *The New Oxford Dictionary of English*. Oxford University Press, Oxford, 1998.

[5] Yannis Haralambous. A Small Tutorial on the Multilingual Features of PATGEN2. in electronic form, available from CTAN as `info/patgen2.tutorial`, January 1994.

[6] Yannis Haralambous and John Plaice. Methods for Processing Languages with Omega. In *Proceedings of the Second International Symposium on Multilingual Information Processing, Tsukuba, Japan*, 1997. available as `http://genepi.louis-jean.com/omega/tsukuba-methods97.pdf`.

[7] Florian Hars. Typo-l email discussion list, 4 January 1999.

[8] Piet Hein. *Grooks*. MIT Press, Cambridge, Massachusetts, 1966.

[9] Douglas R. Hofstadter. *Gödel, Escher, Bach: An Eternal Golden Braid*. Basic Books, 1979.

[10] Tao Jiang, Arto Salomaa, Kai Salomaa, and Sheng Yu. Decision problems for patterns. *Journal of Computer and Systems Sciences*, 50(1):53–63, 1995.

[11] Lauri Karttunen, Tamás Gaál, and André Kempe. Xerox finite-state tool. Technical report, Xerox research Centre Europe, Grenoble, June 1997. `http://www.xrce.xerox.com/research/mltt/fssoft/docs/fst-97/xfst97.html`.

[12] Donald E. Knuth. *The TEXbook*, volume A of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986.

[13] András Kornai. *Extended Finite State Models of Language*. Cambridge University Press, 1999.

[14] Franklin M. Liang and Peter Breitenlohner. `PAT`tern `GEN`eration program for the TEX82 hyphenator. Electronic documentation of `PATGEN` program version 2.3 from web2c distribution on CTAN, 1999.

[15] Mehryar Mohri, Fernando C.N. Pereira, and Michael D. Riley. FSM Library — General-purpose finite-state machine software tools, 1998. `http://www.research.att.com/sw/tools/fsm/`.

[16] Emmanuel Roche and Yves Schabes. *Finite-State Language Processing*. MIT Press, 1997.

[17] Petr Sojka. Notes on Compound Word Hyphenation in TEX. *TUGboat*, 16(3):290–297, 1995.

[18] Petr Sojka. Hyphenation on Demand. *TUGboat*, 20(3):241–247, 1999.

[19] Petr Sojka. Competing Patterns for Language Engineering. Lecture Notes in Artificial Intelligence LNCS/LNAI 1902, pages 157–162, Brno, Czech Republic, Sep 2000. Springer-Verlag.

[20] Petr Sojka and Pavel Ševeček. Hyphenation in TEX — Quo Vadis? *TUGboat*, 16(3):280–289, 1995.

[21] Alan Turing. Computing machinery and intelligence. *Mind*, (59):433–460, 1950.

Summary: Pattern Generation using PatLib Library and Program OPatGen

Paper describes technique of competing patterns as a method for data mining and effective storage. Development of time- and space-effective hyphenation algorithm from already hyphenated word list is a typical application.

The program PatGen, being nearly twenty years old, doesn't suit today's needs (limitation to eight-bit encodings, monolithic, hard to maintain code, etc.). A new pattern generator, OPatGen, suitable for system Ω, has been designed and implemented from scratch in object-oriented manner. An architecture of OPatGen is outlined. It is based on generic library PatLib for pattern handling.

Possible applications of the pattern technology are listed (multi-level and compound word hyphenation, Thai segmentation, optical character recognition).

David Antoš, Petr Sojka
Fakulta informatiky Masarykovy university, Brno
{xantos|sojka}@informatics.muni.cz

Používáme pdfTEX II: prezentace fotografií aneb jak na hypertext

Vít Zýka

V prvním dílu [8] jsme se seznámili s vkládáním obrázků do pdfTEXu. V tomto článku vytvoříme dokument obsahující naskenované fotografie a zamýšlený k projekci data-projektorem. Na této aplikaci si ukážeme tvorbu hypertextových odkazů, záložek a náhledů.

Prezentace fotografií

Také máte fotografie, které byste chtěli ukázat více lidem najednou? Řešením je fotografie naskenovat a pomocí nějakého vhodného softwaru je v celoobrazovkovém režimu promítnout data-projektorem. Je zřejmé, že taková projekce nedosáhne kvality diapozitivu (typické rozlišení data-projektoru je 1024×768 bodů), ale zase máme možnost obraz dodatečně upravit, třeba zvolit lepší kompoziční výřez nebo dorovnat kontrast či barevné podání. Podle mých zkušeností je na povídání o zážitcích z dovolené kvalita promítání uspokojivá.

K projekci se mi osvědčil Acrobat Reader. Umí celoobrazovkový režim, zobrazovat náhledy a záložky pro snazší orientaci, je rozšířený a zdarma. Také umožňuje pohyb po fotografiích pouze pomocí myši, což ocení zejména ti, kteří mají myš zabudovanou do dálkového ovladače projektoru, a nechtějí chodit ke klávesnici nebo hledat klávesy v zšeřelé místnosti. PDF dokument pro Reader vytvářím plain formátem pdfTEXu.

Uživatelský soubor specifikující fotografie

Nejdříve načteme soubor maker, pak uvedeme popisek, který se vloží jako název kořenové záložky a celkový počet fotografií

```
\input vzdia.tex
\diaset{Altaj 2001}{26}
```

a dále, po specifikaci adresáře, základu a koncovky jmen souborů skenů, vkládáme jednotlivé fotografie. U každé uvedeme její popis do záložky a rozlišovací část jména souboru (první jméno fotky je `altaj01.jpg`).

```
\def\diapath{../foto/}  \def\diabase{altaj}  \def\diaext{.jpg}
\dia{Most přes Katuň}{01}
\dia{Belucha}{05}
```

```
% ...
\dia{Příjezd do Čemalu}{70} \end
```

Implementace maker v souboru `vzdia.tex`

Fotografie na celou stránku

Naskenované fotky vkládáme makrem `\dia`.

```
\def\dia#1#2{\tooutline{#1}{0}
  \centerline{\putimage{\diapath\diabase#2\diaext}}\eject}
```

Název fotografie, který je jeho prvním parametrem, se na řádce 8 použije k vytvoření záložky. Jim se budeme věnovat na straně 16. Fotografii umístíme do vycentrovaného řádku a odstránkujeme pomocí `\eject`. V makru `\putimage` zajistíme, aby fotka na stránce zaujímala co největší rozměr.

```
\def\putimage#1{{\pdfximage height\pdfpageheight{#1}%
  \setbox0=\hbox{\pdfrefximage\pdflastximage}%
  \ifdim\wd0>\pdfpagewidth\pdfximage width\pdfpagewidth{#1}%
  \else\pdfximage height\pdfpageheight{#1}\fi%
  \hbox{\pdfrefximage\pdflastximage}}}
```

Na řádce 10 ji cvičně načteme do boxu tak, aby byla vysoká přesně na výšku stránky. Protože pdfTEX sám proporcionálně dopočítá i šířku, můžeme porovnat šířku tohoto boxu s šířkou stránky a podle toho ji definitivně vložit s největším možným zvětšením.

V minulého dílu [8] jsem omylem uvedl starší syntaxi parametrů primitivu `\pdfximage`. To sice platí, ale dnes můžeme použít přímější zápis atributů objektu, včetně geometrické transformace:

`\pdfximage [⟨rule spec⟩] [attr{⟨attributes⟩}] [page` *n*`] file{`*filename*`}`

Takto můžeme do nějakého jiného dokumentu vložit druhou stranu tohoto článku zmenšenou na šířku 6 centimetrů: `\pdfximage width6cm attr{/Matrix [0 1 -1 0 595.3 0] /BBox [0 160 50 0] } page 2 {dia-bul.pdf}`. A co se se stránkou ještě stalo? Byla oříznuta 160 bp zespodu a 50 bp zprava, dále otočena doleva o 90° a posunuta o 21 cm doprava.

Stránky nebudou obsahovat záhlaví ani zápatí. Dále chceme, aby fotografie dosahovala až k okraji stránky a byla vertikálně vycentrovaná. Proto předefinujeme výstupní rutinu, aby nevkládala dodatečné okraje. Nebojte se, bude velmi jednoduchá.

```
\output={\shipout\diabox \advancepageno}
\def\diabox{\vbox to\vsize{
  \vbox to0pt{\hrule width\hsize height\vsize depth0pt\vss}
  \vfil\unvbox255 \vfil}}
```

Trochu vysvětlení si zaslouží řádek 17. Do výstupního vertikálního boxu vkládá linku o velikosti celé stránky zajišťující černé pozadí fotografie. Je uvnitř boxu nulové výšky, takže neposune bod sazby.

Odkazy (links and destinations)

Hypertextový odkaz může být externí (mimo dokument) a interní. Externí odkaz je zadáván pomocí URL (Unique Resource Locator), interní tvoří nejčastěji dvojice odkaz–doskok. V tomto případě jsou obě složky provázány jednoznačným identifikátorem v podobě čísla nebo jména.

Doskok

Doskok se definuje pdfTEX primitivem

`\pdfdest` ⟨`num` *n* | `name{`*refname*`}`⟩ ⟨`dest spec`⟩

Doskok pojmenovaný `PokusnyDoskok` zachovávající aktuální zvětšení dokumentu zadefinujeme takto

```
\pdfdest name{PokusnyDoskok} xyz
```

V naší aplikaci budeme potřebovat doskok na každou stránku s fotografií pro správnou funkci záložek. Musíme tedy vytvořit jméno jednoznačné pro každou stránku. Použijeme-li číslo stránky (je uložen v nultém registru), může vypadat takto

```
\pdfdest name{page:\the\count0} fit
```

Protože vždy budeme chtít zobrazit celou fotografii v největší velikosti, která se vejde do aktuálního okna, použili jsme místo `xyz` přepínač `fit` [7].

Někdy se hodí odkázat část dokumentu ve zvoleném zvětšení. To umožňuje doskok se syntaxí `\pdfdest name{CtyrikratZvetsi} xyz zoom 4000`. Referenční bod sazby není vložením doskoku nijak ovlivněn, ale je zaznamenána jeho poloha. Při doskoku má prohlížeč nastavit tento bod do levého horního rohu okna a zvětšit/zmenšit podle faktoru `zoom`/1000.[1]

Protože budeme vždy zobrazovat jednu fotografii na celou stranu, můžeme doskok umístit do kteréhokoliv místa na stránce, třeba před nebo za fotografii. Lepší však bude vložit jej přímo do výstupní rutiny. V ní bude zaručeno, že na každou stranu bude vložen právě jeden doskok bez ohledu, zda dojde k explicitnímu nebo automatickému zlomu strany[2]. Za řádek 16 přidáme

```
19  \pdfdest name{page:\the\count0} fit
```

Odkaz

Odkaz vytvoříme dvojicí primitivů

`\pdfannotlink` [⟨`rule spec`⟩] [`attr{`⟨`attributes`⟩`}`] ⟨`action spec`⟩

[1] Prohlížeč `xpdf` v0.91 parametr `zoom` zcela ignoruje. Acrobat naopak přidává další, vcelku užitečné akce. Je-li dokument v požadovaném zvětšení menší než okno prohlížeče, je vycentrován; je-li referenční bod příliš u okraje, takže by zobrazená část dokumentu nevyužívala celou plochu okna, náležitě jej posune.

[2] Explicitní stránkový zlom je vyvolán vložením `\penalty=-10000` ve vertikálním módu. Tato penalta je obsažena v plainovém makru `\eject` nebo LATEXovém `\newpage`. Automatický zlom strany provede TEX sám na základě optimalizačního algoritmu plnění strany [3, str. 238].

a \pdfendlink. Aktivní plocha linku je vypočítána z obsahu mezi oběma primitivy. Uvnitř může dojít k řádkovému i stránkovému zlomu, ale musí být ve stejné úrovni boxu nebo skupiny. Jinou možností je specifikovat aktivní plochu pomocí ⟨rule specification⟩. Makro, které vytvoří odkaz na jmenný doskok uvedený v jeho prvním parametru s aktivní plochou textu v druhém parametru, může vypadat takto

```
20 \long\def\hlink#1#2{\pdfstartlink attr{/Border [0 0 0] /H/N }
21   goto name{#1}#2\pdfendlink}
```

Uvedenými atributy požadujeme, aby prohlížeč nekreslil okolo linku žádný rámeček a neprovedl žádné zvýraznění při kliknutí (highliting=none). Celkový výčet možných atributů lze najít v referenčním manuálu [6].

Chtějme v naší aplikaci umožnit kliknutím skok z každé stránky na stránku předcházející. Aktivní plochou takového linku nechť je levá třetina stránky. Nastavené atributy zajistí ‚neviditelnost' odkazu, takže nebude rušit při prohlížení fotek. Odkaz znovu umístíme do výstupní rutiny, za řádek 17.

```
22   \count1=\count0 \advance\count1 by-1
23   \vbox to0pt{\hbox{\hlink{page:\the\count1}{%
24     \vbox to\vsize{\hrule width.3\hsize height0pt depth0pt
25       \vss}}}\vss}
```

I zde je vše vloženo do boxu o nulových rozměrech, aby se neposunul bod sazby. Na řádku 24 vytvoříme box o rozměrech třetiny stránky, a ten vložíme jako druhý parametr makra \hlink. Jmenný odkaz musíme vytvořit stejně, jako pro doskok na řádku 19, jen číslo aktuální stránky zmenšíme o jednu (řádek 22).

Jinou možností, jak přejít na předchozí stránku, je využít tzv. *menu akce* (uživatelsky pojmenované akce). Tento název byl zvolen proto, že tímto způsobem jsou dosažitelné vesměs ty operace, které umožňuje Acrobat ve svých menu. Tedy například `NextPage`, `PrevPage`, `FirstPage`, `LastPage`.[3] Jejich použití umožňuje makro podobné \hlink, jen prvním parametrem bude jméno akce.

```
\long\def\menulink#1#2{\pdfstartlink attr{/Border [0 0 0] /H /N}%
  user{/Subtype /Link /A << /S /Named /N /#1 >> }#2\pdfendlink}
```

Záložky (outlines)

Záložky tvoří jakýsi aktivní obsah usnadňující orientaci v dokumentu. Dobrý prohlížeč je umí zobrazit paralelně s textem. Každá záložka obsahuje zobrazený text, akci (většinou specifikaci doskoku) a počet jejích vnořených záložek. Tento počet určuje hierarchickou strukturu spolu s pořadím vložení záložek.

Vraťme se k našemu příkladu. Na řádku 2 jsme uvedli údaje pro vytvoření kořenové záložky. Kód makra \diaset je totožný s kódem \tooutline, které záložky vytváří a jehož vysvětlení dlužíme čtenáři od prvního použití v makru

[3] Mezi další akce uveďme `GeneralInfo`, `Print` nebo `GoToPage`. Ale pozor! Tyto další pojmenované akce způsobí, že dokument nebude podle referenčního manuálu ‚portable' . Krom toho je podporuje až Acrobat verze 3 a výše.

`\dia` na řádku 8. Synonymum vytvoříme jen proto, abychom nemátli uživatele.

```
28 \let\diaset=\tooutline
```

Definice

```
29 \input il2ascii.tex
30 \def\tooutline#1#2{\iltwoascii{#1}%
31   \pdfoutline goto name{page:\the\count0} count #2{\ascii}}
```

je založena na primitivu

`\pdfoutline` ⟨`action spec`⟩ `[count` ***n***`]` {⟨`general text`⟩}

Celé číslo $|n|$ označuje počet přímých podzáložek. Je-li $n < 0$, pak v počátečním stavu budou podzáložky svinuté. Vynecháme-li parametr `count` ***n***, bude se považovat za nulový. V našem příkladě bude po úvodní titulní záložce odkazující na první fotografii dalších 26 rozvinutých podzáložek. Pro každou ze šestadvaceti fotografií ji přidá makro `\dia`.

Dále jsme na řádku 29 načetli makra pro převod znaků v kódování IL2 na sedmibitové nehackovane ASCII. To souvisí s tím, že Acrobat záložky zobrazuje pouze systémovým fontem v kódování IL1 nebo UNICODE.[4] Makro `\iltwoascii` zadefinuje `\ascii` obsahující překódovaný text.

Chceme-li mít záložky správně česky nebo slovensky, překódujeme vstup do UNICODE. Inspiraci i funkční prototyp tohoto postupu z kódování IL2 nabídl Jirka Osoba v odpovědi v konferenci CSTEXu `gopher://cs.felk.cvut.cz:70/0R63946-69054-gopher_root%3A%5B_lists.cstex%5Dcstex.2002-02%3B1`. Použijeme je pak takto:

```
32 \input 88592-unicode.tex
33 \def\tooutline#1#2{\toUNICODE{#1}
34   \pdfoutline goto name{page:\the\count0} count #2{\UNICODE}}
```

Náhledy stránek (thumbnails)

Pro vytvoření miniaturních náhledů stránek vytvořil Heiko Oberdiek balíček `thumbpdf`. Funguje pro plain TEX i LATEX a do zdrojového souboru pdfTEXu stačí vložit

```
35 \input thumbpdf.sty
```

přeložit TEXem, pak spustit skript v Perlu `thumbpdf` ***tex_file_without_ext*** a na závěr ještě jednou přeTEXovat. Kromě Perlu je potřeba mít ještě nainstalován GhostScript.

Acrobat (Reader) umožňuje vyvolat panel s náhledy, a to i v celoobrazovkovém režimu.[5] Tato funkce se s úspěchem využije při hledání vhodného slajdu při prezentacích. Tlačítko vyvolávající tento panel lze ukázat pomocí

```
\menulink{ShowThumbs}{Náhledy}
```

[4] Tento šestnáctibitový font však nemusí být vůbec nainstalován, jako na mém Acrobat Readeru v4 pro Linux. Pro přenositelnost dokumentu je tedy sedmibitová ASCII vhodnější, byť přijdeme o diakritiku.

[5] Bohužel Acrobat Reader v4 pro Linux neumí panel s náhledy zavřít, čímž je tato funkce pod Linuxem nepoužitelná; `xpdf` ani `gv`+`gs`, pokud vím, náhledy ani záložky v současné době nezobrazují.

Rozměry a další parametry

Tím jsme nadefinovali všechna makra našeho balíčku pro prezentaci fotografií. Ještě zbývá nastavit několik parametrů, hlavně velikost stránky.

```
\pdfpageheight=210 truemm \vsize=\pdfpageheight
\pdfpagewidth =280 truemm \hsize=\pdfpagewidth
\hoffset=-1in \voffset=-1in
\parindent=0pt
\offinterlineskip
```

Výška byla nastavena na výšku nejběžnějšího tiskového formátu A4 – to kdybychom chtěli nějakou fotku tisknout. I tak však budeme muset zapnout parametr tisku `Fit to Page`. Šířku papíru jsme zadali tak, aby poměr stran byl 3:4, tak jak je u monitorů a projektorů zvykem.

Zřetězení článku (atricle threads)

Ještě si popíšeme použití hypertextového objektu, které autoři lokalizace Acrobatu nazývají *zřetězením článku*. S tématem tohoto příspěvku souvisí, protože umožňuje rychlejší navigaci ve složitějším dokumentu. Je-li třeba text v několika sloupcích a vložíme-li každý sloupec do takového vlákna, pak prohlížeč (Reader) kliknutím skočí do prvního sloupce, roztáhne jej na šířku okna, dalším klikáním jej bude vertikálně posouvat, dokud nenarazí na jeho dolní konec a pak zobrazí začátek dalšího sloupce. Dospějeme-li ke konci posledního sloupce, který jsme do vlákna vložili, zobrazí se zase stránka tak, jak byla před vstupem do vlákna. Jiným příkladem použití může být skupinová fotografie, kterou nejdříve chceme zobrazit celou a pak postupně ukázat detaily každého jedince. S pomocí zřetězeného článku se jedním kliknutím posuneme do dalšího výřezu.

Mohlo by se zdát, že tento mechanismus lze nahradit soustavou obyčejných odkazů a doskoků. Vidím zde však dva hlavní rozdíly:

1. Zřetězené články se mohou překrývat. Vstoupíme-li do nějakého stavu článku, pak je aktivní celá plocha okna pro posun dále (se shiftem zpět) v článku. Plochy ostatních článků v dokumentu nebo jiných stavů tohoto článku jsou momentálně neaktivní.
2. Nevejde-li se vertikálně celý blok textu zřetězeného článku do okna, zobrazovač jím (na uživatelovo kliknutí) pohybuje, dokud jej celý nezobrazí, a teprve pak skočí do dalšího stavu článku.

Na vytvoření jednoho stavu článku slouží primitiv

`\pdfthread` [⟨`rule spec`⟩] `[attr{`⟨`attributes`⟩`}]`
⟨`num` *n* | `name{`*refname*`}`⟩

nebo dvojice `\pdfstartthread` `\pdfendthread`. Syntaxe `\pdfstartthread` je stejná jako u `\pdfthread`. Dvojice značek se musí objevit ve vboxu stejné úrovně. Obsah pak stanoví velikost aktivní oblasti stavu článku. U jednoduché značky

se velikost dopočítá (není-li udána pomocí ⟨rule spec⟩) podle rodičovského boxu. Velikost oblasti se dále zvětší o hodnotu registru `\pdfthreadmargin`.

Jak na to v LaTeXu?

Zde uvádím přehled popsaných možností pdfTeXu pro uživatele LaTeXu. Spočívá v načtení patřičného makrobalíku a použití uživatelsky jednoduchého makra. Nejprve splatíme dluh z minulého dílu.

Vkládání obrázků

```
\usepackage[pdftex]{graphicx}
\includegraphics{file}
```

Pokud jméno souboru neuvedeme s příponou, bude se defaultně hledat `.pdf`, je-li při vložení balíčku `graphicx` uveden parametr `pdftex`. Bez tohoto parametru se bude hledat `.eps`. Chceme-li, aby náš dokument šel přeložit jak pdfLaTeXem, tak obyčejným LaTeXem, použijeme

```
\usepackage{ifpdf}
\ifpdf \usepackage[pdftex]{graphicx}
\else\usepackage{graphicx}\fi
```

a neuvedeme příponu jména souboru; pdfTeX navíc bude umět načíst i soubory typu `.png`, `.jpg` a `.tif`. Obrázku z METAPOSTu musíme dát koncovku `.mps` (nebo udělat link např. `ln -s mpfile.1 mpfile1.mps`) a s touto koncovkou jej načíst.

Balík `graphicx` hledá obrázky ve stejných adresářích jako LaTeX makrobalíky[6] plus v adresářích, které nastavíme `\graphicspath{{fig/},{../photo/}}`. Doporučená literatura: [5, 1].

Hypertext

```
\usepackage{hyperref}
\hypertarget{label}{}
\hyperlink{label}{contents}
\href{url}{contents}
```

Chceme-li, aby link nevytvářel okolo `contents` rámeček, nebarvil jej a nezpůsobil jeho inverzi při kliknutí, pak použijeme následující makro `\shyperlink`

```
\def\noBorderLinkAttr{\makeatletter%
  \def\pdfBorderAttrs{/Border [0 0 0] /H /N }%
  \def\hy@colorlink##1{\begingroup}
```

[6] Ve Web2C instalaci (TeXLive) viz řádek `TEXINPUTS.pdflatex=...` v souboru `texmf.cnf`.

```
    \let\Hy@colorlink=\hy@colorlink
    \def\@pdfhighlight{/N } \def\@pdfborder{0 0 0}
    \def\@urlcolor{black}%
    \makeatother}
  \long\def\shyperlink#1#2{{\noBorderLinkAttr\hyperlink{#1}{#2}}}
  \long\def\shref#1#2{{\noBorderLinkAttr\href{#1}{#2}}}
```

Doporučená literatura: [2, 4].

Závěr

Budeme-li chtít rozdělit fotografie do několika skupin – třeba fotografie Filipa, Hanky a Milana, a dále fotografie z pobytu v horách a fotografie ze splutí řeky – můžeme přidat do záložek podtitulky s vlastní rozbalovací skupinou záložek. Protože se tak některé fotky mohou vyskytnout v několika skupinách zároveň, vyplatí se použít referování jednou použitých fotografií, jak bylo popsáno v minulém čísle Zpravodaje. Dokument tak nebude zbytečně narůstat tím, že jeden obrázek je zároveň na několika stránkách.

Na jednoduché aplikaci jsme ukázali základní vytváření hypertextových odkazů, záložek a náhledů. Kód maker je přístupný na adrese `ftp://cmp.felk.cvut.cz/pub/cmp/users/zyka/dia`.

Odkazy

[1] David P. Carlisle. *Package in the 'graphics' bundle*, 1999. `TeXLive6: \$TEXMF/doc/latex/graphics/grfguide.ps.gz`.

[2] Heiko Oberdiek. *PDF information and navigation elements with hyperref, pdfTEX, and thumbpdf*, 1999. `TeXLive6: \$TEXMF/doc/latex/hyperref/slides.pdf`.

[3] Petr Olšák. *TEXbook naruby.* Konvoj Brno, 1. edition, 1997. Czech.

[4] Sebastian Rahtz. *Hypertext marks in LATEX: the* `hyperref` *package*, 1998. `TeXLive6: \$TEXMF/doc/latex/hyperref/manual.pdf`.

[5] Keith Reckdahl. *Using Inported Graphics in LATEX 2ε*, 1997. `TeXLive6: \$TEXMF/doc/guides/epslatex/epslatex.ps.gz`.

[6] Adobe systems Incorporated. *PDF reference manual, v. 1.4*, second edition, 2000. `http://partners.adobe.com/asn/developer/acrosdk/DOCS/PDFRef.pdf`.

[7] Hàn Thê Thành, Sebastian Rahtz, and Hans Hagen. *The pdfTEX user manual.* Pragma, 2001. `http://www.tug.org/applications/pdftex/pdftex-s.pdf`.

[8] Vít Zýka. Používáme pdfTeX: vkládání obrázků. *Zpravodaj Československého sdružení uživatelů TEXu*, 11(4):181–186, December 2001.

Summary: Using pdfTEX II: document for photographs presentation; how to create a hypertext

This article describes making hypertext objects by pdfTEX. We focus on the hyperlinks, bookmarks, thumbnails, and article threads. The usage is demonstrated on a simple application–the document for photographs projection. We show creating hypertext on both the pdfTEX primitive level and with standard LATEX macro-packages. This text follows the last issue article *Using pdfTEX: graphics inclusion*, CSTUG Zpravodaj 4/2001.

Vít Zýka, zyka@cmp.felk.cvut.cz

Textové mezery v TEXu

Josef Tkadlec

S tím, jak roste kvalita tisku dostupného běžnému uživateli, roste i význam respektování typografických pravidel. Máme-li k dispozici kvalitní fonty, pak i „drobnosti“ mohou významně přispět k přiblížení se typografickému ideálu: k sazbě, která v hrubém pohledu vypadá jednolitě šedivě (takže při četbě neruší nepatřičnými bílými plochami) a přitom jsou zřetelně oddělena jednotlivá slova či logické celky. Někomu připadají některá typografická pravidla jako zbytečná nebo jen jako „pouhá“ estetická záležitost. Myslím si, že valná většina má i ryze praktický význam – umožňují, aby se co nejvíce snížilo čtenářovo úsilí potřebné k „dekódování“ napsaného textu.

V tomto článku se budu zabývat *českými* typografickými pravidly psaní mezer, některé národní typografie mají pravidla odlišná. Například v českých typografických pravidlech se upřednostňuje rovnoměrná šedivost stránky, zatímco jinde spíše vyznačování logických celků a grafické znázornění pomlk při čtení – to se projevuje například ve volbě mezery za tečkou na konci věty (zúžená nebo naopak rozšířená). Nemyslím si, že by česká pravidla byla ta nejlepší a jiná méněcenná, oba přístupy mají svá pro a proti.

Typografická pravidla ve starší literatuře [1, 3] vycházejí z omezených možností ruční sazby (hodně diskrétní změny velikostí) a jsou uváděna spíše na příkladech. Pro počítačovou sazbu jsou jednak zbytečně hrubá, jednak nevhodně formulovaná. Novější literatura se zase zabývá spíše některým konkrétním produktem, vychází z mezer, které má k dispozici, a dost často se vyskytují obraty

typu „mělo by to být jinak, ale protože to nejde (rozuměj: konkrétní produkt to neumí), uděláme to takto“. V tomto článku se pokouším podat přehledný, ucelený a zdůvodněný systém pracující se spojitými změnami velikostí, kategorizující psaní mezer podle významu a použití (ne podle konkrétních velikostí) a náměty na TEXnická řešení.

Snažím se nezavádět nové pojmy, i když v některých situacích to vede k poněkud krkolomnějšímu vyjadřování. Bohužel, v typografické literatuře se objevují různé terminologické nejasnosti a nesrovnalosti. *Šířkou znaku* se v definicích pevné a poloviční mezery myslí nejspíše šířka boxu, v definicích zúžené a základní mezery spíše šířka kresby. Pod pojmem *pevná mezera* se někde rozumí mezera, ve které nedojde ke zlomu řádku, někde (navíc) mezera konstantní šířky, někde dokonce jeden konkrétní typ mezery – tohoto se zde přidržím. Nejproblematičtější je asi používání pojmu *zúžená mezera*. Ten se vyskytuje všude tam, kde se jedná o mezeru menší, než je nějaké stanovené konstantní optimum, přičemž se nerozlišují různé situace:

1. Chceme opticky vyrovnat světlost – za čárkou, tečkou, apostrofem, kolem pomlčky, … Pouze v takovém případě mluvím o *zúžené* mezeře.
2. Chceme zvýraznit blízkost či sounáležitost výrazů, např. cifer čísla nebo čísla a jednotky. Pak mluvím o *pevné* mezeře.
3. Chceme dát na řádek více textu. Pak mluvím o *stahování* a tuto mezeru nijak zvlášť neoznačuji – mezery považuji obecně za pružné, mluvím jen o jejich velikostech: základní, nejmenší, největší.

Rozměry uvádím v `em` (kuželka, čtverčík) a přestože číselné hodnoty pro mezery jsou vesměs podíly malých celých čísel – v pravidlech násobky 1/12, v TEXu (přibližně) násobky 1/18 – používám zápis v desetinném tvaru, protože jednotlivé hodnoty tak lze snáze porovnat. Body v české tradici jsou tzv. Didotovy body (v TEXu `dd`), které jsou o něco větší, než body v americké tradici (`pt`), a ty jsou nepatrně menší, než v současnosti prosazované body (`bp`). V tomto textu je použit font `csr10`, pro jiné písmo mohou být poměry jiné, ale principy zůstávají stejné. Formulace pravidel vycházejí z [1].

Děkuji P. Olšákovi za pomoc při TEXnických řešeních, veškeré nedostatky ovšem padají na moji hlavu.

1. Nerozdělitelná mezera

Nazývá se také *nedělitelná* nebo *nezlomitelná*, nemá v ní dojít ke zlomu řádku. Používá se v těchto případech:

- Za jednopísmennými předložkami a spojkami (k, o, s, u, v, z, a, i), jak verzálkami, tak minuskami. Výjimka je povolena pro minusku „a“, v [3] se uvádí možnost výjimky i pro ostatní v úzké sazbě do 25 liter.
  Setkal jsem se s následujícím upřesněním výjimky pro „a“: v případě, že se jedná o větnou spojku ve významu slučovacím. Tím se sjednocuje situace se spojkou „i“ („ten a ten“ je podobné jako „ten i ten“), zabrání se nevhodnému dělení „a proto“ apod., postihnou se (některé) situace, kdy je spojka „a“ za čárkou – myslím si, že i tyto případy (obecněji za interpunkcí nebo závorkou) by měly být ošetřeny. Nejjednodušší je této výjimky nevyužívat, zhoršíme tím ale možnost lámání řádků.
- Mezi zkratkou titulu nebo jména a příjmením (dr. Vávra, S. Kovář, …).
- Mezi číslicí a označením počítaného předmětu (100 metrů, 100 m, 10. kapitola, …).
- Mezi dnem a měsícem data. Je-li datum vyjádřeno jen čísly, pak i mezi měsícem a rokem (podle [2] je povolena výjimka v úzké sazbě).
- V ustálených zkratkách (př. n. l., a. s., …).

2. Základní mezera

Její základní velikost má být $0{,}33\overline{3}$ em, její šířka má být v rozmezí mezi šířkami písmen „l“ a „n“. Podle [3, 2] nemá přesahovat 0,5 em, což je ztotožňováno se šířkou písmene „n“, podle [2] nemá být menší než 0,25 em. Pro úzké nebo vysoké písmo je velikost úměrně menší, pro široké písmo úměrně širší.

V TEXu se standardně použije mezera navržená ve fontu těmito parametry: základní velikost (`\fontdimen2`), největší „dobré“ rozpálení (`\fontdimen3`) a největší stažení (`\fontdimen4`). Pro `csr10` je to po řadě $0{,}33\overline{3}$ em, $0{,}16\overline{6}$ em a $0{,}11\overline{1}$ em, nejmenší velikost je tedy $0{,}22\overline{2}$ em a největší „dobrá“ velikost je 0,500 em, což odpovídá šířkám kreseb písmen „l“ a „n“ (šířky boxů jsou $0{,}27\overline{7}$ em a $0{,}55\overline{5}$ em).

Mezera se v TEXu může rozpálit na více než největší „dobrou“ velikost – ta odpovídá situaci, kdy `\hbadness` má hodnotu 100. Při větším rozpalování ale tato hodnota výrazně (se třetí mocninou) narůstá, navíc o ní máme informaci při překladu, takže to lze pohlídat.[1] Lze také nastavit základní velikost na 0,25 em a největší stažení na $-0,25$ em, což by vedlo k těsnější sazbě tak, jak je tomu u některých jiných sázecích programů.

Chceme-li nastavit nejmenší velikost základní mezery na 0,25 em, můžeme předefinovat největší povolené stažení (v `csplainu` je `\tenrm` už zavedeno):

```
\font\tenrm=csr10  \fontdimen4\tenrm=0.08333em
```

[1] V LATEXu se standardně vypisuje informace o „underfull“ boxu v případě, že je `\hbadness` větší než 1000, což pro `csr10` nastane při rozpálení na asi 0,7 em.

Lze také předefinovat `\spaceskip` na nenulovou hodnotu – v takovém případě má přednost před mezerou danou hodnotami `\fontdimen`.

Základní mezeru zadáváme ve zdrojovém textu obvykle znakem pro mezeru, nerozdělitelná varianta se zadává vlnkou (`~`).

3. Verzálková mezera

Při psaní verzálkami (velkými písmeny) je třeba používat větší mezeru, protože místa mezi slovy jsou méně prosvětlená. Její základní velikost má být 0,5 em a neměla by být širší než $0{,}66\overline{6}$ em. Podle [3] ji lze stáhnout na $0{,}33\overline{3}$ em. Při vyrovnávání (doplňování malých mezer mezi písmena) se úměrně zvětšuje.

Pokud není verzálková mezera pro dané písmo navržena, je asi nejvhodnější vyjít ze základní mezery a zvětšit ji o $0{,}16\overline{6}$ em – bere se tím do úvahy typ a velikost písma.

Nejpřirozenější zavedení v TEXu je asi předefinování hodnoty použité mezislovní mezery:

```
\def\CapCor {0.16667em\relax}
\def\capspaces {\spaceskip=\fontdimen2\font plus \fontdimen3\font
  minus \fontdimen4\font \advance\spaceskip by \CapCor}
\def\nocapspaces {\spaceskip=0pt\relax}
```

Takto zavedený příkaz `\capspaces` je nutno použít při každé změně fontu. Změna hodnoty `\spaceskip` je na rozdíl od změny `\fontdimen` lokální, takže příkaz `\nocapspaces` většinou není zapotřebí.

Pro krátké texty psané explicitně verzálkami se někdy doplňuje mezera navíc příkazem `\␣`. Pokud text nezarovnáváme do bloku (například v nadpisech), použije se základní velikost základní mezery a dospějeme k přijatelné hodnotě $0{,}66\overline{6}$ em. Je-li příkaz `\␣` předefinován na zúženou mezeru (viz část 5), dostaneme $0{,}58\overline{3}$ em a můžeme navíc upravit mezery např. mezi písmeny „A“ a „V“ (viz Ukázka 1). Lze také upravit definici `\␣` tak, aby se po mezeře přidalo právě $0{,}16\overline{6}$ em.

Méně praktické mi připadá zavedení zvláštního příkazu pro tuto mezeru nebo doplňování mezery navíc – například LATEXovským příkazem `\,`.

4. Mezera šířky tečky, pevná mezera

Mezera šířky tečky se používá při oddělování některých cifer v zápisu čísel (netýká se to čtyřciferných celých čísel v textu a letopočtů). Je nerozdělitelná (je to součást výrazu) a má šířku tečky (nejspíš boxu).

1) MEZERA VE SLOVECH LETNÍ OLYMPIÁDA

2) MEZERA VE SLOVECH LETNÍ OLYMPIÁDA

3) MEZERA VE SLOVECH LETNÍ OLYMPIÁDA

4) MEZERA VE SLOVECH LETNÍ OLYMPIÁDA

Ukázka 1. Mezery mezi verzálkami: 1) základní mezery, 2) verzálkové mezery se zúžením („A V“), 3) verzálkové mezery, 4) dvojnásobek základní mezery.

Pevná mezera[2] se používá mezi číslem a označením jednotky (nebo symbolem, např. %, §) a za pomlčkou použitou na začátku řádku jako vyznačení přímé řeči. Je nerozdělitelná, doporučované velikosti odpovídají nejmenší velikosti základní mezery, tj. 0,25 em.

Pevná mezera by nejspíš měla mít šířku právě nejmenší základní mezery – je nepružná a určitě by neměla být větší (chceme zdůraznit, že výrazy k sobě patří alespoň tak, jako slova), na druhou stranu má charakter základní mezery. Navíc by nejspíš neměla být menší než mezera šířky tečky – pokud napíšeme 10 000 km, pak k sobě cifry patří alespoň tak těsně, jako jednotka k číslu. Protože nejmenší základní mezera a šířka tečky jsou obvykle stejné (0,25 em), bývá velikost pevné mezery těmito podmínkami jednoznačně určena a bývá stejná jako šířka tečky.

V `csr10` je to složitější, protože tečka má šířku $0{,}27\overline{7}$ em a nejmenší základní mezera $0{,}22\overline{2}$ em. Je sice možné předefinovat stažitelnost základní mezery, ale nejspíš lze použít jak mezeru šířky $0{,}22\overline{2}$ em, tak kompromisně mezeru 0,25 em – rozdíly jsou malé, navíc asi ne příliš často dojde ke stažení základní mezery pod 0,25 em. Někdo používá mezeru $0{,}16\overline{6}$ em (nejspíš proto, že je jediná přijatelná nepružná, která je už zavedena), ta je ale dost malá (zvláště při rozpálení).

Mezera 0,25 em se doporučuje jednotně i v matematických vzorcích [1, 4]. TEXovský systém je propracovanější, základní velikosti mezer jsou: kolem relací $0{,}27\overline{7}$ em, kolem operací $0{,}22\overline{2}$ em, kolem operátorů, funkcí a za interpunkcí $0{,}16\overline{6}$ em.

Jako nejpřehlednější mi připadá použití „_“, případně s ponecháním původního významu `\sb` v matematickém módu. Méně problematické je použití příkazu `\:`.[3] Při používání různých stupňů písma je asi lepší vyrobit pevnou mezeru z příslušných `\fontdimen`. To je ukázáno pro `\SPpevna`, kde se navíc ošetřují i verzálkové mezery – předpokládá se, že pouze v takovém případě je použito nenulové `\spaceskip` podle části 3. Příkaz `\:` je zaveden jako mezera šířky tečky.

[2] V [1, 3] se označuje jako pevná zúžená mezera.

[3] Ten je v LATEXu definován jako ekvivalent `\>`, což odpovídá mezeře kolem operací.

1) Stojí to 10 000,00 Kč. Rychlostí 5 m/s jsme urazili 1/2 km.
Víme, že 5% roztok obsahuje 5 % látky. Koupili jsme 10 kg cukru.

2) Stojí to 10 000,00 Kč. Rychlostí 5 m/s jsme urazili 1/2 km.
Víme, že 5% roztok obsahuje 5 % látky. Koupili jsme 10 kg cukru.

3) Stojí to 10 000,00 Kč. Rychlostí 5 m/s jsme urazili 1/2 km.
Víme, že 5% roztok obsahuje 5 % látky. Koupili jsme 10 kg cukru.

Ukázka 2. Mezera šířky tečky a pevná mezera: 1) 0,16$\overline{6}$ em, 2) 0,22$\overline{2}$ em, 3) 0,250 em.

```
\def\:{\hphantom{.}}
\def\SPpevna {\kern\fontdimen2\font \kern-\fontdimen4\font
  \ifdim\spaceskip=0pt \else\kern\CapCor\fi}
\catcode`\_=\active  \def_{\ifmmode\sb\else\SPpevna\fi}
```

5. Zúžená mezera

Zúženou mezerou zde rozumím mezeru, která se zmenšuje proto, aby se opticky vyrovnala velká světlost u některých znaků. Má se požívat za čárkou, tečkou, apostrofem (odsuvníkem), před pomlčkou a za ní, před českými uvozovkami a za nimi, u písmen s větším světlem (A, T, P, V, W, Y, y, v, ...).

Asi nemá cenu ve zdrojovém textu řešit situace s písmeny (jedině snad při psaní verzálkami, tam to bývá znatelnější), to by měl sázecí program dělat automaticky. Celkově je ale lepší zužování používat – sazba dostává kompaktnější tvar, který vypadá lépe, obvykle se zlepší možnosti lámání řádků.

V některých národních typografiích se naopak doporučuje mezeru za některými interpunkčními značkami rozšířit. To je v TEXu implicitní nastavení (`\nofrenchspacing`), které lze zrušit příkazem `\frenchspacing`. Trochu jiným způsobem (mezerou, která je částí boxu) je to v `cmr10` řešeno pro uvozovky.

Ve všech uvedených případech se předpokládá zúžení o stejnou hodnotu. To je zjevně zjednodušení, mělo by záležet na tom, ve kterém případě je použito – protože to je ale otázka odborně typografická, nebudu se jí zde zabývat. Doporučované hodnoty zužování pro nižší stupně písma jsou uvedeny v tabulce 1. Pro větší stupně písma (podle [1] od 18 bodů) se má zužovat o polovinu šířky tečky. Nejspíše se myslí šířka kresby – tečka obvykle bývá široká 0,25 em a zužování o 0,125 em nezapadá do výše uvedeného schematu (ve větších stupních bývají velikosti mezer a tedy i zužování menší). V `csr10` má kresba tečky asi 0,11 em, takže vychází zužování o asi 0,055 em. Lepší je asi vycházet z příkladu pro 12bodové písmo, tj. zužovat o 0,08$\overline{3}$ em.

písmo	základní mezera	zúžená mezera	zúžení
8 bodů	0,375 em	0,250 em	0,125 em
10 bodů	0,400 em	0,300 em	0,100 em
11 bodů	$0{,}3\overline{63}$ em	$0{,}2\overline{72}$ em	$0{,}0\overline{90}$ em
12 bodů	$0{,}33\overline{3}$ em	0,250 em	$0{,}08\overline{3}$ em

Tabulka 1. Velikosti mezer podle [1, 3].

V [1] to vypadá, jako by hodnota 0,25 em byla jak velikost nejmenší základní mezery, tak konstantní velikost zúžené mezery.[4] Tento přístup je pochopitelný pro ruční sazbu (zjednodušuje ji), z logiky věci by se ale i zúžená mezera měla stahovat, měla by být o konstantní hodnotu menší než základní mezera. To odpovídá pravidlu z [2], podle kterého nejmenší mezera nesmí být menší než $0{,}16\overline{6}$ em – k této hodnotě dojdeme zúžením nejmenší základní mezery právě o $0{,}08\overline{3}$ em.

V `csr10` opět narážíme na to, že nejmenší základní mezera je menší než 0,25 em, takže bychom se při největším stažení dostali k příliš malé mezeře.[5] Měli bychom tedy se zužováním v určitém okamžiku přestat, čemuž se můžeme přiblížit tím, že zužování bude pružné – můžeme zmenšit povolené stažení mezery tak, aby nejmenší mezera byla uvedených $0{,}16\overline{6}$ em.

```
\def\zuzeni {\hskip-.08333em minus -.02778em\relax}
```

Přirozené by bylo zužování sčítat, kvůli špatné kontrole nejmenší hodnoty bych to ale dělal jen tehdy, je-li jeden ze znaků pomlčka – například dvojí zúžení mezi tečkou a otevírací uvozovkou by mohlo dopadnout nepříjemně.

5.1. Čárka, tečka, apostrof

V TEXu lze zužování mezery za tečkou, čárkou či apostrofem řešit přes aktivní znaky. V příkladu pro apostrof je ukázáno, jak zajistit standardní chování tohoto znaku v matematickém módu – je třeba mj. přepsat definici `\pr@m@s` v místě, kde je tento znak aktivní. Pokud to nepotřebujeme, můžeme si odpustit definici `\PRIME` a zjednodušit definici apostrofu na `\apost\TestZa`.

```
\def\apost {'}  \catcode`\'=\active  \let\PRIME='
\def'{\ifmmode\expandafter\PRIME\else\apost\expandafter\TestZa\fi}
\def\TestZa {\afterassignment\ZuzZa\let\tmp= }
```

[4] Možná proto někdo používá místo zúžené mezery mezeru pevnou – při stahování to moc nevadí, ale při rozpalování to nevypadá dobře.

[5] To je vidět zejména za zkratkami jako „tzv.“ či „sv.“, protože chybí kerning ve „v.“.

```
\def\ZuzZa {\expandafter\ifx\space\tmp \tmp\zuzeni \else\ifx~\tmp
  \tmp\zuzeni \else\expandafter\expandafter\expandafter\tmp\fi\fi}
{\catcode‘\@=11  \def\pr@m@s {\ifx’\next\let\nxt\pr@@@s
   \else\ifx^\next\let\nxt\pr@@@t\else\let\nxt\egroup\fi\fi\nxt}}
```

Někomu mohou nezúžené mezery za čárkou nebo za tečkou končící větu vyhovovat nebo alespoň nevadit, rozhodně by se ale mělo zužovat za tečkou, která není na konci věty – nezúžená mezera v takovém případě (zejména při rozpálení) vypadá rušivě. Za tečkou nekončící větu se obvykle používá příkaz `\␣`, byť standardně s jiným výsledkem – místo rozšířené mezery se použije základní. Chceme-li tento příkaz předefinovat, je třeba přepsat i definici „`~`“. Pro zúženou nerozdělitelnou mezeru používám příkaz `\~`. Zúžení je provedeno až za mezerou, abychom mohli testovat, zda poslední mezera byla zúžena (viz dále).

```
\let\SP=\   \def~{\nobreak\SP}
\def\ {\SP\zuzeni}
\def\~{\nobreak\ }
```

5.2. Pomlčka

V české typografii se obvykle používá pomlčka „`--`“ šířky 0,5 em, přičemž nemá být na začátku řádku (pokud neuvozuje přímou řeč). Zužování lze zadávat explicitně jako v předcházejícím případě („`\~--\␣`“) nebo lze psát „`~\=`“ pro:

```
\def\=~{\ZuzPred\hbox{--}\TestZa}
\def\ZuzPred {\leavevmode\ifdim\lastskip>0pt \zuzeni\fi}
```

Tato definice je vhodná i pro použití pomlčky bez mezer, `\hbox` umožní zabránit zlomu řádku za pomlčkou. Chceme-li zužování kolem pomlčky „sčítat“, lze testovat nenulovost `\lastskip` a za pomlčkou zúžit *před* mezerou (ne za ní).

Někdy se používá i americká pomlčka „`---`“ šířky 1 em, ta se doporučuje spíše v akcidenčních tiskovinách (letáky, vizitky, …), pro běžný text se považuje za příliš širokou: v amerických textech se používá bez mezer, takže slova jsou od sebe vzdálena 1 em, což je stejně jako „`--`“ se zúženými mezerami v základní velikosti (to je dokonce opticky oddělenější); při použití „`---`“ by se bez zužování došlo při největším „dobrém“ rozpálení až k 2 em.

5.3. Uvozovky

České dvojité uvozovky v `csr10` byly odvozeny z amerických: mají stejnou šířku boxu, kresba je posunuta směrem k textu, který obklopují, a do správné výšky.

Tím ale obsahují na vnější straně mezeru asi 0,07 em, kterou mít nemají.[6] Pro LaTeX jsou v `czech.sty` zavedeny i jednoduché uvozovky `\clq` a `\crq`, zvláště pravá ale obsahuje vpravo mezeru navíc. Upozorňuji, že tyto problémy při použití jiného fontu nemusejí vůbec vzniknout.

V následujících definicích jsou přebytečné mezery odstraněny, je doplněno zužování a kerning před tečkou a čárkou.

```
\def\UVkor  {\kern-0.07em }
\def\UVkern {\kern-0.08333em }
\def\allowhyphens {\nobreak\hskip0pt\relax}
\def\clq {\allowhyphens{,}}
\def\crq {{‘}}
\def\clqq {\leavevmode\hbox{}\UVkor\allowhyphens\char254 }
\def\crqq {\char255 \UVkor\hbox{}}
\def\cq  #1{\ZuzPred\clq #1\crq \TestZaUV}
\def\cqq #1{\ZuzPred\clqq#1\crqq\TestZaUV}
\def\TestZaUV {\afterassignment\ZuzZaUV\let\tmp= }
\def\ZuzZaUV {\ifx.\tmp\UVkern\else\ifx,\tmp\UVkern\fi\fi\ZuzZa}
```

Příkaz `\clq` je v `czech.sty` vyroben z apostrofu příkazem `\set@low@box` (v plainTeXu lze použít `\lower`), zde je pro jednoduchost zaveden jako čárka, která vypadá stejně (kromě `\tt`). Složené závorky v definicích jednoduchých uvozovek zabrání nepatřičným ligaturám – například „`!‘`“ dá „¡“, dvě čárky v cs--fontech dají otevírací dvojitou uvozovku. Příkaz `\allowhyphens` umožní dělení slova za levou uvozovkou; je před znakem uvozovky, aby zůstal zachován kerning za ním, může tam být, protože tento znak má nulové `\lccode`.

6. Pevná zúžená mezera

Podle [1] se jednotlivé údaje data oddělují pevnou mezerou, nikdy ne proměnnou.

Je-li datum vypsáno čísly, měla by to být nerozdělitelná mezera velikosti nejmenší zúžené (je za tečkou), tj. obvykle $0{,}16\overline{6}$ em. Je-li měsíc vypsán slovem, připadá mi toto pravidlo sporné: proč by jiné mezery měly být ve spojeních „15. kapitola“ (zúžená) a „15. ledna“ (pevná zúžená), proč by jiné mezery měly být ve spojení „roku 2002“ (základní) a „ledna 2002“ (pevná). Nejspíše by bylo vhodné tuto mezeru používat i v ustálených zkratkách (př. n. l., a. s., …).

V LaTeXu můžeme využít `\,` o velikosti právě $0{,}16\overline{6}$ em, která je zavedena takto (v plainTeXu je definována jen v matematickém módu):

[6] Některé varianty `czech.sty` v LaTeXu tuto mezeru v příkazech `\clqq` a `\crqq` odstraňují při zlomu řádku.

1) Za tečkou, čárkou, apostrofem, kolem pomlčky a na vnějších stranách uvozovek by měla být mezera tzv. „zúžená“ – doporučuje se o polovinu šířky tečky. Za ‚tečkami na konci věty‘ nejspíš není nutné ‚zužovat‘, určitě by se ale mělo zužovat za tečkami „uvnitř“, tj. za řadovými číslovkami a různými zkratkami – větnými, iniciálami, tituly aj. Podotýkám, že v některých národních typografiích je tomu opačně. Doporučuje se např. mezeru za tečkou na konci věty naopak rozšířit. Při psaní letopočtu, jako např. 1. 2. 2002, se používá pevná zúžená mezera. Ta se dá použít i u ustálených zkratek. Příklady: Skončila 2. světová válka. Narodil se r. 256 př. n. l. Pan A. Novák. Technostroj, a. s., závod Praha.

2) Za tečkou, čárkou, apostrofem, kolem pomlčky a na vnějších stranách uvozovek by měla být mezera tzv. „zúžená“ – doporučuje se o polovinu šířky tečky. Za ‚tečkami na konci věty‘ nejspíš není nutné ‚zužovat‘, určitě by se ale mělo zužovat za tečkami „uvnitř“, tj. za řadovými číslovkami a různými zkratkami – větnými, iniciálami, tituly aj. Podotýkám, že v některých národních typografiích je tomu opačně. Doporučuje se např. mezeru za tečkou na konci věty naopak rozšířit. Při psaní letopočtu, jako např. 1. 2. 2002, se používá pevná zúžená mezera. Ta se dá použít i u ustálených zkratek. Příklady: Skončila 2. světová válka. Narodil se r. 256 př. n. l. Pan A. Novák. Technostroj, a. s., závod Praha.

3) Za tečkou, čárkou, apostrofem, kolem pomlčky a na vnějších stranách uvozovek by měla být mezera tzv. „zúžená“ – doporučuje se o polovinu šířky tečky. Za ‚tečkami na konci věty‘ nejspíš není nutné ‚zužovat‘, určitě by se ale mělo zužovat za tečkami „uvnitř“, tj. za řadovými číslovkami a různými zkratkami – větnými, iniciálami, tituly aj. Podotýkám, že v některých národních typografiích je tomu opačně. Doporučuje se např. mezeru za tečkou na konci věty naopak rozšířit. Při psaní letopočtu, jako např. 1. 2. 2002, se používá pevná zúžená mezera. Ta se dá použít i u ustálených zkratek. Příklady: Skončila 2. světová válka. Narodil se r. 256 př. n. l. Pan A. Novák. Technostroj, a. s., závod Praha.

Ukázka 3. Zúžené mezery: 1) bez zužování (a s původními uvozovkami), 2) se zužováním kromě mezer za tečkami na konci věty a za čárkami, 3) se zužováním.

1) Jejda... To ne. Ach jo... Co to dělám... Nic.
Dny: pondělí, úterý, ... Měsíce: leden, únor, ..., prosinec.
Budu si to muset... rozmyslet. ... Už to mám!

2) Jejda… To ne. Ach jo… Co to dělám… Nic.
Dny: pondělí, úterý, … Měsíce: leden, únor, …, prosinec.
Budu si to muset… rozmyslet. … Už to mám!

3) Jejda… To ne. Ach jo… Co to dělám… Nic.
Dny: pondělí, úterý, … Měsíce: leden, únor, …, prosinec.
Budu si to muset… rozmyslet. … Už to mám!

4) Jejda… To ne. Ach jo… Co to dělám… Nic.
Dny: pondělí, úterý, … Měsíce: leden, únor, …, prosinec.
Budu si to muset… rozmyslet. … Už to mám!

Ukázka 4. Výpustky s mezerami: 1) 0 em, 2) $0{,}08\overline{3}$ em, 3) $0{,}11\overline{1}$ em, 4) $0{,}16\overline{6}$ em. Výpustky obsahují mezeru i za poslední tečkou.

```
\def\,{\ifmmode\mskip\thinmuskip\else\thinspace\fi}
```

7. Poloviční mezera

Je nepružná a nerozdělitelná, má být mezi tečkami výpustky (elipsy, „tří teček“) a mezi výpustkou a přiléhajícím znakem. Výpustka obvykle přiléhá k předcházejícímu slovu, výjimkou jsou případy, kdy je použita na začátku věty (přiléhá k následujícímu slovu) nebo v neúplném výčtu (za čárkou je oddělena mezerou). Tuto mezeru lze také použít mezi dvěma uvozovkami, aby nesplývaly.

Nejlepší by bylo mít výpustku ve fontu (nejlépe jako ligaturu „`...`“) s mezerami upravenými kerningy. Současní typografové připouštějí, aby byly hodně malé, podstatné je, aby mezera mezi tečkami a mezi krajní tečkou a přilehlým znakem vypadala stejně. V příkazu `\dots` se používá polovina základní velikosti základní mezery ($0{,}16\overline{6}$ em), která je v matematických výrazech za interpunkcí. Je to zároveň velikost pevné zúžené mezery, takže v tomto případě má charakter mezislovními mezery. Lepší je asi použít polovinu pevné (tj. nejmenší základní) mezery [2]. V `csr10` (ne v `cmr10`) je zaveden kerning mezi tečkami právě o této velikosti ($0{,}11\overline{1}$ em), takže lze psát „`...`“ a doplňovat pouze mezery kolem výpustky. Nabízí se také použití poloviny pevné zúžené mezery (nejmenší mezislovní mezery za tečkou). Použití poloviny základní velikosti základní mezery mezi *kresbami* teček odpovídá pro `csr10` použití boxů bez mezer.

Zautomatizovat doplňování mezer kolem výpustky předefinováním příkazu

`\dots` je komplikované, protože výpustka se může objevit v různých situacích a tento příkaz může být ve zdrojovém textu použit různým způsobem (`\dots`, `\dots{}`, `{\dots}`). Nejjednodušší by bylo přidat do boxu pro výpustku mezery po obou stranách, ty by ale byly rušivé na začátku odstavce a při zlomu řádku.

```
\def\.{\kern0.11111em }  \def\Dots {.\..\..\.}
\let\DOTS=\dots
\def\dots {\ifmmode\DOTS\else\ifvmode\else
  \ifdim\lastskip=0pt \.\fi\fi\Dots\fi}
\everypar {\hskip1sp\hskip-1sp\relax}
```

V matematickém módu je ponechána původní definice – jednak kvůli velikosti mezery, jednak proto, že někdy bývá příkaz `\dots` předefinován tak, aby umístil výpustku do vhodné výšky. Test `\ifvmode` má zabránit mezeře před výpustkou na začátku odstavce – to nefunguje, pokud je použit příkaz `\noindent`. V takovém (dosti výjimečném) případě lze použít `\Dots` nebo lze v dokumentu využít uvedený příkaz `\everypar` – pak by ale na začátku odstavce nefungovalo dobře makro pro psaní pomlčky se sčítáním zúžení, v LaTeXu navíc bývá tento příkaz v různých prostředích (včetně `document`!) využíván. Za výpustku se přidává poloviční mezera,[7] která – oproti původnímu příkazu `\dots` – zmizí při zlomu řádku. Slovo následující za výpustkou bez mezislovní mezery nebude rozděleno – to lze umožnit použitím `\allowhyphens`. Mohli bychom také v definici mezery místo `\kern` použít `\nobreak\hskip`, to ale pro změnu zabrání zlomu řádku v mezeře za výpustkou.

8. Mezera před interpunkcí

Před dvojtečkou, středníkem (pokud nejsou za zkratkovou tečkou), vykřičníkem a otazníkem má být úzká mezera. Její velikost má být 1 bod do 12bodového písma a 2 body pro písmo 12bodové až 24bodové. V [2] je uvedeno $0{,}08\overline{3}$ em.

Smyslem je opticky oddělit znak od slova, což v jiných případech (tečka, čárka, …) není nutné. Od tohoto pravidla se někdy upouští (zvláště pro středník, případně i pro dvojtečku). V ukázce uvádím i použití mezery $0{,}05\overline{5}$ em (nejmenší násobek osmnáctiny), která se zdá být dostatečná a asi lépe odpovídá tomu, že nejmenší základní mezera je menší než doporučovaných 0,25 em.

Doplňovat mezeru před dvojtečkou, středníkem, vykřičníkem a otazníkem ve zdrojovém textu není vhodné, protože některé fonty mohou mít tuto mezeru již ve znaku (tzv. nálitek, v `cs`-fontech není). Asi nejlepší cesta je přes aktivní znaky, jak ukazuje příklad pro dvojtečku:

[7] Doplňování mezery za výpustku by mohlo vadit před mezerou nebo před uvozovkami.

1) Jak se máš? Ani se mne neptej, není to ono! Mohlo to ovšem být i horší. Co Pavel? Nevím. A co Oto? Nic. Pozor! Uvádíme seznam: pondělí, úterý, atd. Měsíce: leden, únor, atd. Byl jsem tam; ničeho jsem si ale nevšiml. Bylo to ono; nebylo to ono.

2) Jak se máš? Ani se mne neptej, není to ono! Mohlo to ovšem být i horší. Co Pavel? Nevím. A co Oto? Nic. Pozor! Uvádíme seznam: pondělí, úterý, atd. Měsíce: leden, únor, atd. Byl jsem tam; ničeho jsem si ale nevšiml. Bylo to ono; nebylo to ono.

3) Jak se máš? Ani se mne neptej, není to ono! Mohlo to ovšem být i horší. Co Pavel? Nevím. A co Oto? Nic. Pozor! Uvádíme seznam: pondělí, úterý, atd. Měsíce: leden, únor, atd. Byl jsem tam; ničeho jsem si ale nevšiml. Bylo to ono; nebylo to ono.

Ukázka 5. Mezery před dvojtečkou, středníkem, vykřičníkem a otazníkem: 1) 0 em, 2) $0{,}05\overline{5}$ em, 3) $0{,}08\overline{3}$ em.

```
\def\SPint {\ifmmode\else\kern.05556em \fi}
\def\dvojtecka{:}  \catcode`\:=\active  \def:{\SPint\dvojtecka}
```

Test na matematický mód umožňuje použít dvojtečku v matematických výrazech v původním významu. Tato mezera by neměla být mezi znaky (například dvěma vykřičníky) ani za zkratkovou tečkou – to uvedená definice neřeší.

9. TEXnické poznámky

Nejlepší způsob zavedení mezer je úprava stávajících příkazů tak, aby se jejich význam zachoval, rozšířil či jinak upravil (`\clqq`, …, `\dots`, `\:`, `\,`, `\␣`).

Méně praktické je zavádění nových příkazů – jejich jména by obvykle musela být delší, abychom se vyhnuli kolizím s již zavedenými definicemi.

Další možnost je předefinovat některé stávající příkazy tak, aby měly jiný význam – pak je třeba si dávat pozor, abychom je nechtěli použít ve významu původním. Zde se využívaly příkazy pro akcenty, které jsou krátké a dobře zapamatovatelné: příkaz `\~` jako zúžená nerozdělitelná mezera, příkaz `\=` jako alternativní „dvě čárky“ (pomlčka), posloupnost `\.`, `\,`, `\:` pro mezery. V LaTeXu je problém s příkazy `\'`, `\‘`, `\=` a `\-`. Protože se v prostředí `tabbing` předefinují, jsou v některých situacích (`minipage`, `\parbox`, `\footnote`, …) obnoveny z pomocných příkazů – musíme je tedy předefinovat i tam. Například po předefinování `\=` použijeme:

```
\makeatletter \let\@acciii=\= \makeatother
```

Lze také využívat tzv. aktivní znaky, pak mohou vyvstat problémy v situacích, kdy bychom je chtěli použít v původním významu: tečku jako desetinnou tečku při zadávání hodnot, ve jménu souboru, při názorném psaní výpustky; čárku ve významu desetinné čárky, v matematických výrazech (typ Punct), jako oddělovač parametrů (např. v LaTeXu v prostředí `picture`); dvojtečku jako relaci; středník jako typ Punct; … Některé případy se dají ošetřit vhodným testem (viz část 8 pro dvojtečku), v jiných lze znaky „zneaktivnit“ předefinováním kategorie, obvykle na 12 (Other), v případě podtržítka na 8 (Subscript). Pokud například chceme použít jinak aktivní tečku v názvu souboru, můžeme předefinovat příkaz `\input`:

```
\let\INPUT=\input
\def\input {\catcode‘\.=12 \Input}
\def\Input #1 {\catcode‘\.=\active\INPUT #1 }
```

S dalším problémem jsme se už setkali u apostrofu – aktivní znaky se mohly vyskytnout v dřívějších definicích jako neaktivní. Například `\caption` v LaTeXu vysází dvojtečku bez interpunkční mezery.

V konstrukcích typu `verbatim` (výpisy zdrojového textu), je nutno vrátit původní význam příkazu `\␣` a rozšířit množinu speciálních znaků o zavedené aktivní znaky (znak „_“ už mezi nimi je, čárku a tečku zde neuvažujeme):

```
\let\DOSPECIALS=\dospecials
\def\dospecials {\let\ =\SP\DOSPECIALS\do\’\do\:\do\;\do\!\do\?}
```

Někdy je třeba zabránit expanzi příkazů obsahujících konstrukce s `\if...` – například při zápisu do souboru pro vytváření obsahu nebo seznamů. V LaTeXu to lze ošetřit definováním pomocí `\DeclareRobustCommand` (pro aktivní znak by se ale předefinoval i příkaz `\␣`) nebo použitím příkazu `\protect`. Je také možné se podobným situacím vyhýbat – nepoužívat takové příkazy v příslušných situacích nebo nepoužít definici s variantou (záleží na typu psaného dokumentu).

Úpravy mezer by měly být převedeny na celočíselné násobky základní mezery při imitaci psacího stroje. To lze vyřešit předefinováním příkazu `\tt` (v LaTeXu i `\ttfamily`). Přepnutí `\tt` je pak nutné vymezit do skupiny, aby se po jejím ukončení obnovily původní mezery.

```
\let\TT=\tt
\def\tt {\def\CapCor{0pt\relax}\def\SPpevna{~}\def\zuzeni{}%
  \def\dots{\DOTS}\def\SPint{}\def\UVkor{}\def\UVkern{}\TT}
```

Pokud nebylo napsáno jinak, byly v tomto textu použity pro mezery tyto hodnoty: 0,25 em pro pevnou mezeru, zúžení podle části 5, $0{,}11\overline{1}$ em pro poloviční mezeru, $0{,}05\overline{5}$ em pro mezeru před interpunkcí. Zužovalo se za tečkou nekončící větu, za apostrofem, na vnější straně uvozovek a kolem pomlčky (tam se zužování sčítalo), ne za čárkami a ostatními tečkami. Interpunkční mezera se nedávala před středník.

Odkazy

[1] *Základní pravidla sazby.* ON 88 2503, 1975.
[2] Kočička, P., Blažek, F., Mohelská, L.: *Praktická typografie.* Computer Press, Praha, 2000.
[3] Pop, P., Fléger, J., Pop, V.: *Sazba I. Ruční sazba.* Státní pedagogické nakladatelství, Praha, 1984.
[4] Wick, K.: *Sazba matematických vzorců čtyřřádkovým způsobem.* Nakladatelství ČSAV, Praha, 1961.

Summary: Text spaces in TEX

Together with increasing quality of print, available to normal users, the significance of following the typographical rules grows. These rules may seem to be purely aesthetical matter. However, most of them have practical meaning: they enable to reduce reader's effort needed for "decoding'' the written text. This article deals with *Czech* typographical rules of using spaces.

Josef Tkadlec, `tkadlec@fel.cvut.cz`

TUG 2002 Annual Meeting and Conference September 1–7, Trivandrum, India

Even though the official deadline has passed, there is still time to submit a proposal for a paper to be presented at the most exciting TeX event of 2002, the international conference of TEX Users Group is scheduled to be conducted

in India during September 1–7, 2002 at Park Center, Technopark, Trivandrum, Kerala.

The theme for TUG 2002 is 'Stand up and be proud of TEX!'. Show us why it is still the typesetting tool of choice, the range of material it can handle, and especially how it can coexist with the new world of XML. We want to hear about

- using TEX to typeset XML
- multilingual typesetting using Omega
- high-quality hyperdocuments using pdfTEX
- fonts for non-Latin languages
- new directions for METAFONT and METAPOST

Full conference details can be found at `http://www.tug.org.in/tug2002/`

Proposals for papers should be sent to `papers@tug2002.tug.org.in`

We can accept proposals for a further two weeks (until the end of March 2002), and will notify authors of the acceptance at the end of that period. Other dates are:

- 12 April 2002: preliminary program available
- May 2002: send first version of full paper
- July 2002: send final version of full paper
- 4th–7th September 2002: TUG conference in India

Email addresses for contact:

`tug2002@tug.org.in`	– General information
`papers@tug2002.tug.org.in`	– Submission of papers
`finance@tug2002.tug.org.in`	– Financial matters
`travel@tug2002.tug.org.in`	– Travel information
`media@tug2002.tug.org.in`	– Media contact

Radhakrishnan
For TUG 2002 Organising Committee

TUGboat 20(3), September 1999

TUGboat 20(4), December 1999

TUGboat 21(4), December 2000

Errata

Omylem šéfredaktora byly vynechány obsahy dvou čísel 20. ročníku TUGboatu a byly zveřejněny až obsahy 21. ročníku. Navíc došlo k další chybě ve Zpravodaji č. 1–3/2001. Nadpis na straně 151 má být TUGboat 21(2), June 1999. Redakce se za tyto chyby omlouvá.

Redakční poznámka

Zdeněk Wagner

Právě jste nalistovali poslední stránku prvního čísla Zpravodaje roku 2002. Toto číslo se od předchozích liší dvěma rysy.

Ten první je, doufám, neviditelný. Dobré je totiž to, čeho si čtenář nevšimne. V tomto čísle by to měl být výsledek práce redakční rady. Při zanášení korektur jsem si ještě více uvědomil, jak je spolupráce důležitá. Mnoha překlepů bych si sám nevšiml. A pokud ani čtenář žádné překlepy nenajde, bude to dobré vysvědčení za práci redakční rady.

Druhý rys je, bohužel, patrný na první pohled. Zpravodaj je tenčí než předchozí čísla. Snížení rozsahu však není prvotním záměrem. Výbor CSTUGu soudí, že dodržení pravidelnosti vydávání čtyř čísel ročně je důležitější než zachování rozsahu.

Zpravodaj chce být vaším časopisem. Chce upokojit jak experty, kteří již pronikly do nejhlubších tajů TEXu a souvisejících programů, tak ty čtenáře, kteří se teprve rozkoukávají a přemýšlejí, jak to či ono typografické pravidlo realizovat TEXovými prostředky. A chce též upozorňovat na netradiční možnosti využití TEXu a informovat o aktuálním dění v oblasti TEXu i typografie obecně.

Zpravodaj může být vaším časopisem pouze tehdy, najde-li se dostatek autorů, kteří se budou chtít o své znalosti a zkušenosti podělit. Redakce věří, že autorů je v CSTUGu dost a že bude brzy zavalena kvalitními články, takže se znovu podaří zvýšit rozsah na 64 stran.

Zdeněk Wagner
zpravodaj@cstug.cz

Zpravodaj Československého sdružení uživatelů TEXu
ISSN 1211-6661 (tištěná verze), ISSN 1213-8185 (online verze)

Vydalo:	Československé sdružení uživatelů TEXu vlastním nákladem jako interní publikaci
Obálka:	Antonín Strejc
Počet výtisků:	750
Uzávěrka:	14. března 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. 05-740233
Adresa:	CSTUG, c/o FI MU, Botanická 68a, 602 00 Brno
fax:	05–412 125 68
e-mail:	cstug@cstug.cz

Zřízené poštovní aliasy sdružení CSTUG:

`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 CSTUGu

`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 CSFAQ

`bookorders@cstug.cz`
objednávky tištěné TEXové literatury na dobírku

ftp server sdružení:
`ftp://ftp.cstug.cz/`

www server sdružení:
`http://www.cstug.cz/`

Uzávěrka příštího čísla: 30. května 2002

Podávání novinových zásilek povoleno Českou poštou, s. p. OZJM Ředitelství v Brně č. j. P/2–1183/97 ze dne 11. 3. 1997.