TEX METAFONT TEX METAFONT TEX METAFONT TEX METAFONT

# 3

# ZPRAVODAJ 94

ČESKOSLOVENSKÉHO SDRUŽENÍ UŽIVATELŮ TEXU

# OBSAH



*Z dopisu našeho kolegy Petra Škody* (otištěno bez vědomí autora)

---

... můžu jen doporučit `dviscrs` k použití na strojích typu XT notebooocích s procesorem V30 a jiném drobném strojstvu (velká verze `dviscr` záhadně zatuhne s odkazem na špatné přerušení koprocesoru, který tam pochopitelně není...)

Pokud někdo chce namítnout, že provozovat TeX na XT je zvrhlost, můžu oponovat, že mám nainstalovanou podmnožinu CSTeXu-94 na subnotebooku QUADERNO (20 MB HD, 16 MHz V30, 1 MB ram, MSDOS 5.0 v PROM) a chodí velmi pěkně. QUADERNO má vlastní videokartu DCGA 640 × 400 pixelů s 8 stupni šedi a zjistil jsem, že krásně chodí volba `dviscrs /oa19`. Pak to máte k nerozeznání od VGA (skoro, těch chybějících 80 vertikálních řádků je možno při porovnání odhalit, ale skoro si toho nevšimnete). Odezvy při prohlížení jsou asi tak 3–5 s při změně stránky (změna gray scale). Pro kompilaci je nutné použít TeX86 (pro XT).

Je to nádhera vidět i na takhle malém (14 × 20 × 3 cm) stroji „opravdové DTP“ (zkuste si na XT pustit Venturu, Xpress či Pagemaker :–)). Takže promiňte za agitaci na předpotopní konfiguraci :-). Někteří lidé považuji QUADERNO za digitální diktafon, nebo lepší organizer. Po shlédnutí TeXu a výstupu na skoro VGA displeji (bohužel není podsvícený) padají do mrákot :-)). Myslím, že i toto je jeden z důkazů, jak je TeX geniálně vymyšlen...

# Ze života CSTUGu

Petr Olšák

Určitě si mnozí z vás vzpomínají, že dostali pozvánku na kurzy TEXu v Jevíčku a že tyto pozvánky dorazily relativně velmi pozdě, takže na rozhodování nebylo moc času. Také to byl termín, který, jak se později ukázalo, kolidoval s nejdůležitější softwarovou událostí v naší republice, s veletrhem INVEX-94.

Možná také někteří z vás měli pocit, že si v té pozvánce děláme legraci. Chtít totiž za dvoudenní kurz 280 Kč, přičemž v této částce je zahrnuto také stravování a ubytování, to skoro zavání nějakou levárnou. Když to porovnáme s cenami kurzů na programy typu T602, Norton Commander apod., na nichž se přihlášený patrně naučí správně a v pravé chvíli zmáčknout klávesu Esc, nemůžeme si myslet nic jiného.

Nebyla to levárna. Cena ubytování a stravování totiž činila v domově mládeže při gymnáziu v Jevíčku za jeden den neskutečných 115 Kč. Například za suroviny na oběd si účtovali 12,70 Kč. Když uvážíme současné ceny masa (a skutečně jsme tam maso měli), nedovedeme příliš pochopit, jak je to možné.

Karel Horák dále rozhodl, že tuto akci nebudeme brát jako výdělečnou. Znamená to, že zvolil minimální „konferenční příplatek“ tak, aby to akorát pokrylo náklady na ubytování a dopravu lektorů a přípravu některých materiálů (xerox). CSTUG tedy finančně z této akce vyšel bez ztráty desítky, ale také bez jediné koruny navíc. Ani lektoři neviděli žádnou odměnu za svůj výkon. Dlužno podotknouti, že s touto představou jsme do toho my lektoři šli, takže toto berte jen jako konstatování a nikoli jako výčitku.

Přes počáteční problémy související se skutečností, že byla pozvánka rozeslána pozdě, se velmi rychle naplnily ubytovací kapacity v Jevíčku, takže jsme dokonce museli některé přihlášky odmítnout. Z ohlasu na přihlášky můžeme soudit, že taková akce je dosti žádaná a že bychom se mohli někdy pokusit o její zopakování. Možná zase na jiném místě a rozhodně s větší časovou rezervou mezi rozesláním pozvánek a vlastní akcí. Ideální by bylo, kdyby se taková akce předem ohlásila například na stránkách tohoto bulletinu. To by ovšem tento časopis musel *pravidelně* vycházet, což jak všichni vidíme, se zatím neděje. Bohužel je navíc

marné očekávat, že by se tato situace zlepšila, protože se asi nenajde dobrovolník, který by shromažďoval články do bulletinu a časopis pravidelně formátoval. Navíc příspěvků do časopisu je proklatě málo.

Ještě se vrátím k Jevíčku. Kurzy jsme pojali více méně teoreticky. Tj. všichni tři lektoři (Petr Sojka – LaTeX, Petr Olšák – TeX a Karel Horák – METAFONT) kladli důraz na vysvětlení softwarových souvislostí a nikoli postupu, jak se co mačká při práci pod jedním konkrétním systémem. Tím se asi tyto kurzy diametrálně lišily od kurzů typu T602, o kterých jsem se již zmínil. Myslím, že právě proto mohly přinést přihlášeným mnohem trvalejší hodnoty. Je pravda, že tři lidé, kteří se přihlásili za Národní knihovnu v Praze, to po první mé přednášce vzdali. Když zjistili, že jim nebudu vysvětlovat, kde najdou na klávesnici Esc, jali se z tohoto kurzu ustoupiti. Ukázalo se, že došlo k nedorozumění, protože neviděli text pozvánky, ale někdo je na ty kurzy poslal. Sami přiznali, že kdyby viděli text pozvánky, dospěli by k závěru, že to není určeno pro ně, ale pro lidi starající se spíš o údržbu systému a TeXu na jejich pracovišti. Obávám se ale, že kurzy typu „vlevo nahoře na klávesnici vidíte klapku s označení Esc, zmáčkněte ji na povel – teď“ nejsou v souvislosti s TeXem vůbec myslitelné. Pokud se mýlím, pokuste se mi to, prosím, nějakými jemnými prostředky naznačit.

Byl jsem tam na všech třech kurzech a z rozhovorů s ostatními přihlášenými jsem nabyl dojmu, že to mělo pozitivní ohlas. Dokonce jsem asi svou vlastní přednáškou dosti pomotal hlavu jedné v TeXu začínající kolegyni. Když se naše skupina vracela večer z restauračního zařízení, četla tato kolegyně na obchodě nad výlohou nápis techtil a nikoli textil.

Určitě by bylo lepší, kdyby napsal jakési hodnocení kurzů někdo z přímých účastníků, který ale nebyl zainteresován organizací. Mohli bychom se možná dozvědět více kritických připomínek, které by ukázaly, čeho se máme vyvarovat při pořádání takových kurzů příště.

Kromě kurzů v Jevíčku jsme se na podzim roku 1994 pustili do další akce, která souvisela s propagací TeXu. Jednalo se o prezentaci CSTUGu na výstavě neziskových organizací. Asi na začátku října jsme dostali nabídku od Výstavního výboru výstavy, že za 1 000 Kč můžeme na výstavě dostat stánek a vystavovat tam naše aktivity. Výstava se konala 1. až 3. 12. 1994. Rozhodli jsme s Karlem, že za tisícovku do toho jdeme. Nabídka nám připadala podobně levná a neskutečná, jako asi vám připadaly ceny našich kurzů. Ve svých představách jsem vytvořil několik variant možné prezentace. Ta minimální počítala s tím, že výstavní plo-

chu pojmu jako nástěnku, vyvěsím tam pár informací a obrázků a jdu od toho. Jiné alternativy počítaly s tím, že bychom tam vystavovali knihy a měli tam dokonce počítač. Na předběžné schůzce s výstavním výborem jsem proto objednal i zásuvku 220 V až do stánku. Rozeslal jsem podrobnou informaci o výstavě členům výboru CSTUGu a doufal, že touto cestou najdu potřebný počet dobrovolníků, kteří by byli ochotni se střídat a držet po celou dobu výstavy trvalou službu. Bohužel, ohlas byl minimální, a proto mi nezbývalo, než přistoupit na minimální program.

S Karlem jsme den před zahájením výstavy naaranžovali stánek. Jako podklad jsme použili výtisky METAFONTu v „proof“ módu některých písmen z Computer Modern rodiny fontů. Dále za podklad posloužily METAFONTové obrázky z Karlovy a mé dílny a ukázky ze skript kolegy Donta. Na tomto podkladě zářilo slunce s nápisem TEX a dále tam bylo několik jednoduchých nápisů. Na poličkách jsme měli rozloženy některé sborníky a časopisy. Dále jsme připravili letáky na rozdávání a to bylo vše.

Děkuji všem, kteří se nabídli, že budou držet službu u stánků. Při aranžování nám pomohl Miroslav Dont, první den výstavy se u stánku vystřídal Karel Horák se Zdeňkem Wagnerem, druhý den jsem to táhnul já, v poledne mě zastoupil pan Štefan Porubský a poslední den se vystřídali student Petr Macháček a můj kolega z FEL Martin Bílý. Po celou dobu mi pomáhala s přípravou a udržováním stánku na výstavě moje žena, které bych chtěl také touto cestou poděkovat. Nemá to se mnou v souvislosti s TEXem lehké.

Nakonec se ukázalo, že by při takto „pokrytých“ službách u stánku bylo možné přistoupit i k lepší než minimální variantě, například předvádět něco na počítači. Nejsem si ale zcela jist, zda by tato snaha byla korunována nějakým konkrétním výsledkem, počítatelným např. v počtu dalších lidí, kteří by měli aspoň trochu představu o tom, co je to TEX. Je pravda, že jsme na výstavě rozdali stovku letáků. Přitom jsme se s nimi nevnucovali, spíš naopak. Dávali jsme je až těm, kteří na základě vystavených materiálů projevili zájem o bližší seznámení s naší aktivitou. I to se dá považovat celkem za úspěch.

Z globálního hlediska to byla výstava různorodá a mě osobně celkem zaujala. Svou činnost tam prezentovaly různé nevládní a neziskové organizace. Pro ilustraci uvedu např. různé nadace na pomoc zdravotně postiženým, Armáda spásy, Svaz vojenské mládeže, Občanské sdružení Česko-slovenské mosty, Společnost přátel Mongolska, atd. Poděkování patří i sponzorům výstavy a hlavně výstavnímu výboru, který mi připa-

dal, že sestává z nadšenců podobným způsobem zanícených, jako jsme my do TEXu. Hlavním posláním výstavy měla být prezentace existence bohatého života v nekomerčním sektoru. Tato prezentace by mohla ovlivnit zákonodárné činitele při přijímání zákonů týkající se této problematiky.

V závěru svého povídání bych se chtěl zmínit o přípravách či spíše nepřípravách nového CSTEXu. Rozhodli jsme se, že by bylo záhodno do současného CSTEXu zařadit několik novinek. Především nové vzory dělení slov vygenerované strojově, dále LATEX 2ε a asi nové verze Mattesových ovladačů. Také by tam neměla chybět česká a slovenská podpora pro sazbu PostScriptovými fonty.

Celý problém příštích verzí CSTEXu ovšem závisí na dostatečném počtu dobrovolníků, kteří jsou ochotni investovat do toho svou práci. Vznik a současný život CSTEXu se dá, při troše zjednodušení, vyložit takto. Vždy existoval člověk, který potřeboval TEX např. na svém pracovišti uvést do nějakého provozně dobrého stavu. Když už do toho tu práci věnoval, přidal ještě trochu další práce a spolupráce s některými kolegy, aby výsledek byl prezentovatelný v rámci CSTUGu.

V tuto chvíli ale koncepce CSTEXu naráží na velmi problematickou bariéru. Znovu sice existuje člověk, nebo dokonce skupina lidí, kteří pracují na uvedení TEXu do pokud možno perfektního provozu. Tentokrát se jedná o lidi v Brně. Jediný problém je, že tam zatím provozují TEX, jehož kódování fontů a základní koncepce jsou zcela nekompatibilní se současným stavem CSTEXu. Od své koncepce nejsou ochotni ustoupit, protože je jejich postup více přijatelný na mezinárodním fóru. Také asi nebudou ochotni zabývat se a příliš precizovat DOSovskou instalaci (a už vůbec ne MS-Woknovou), protože hlavním předmětem jejich zájmu je UNIX, zejména Linux.

Nemalým problémem je též skutečnost, že v CSTEXu je zahrnut velký díl Mattesovy práce, ovšem balík emTEXu tam není úplný a v originální koncepci (tj. např. s originální strukturou adresářů). To je v rozporu s požadavky samotného Mattese.

Rád bych, aby se k dalšímu osudu CSTEXu v souvislosti s naznačenými skutečnostmi vyjádřila širší členská základna. Proto uvítám diskusi na toto téma na lednovém valném shromáždění. Já sám totiž v současné době necítím potřebu CSTEX dále vylepšovat a tím i udržovat krok se světovým vývojem. Můj konzervatismus pramení z toho, že pracuji s jednou pevně daným a neměnným plainem a nikoli s prudce se rozvíjejícím

LaTeXem. Rád bych proto přenechal aktivitu kolem CSTeXu Brňákům. To ale přinese problémy, související s celkovou a základní změnou koncepce, hlavně na úrovni kódování CSfontů. Přibudou potíže s kompatibilitou. Je to ta nejsprávnější cesta? Není to podraz pro všechny, kteří považují TeX za něco stálého a v tom vidí jeho velikou přednost? Pro vysvětlení: TeX skutečně *je* stálý a neměnný, zde si ale klademe otázku, zda budeme bazírovat na neměnnosti toho, co se týká předpony CS.

Mluvil jsem o CSTeXu také s panem Wagnerem. Říkal, že je ochoten implementovat do CSTeXu nový LaTeX 2ε bez závislosti na tom, jaké kódování se zvolí. Použije k tomu zejména balík maker pana Zlatušky s jeho laskavým svolením. K práci přistoupí až poté, co se na síti objeví nová záplata LaTeXu 2ε, která je avizována na konec roku 1994. Znamená to tedy, že stejně nejdůležitější část novinek v CSTeXu je možné provést až v lednu. Práci proto v tuto chvíli neodkládáme kvůli neshodám v nějakém kódování (ačkoli tyto neshody tu jsou), ale kvůli čekání na nejčerstvější verzi LaTeXu 2ε.

11. 12. 1994 *Petr Olšák*

**Na doplnění: Moje zážitky z výstavy**

Souhlasím s Petrem přinejmenším v tom, že naše účast na zmíněné výstavě byla zajímavá zkušenost, přesto že v poslední fázi příprav, kdy už šlo opravdu do tuhého a bylo třeba nejen přiložit ruku k dílu, ale také obětovat dost nedostatkového času, jsem začínal o účelnosti takovéto „prezentace“ oprávněně pochybovat. Když mě vystřídal kolega Wagner, ještě jsem zběžně prošel většinu stánků a v uvedeném pozorování jsem se jen utvrdil. Ale konec konců jsem zas jednou po dlouhé době strávil dopoledne nečinorodým lenošením, trochu si početl ve vystaveném TUGboatu a občas věnoval úsměv neopatrným návštěvníkům, kterým se nepodařilo včas našemu stánku vyhnout; z nich projevily hlubší zájem pouze dvě ženské bytosti, z nichž jednu nejvíce nadchla možnost dělat „takové krásné odmocniny“, jež dosud musela do šestsetdvojkou psaných kandidátských prací dopisovat ručně, a ta si už u mně vyzvedla i instalační diskety, zatímco druhá pro změnu trochu dostala mne: nejdřív mě nechala zasvěceně vychvalovat možnosti TeXu a METAFONTu, aby se nakonec zeptala, jestli bych její chudé nadaci neudělal tedy značku, kterou zatím kreslí ručně. Ve své stařecké vstřícnosti jsem asi po týdnu značku dle svěřené předlohy skutečně v METAFONTu vyrobil a na adresu nadace odeslal návrh hlavičkového papíru. Dodnes jsem však nedostal žádnou odpověď (asi tam vyřizuje korespondenci někdo takový jako já).

*Karel Horák*

# Pragmatický prístup k odstavcom

Philip Taylor

Vždy ma privádza do úžasu to, ako veľa užívateľov TEXu (vrátane niektorých celkom vynikajúcich) je oboznámených s najtajnejšími oblasťami TEXu, ale ak sú postavení pred úlohu, ktorá by mala byť najľahšou zo všetkých — presvedčiť daný odstavec, aby sa správne vysádzal bez vygenerovania správy `underfull \hbox` alebo zneužitia príkazu `\spaceskip` a jemu podobných príkazov — zdajú sa byť neschopní dosiahnuť niečo blízke vyhovujúcemu riešeniu. Vždy keď dostanem TEXovský dokument od takéhoto užívateľa a spustím ho, nájdem asi tucet výstražných hlásení, ktoré autor, zdá sa, celkom šťastne ignoroval, alebo ak je celý dokument vysádzaný so zväčšenými medzerami medzi slovami a bez rozdelovania slov, pýtam sa sám seba, prečo sa táto zdanlivo jednoduchá úloha zdá byť pre mnohých taká ťažká.

Čestne povedané, časť výčitiek súvisí s Knuthom. Zatiaľ čo Boson SlowCoach a jeho súperi sa zdajú byť šťastní, ak vytlačia odstavec s najdesivejšími medzerami medzi písmenami, alebo ak vytlačia riadok len s troma slovami a s obrovským medzislovným priestorom, Don sa rozhodol, že TEX nebude robiť nič také: buď bude odstavec vytlačený správne, alebo nebude vytlačený vôbec! A samozrejme väčšina z nás, ak nie všetci, súhlasíme s Donom; kvôli čomu inému by sme sa mali vyhnúť zázrakom WYSIWYGu Bosonu kvôli kabalistickej zložitosti jazyka TEX (pokiaľ samozrejme nie sme všetci intelektuálni masochisti, ako sa mi občas zdá). Ale časť výčitiek, aby sme ostali čestní, súvisí tiež s knihou *The TEXbook*; kým viac než primerane opisuje parametre, ktoré riadia TEXovskú sadzbu odstavcov, menej však informuje o metódach, pomocou ktorých môžeme získať vhodné hodnoty pre tieto parametre.

V tomto článku chcem ukázať to, čo nazývam „*pragmatický prístup k odstavcom*“, kým nejaký matematický génius nepríde so vzorcom alebo s algoritmom, ktorým môžu byť určené vhodné hodnoty týchto parametrov pre ľubovoľné kombinácie fontov, miery,[1]) odsadzovania, vzory

[1]) Používam pojem „miera“ v typografickom zmysle, majúc na mysli šírku tlačenej strany bez okrajov; pre viacstĺpcovú prácu to zodpovedá šírke jednotlivého stĺpca bez okrajov a medzier.

delení slov, atď., atď., atď., menej smrtiacim, ako keď budem pokračovať v riešení nezávideniahodnej úlohy tlačenia textu pre niektoré zdanlivo ľubovoľné kombinácie týchto premenných, a presvedčujúcim tých, pre ktorých tlačíme, že to *nepôjde*, kým to nebude spĺňať naše vlastné, do určitej miery bezchybné štandardy estetickej dokonalosti rovnako ako tie Donove a TEXovské.

Musíme začať uvažovaním tých parametrov, ktoré (a) najviac ovplyvňujú správnosť vysádzania daného odstavca a (b) môžu byť logicky uvažované pod patronátom sadzača na rozdiel od tých, ktoré ovplyvňujú sadzbu, ale sú striktne pod kontrolou dizajnéra. Nasledujúca tabuľka, hoci nie úplná, uvádza niektoré dôležité parametre, ktoré spadajú pod tieto dve záhlavia

| Vymedzené sadzačom | Určené dizajnérom |
|---|---|
| `\emergencystretch` | `\font` |
| `\pretolerance` | `\hsize` |
| `\tolerance` | `\patterns` |
| `\hbadness` | `\parindent` |
| `\hfuzz` | `\fontdimen` $n$ |

Samozrejme, že dizajnér sa bude pravdepodobne snažiť zadať horné ohraničenia na všetky vstupy v stĺpčeku sadzača, zatiaľ čo sadzačovi bude poradené, aby sa skutočne neplietol do vstupov v dizajnérovom stĺpci (ak on alebo ona chce byť vôbec niekedy znovu najatý!).

Prejdime k metóde, verím že bude jednoduchosť sama, hoci trochu zložitá na vysvetlenie:

1. Spracujte text s citlivými implicitnými hodnotami piatich „sadzačovych parametrov“. Vyhovujúce implicitné (`default`) hodnoty môžu byť:

```
\emergencystretch = 0 pt
  \pretolerance = 150
   \tolerance = 250
    \hbadness = 150
     \hfuzz = 0 pt
```

   Ak od TEXu nedostaneme žiadne chybové hlásenie, je všetko v poriadku a úloha je skončená.

   Ak nie, potom chybové hlásenia môžeme rozdeliť do dvoch skupín: tie, ktoré reprezentujú `overfull \hboxy` (preplnené rámčeky a teda

skutočné chyby) a tie, ktoré reprezentujú `underfull \hboxy` (nenaplnené rámčeky a sú to teda skôr upozornenia ako chyby). Z uvedených je potrebné sa zaoberať najskôr prvými.

2. Znovu spracujte text, ale teraz nastavte `\tolerance = 9999`; najjednoduchšie sa to dá urobiť vypustením príkazu pre `\tolerance` z preambuly a inicializovať ho priamo v príkazovom riadku, ako napr.:

   ```
   TeX „\tolerance = 9999 \input text\
   ```

   Znovu musíme chybové hlásenia rozdeliť na skutočné chyby a upozornenia, na rovnakom základe ako minule.

   Ak sa už nevyskytli žiadne skutočné chyby, prejdite na krok 7; ak tu sú ešte stále skutočné chyby, tak text a formát môžu byť logicky považované za patologicky zlé a budú vyžadovať ďalšie akcie.
3. Overte veľkosť `overfull \hboxov`; ak nie sú väčšie, ako horné ohraničenia, určené dizajnérom na `\hfuzz`, nastavte `\hfuzz` na maximálnu hodnotu preplnenia a prejdite na krok 7.
4. *Ešte stále* sa vyskytujú `overfull \hboxy`. Prezrite `log`-súbor, zistite riadok (riadky), v ktorých sa objavili a určte, či je problém v neadekvátnom delení (ten je väčšinou riešiteľný), alebo v širokej tabuľke, nerozdeliteľnom vzorci, a pod. Ak je problém spôsobený nesprávnym delením, prejdite na krok 6.
5. Problém je mimo rámca tohoto článku! Zvážte vytlačenie tabuľky, presahujúcej koniec riadku v menšom fonte, zmenšite `\tabskip` a pod. Spýtajte sa autora, či nerozdeliteľný vzorec skutočne *nemôže byť* rozdelený. Použite svoju vlastnú heuristiku na riešenie úlohy, a keď prekonáte tieto ťažkosti, pokračujte v tejto procedúre od kroku 7.
6. Problém je spôsobený neadekvátnym delením slov. Zistite, či je potrebné rozšíriť zoznam v `\hyphenation`, alebo je potrebné pridať explicitné delenia do slova, presahujúceho riadok. Pridanie do zoznamu delení bude vhodné, ak v slove nie je nič nezvyčajné, ale vyskytlo sa málo deliacich bodov alebo nesprávne deliace body; explicitné delenie slova by sa vyžadovalo, ak slovo obsahuje určitý znak brániaci deleniu, napr. písmeno s akcentom, alebo ak pred slovom nie je medzera. Rozšírte delenia a pokračujte krokom 2.
7. Úspech! Už nemáte žiadne skutočné chyby. Teraz zostáva už len optimalizovať dokument tak, aby konečná verzia reprezentovala „najlepšiu možnú sadzbu“, v nejakom vágnom zmysle.

   Všimnite si najhoršie prípady v `underfull \hboxoch`. Hodnotu `\tolerance` nastavte na túto hodnotu a `\hbadness` na hodnotu o 1 nižšiu a znovu spracujte text. Nemali by sa vyskytnúť žiadne skutočné

chyby, ani `underfull \hboxy`, ktorých nekvalita (`badness`) prekračuje parameter `\tolerance`.

Teraz prichádza prekvapujúca časť. Mohli by sme si opodstatnene myslieť, že sme určili spodnú hranicu parametra `\tolerance`, ale mnohé dokumenty dokazujú, že to nie je tak. Nastavte `\tolerance` na hodnotu o 1 nižšiu ako v predchádzajúcom kroku a spracujte dokument znova. Ešte vždy *môže* byť vysádzaný korektne! Zmysel tohoto je sotva patrný, ale môže sa stať zrozumiteľným pri predstave, že tento TEXovský koncept „ideálneho“ odstavca je taký, ktorý minimalizuje *celkovú* špatnosť (`badness`); TEX *nemá* záujem o minimalizáciu špatnosti v nejakom jednotlivom riadku. Teda teraz môžeme mať odstavec, v ktorom sú dva alebo viaceré riadky „zlé“ v nejakom zmysle, zatiaľ čo špatnosť najhoršieho riadku bola zmenšená. Celkove je odstavec „horší“, ale z estetického hľadiska sa môže javiť ako „lepší“ (odstavec, pozostávajúci výhradne z kratších riadkov môže vyzerať lepšie ako odstavec, v ktorom vytŕča jeden extrémne kratší riadok).

Ak je odstavec korektne vytlačený s hodnotou `\tolerance` o jedna nižšou, ako je zrejmá dolná hranica, môžeme to rovnako dobre zopakovať. Vždy, keď je odstavec vytlačený bez chýb, nastavte `\tolerance` na hodnotu o jedna menšiu, ako je ohlásená najhoršia špatnosť, a opakujte postup. Môže sa stať, že ďalšie zmenšenie v `\tolerance` nie je možné a objaví sa `overfull \hbox`; nastavte predchádzajúcu hodnotu a skončite. Globálna optimalizácia je teraz skončená — určili sme optimálnu hodnotu `\tolerance`. Ak je menšia alebo rovná ako hodnota, ktorú uviedol dizajnér ako hornú hranicu, naša práca je spravená; ak nie, musíme vyvolať `\emergencystretch` a potom, možno, uskutočniť lokálnu optimalizáciu.

8. Zmenšite `\tolerance` na hodnotu dolnej hranice určenej dizajnérom a nastavte `\emergencystretch` na malú kladnú veľkosť. Správanie parametra `\emergencystretch` a konkrétne jeho interakcia s ďalšími uvažovanými parametrami sú slabo pochopené a v skutočnosti sú predmetom výskumu v projekte LATEX3. Napriek tomu zdravé pravidlo, založené na skúsenostiach, je nastaviť ho na 1 `em` (založené na základnom textovom fonte); táto hodnota môže byť, hoci sa to môže zdať zvláštne, rovnako vyhovujúca pre obe — širokú aj úzku — miery.

   Nastavte `\hbadness` na hodnotu o jedna menšiu, ako je `\tolerance`, a spracujte text znovu. Ak boli hlásené `overfull` boxy, tak máme problém: *môžeme* zväčšiť hodnotu `\emergencystretch`, ale toto náhle vedie k množstvu nenaplnených rámčekov a odstrašujúcej estetike; za

týchto okolností je pravdepodobne lepšie uskutočniť lokálnu optimalizáciu, čo je vlastne ďalší krok.

Pre každý riadok, hlásený ako preplnený alebo nenaplnený, uvážme odpovedajúci odstavec a zistime, či ďalšie delenie slov môže umožniť TEXu zbaliť pár prebytočných medzier alebo niekoľko znakov v riadku; tiež zvážte, či jeden alebo viaceré parametre, uvedené v Dodatku A (napr. `\doublehyphendemerits`, `\exhyphenpenalty`) môže prinútiť TEX uspokojiť sa so sadzbou tohoto odstavca, ktorá by nebola perfektná. Ak to je nevyhnutné, zvážte zmenu jedného alebo viacerých týchto parametrov len počas trvania odstavca tak, že ho uzavriete medzi dvojicu `\begingroup`/`\endgroup` a zmeníte parametre hneď za `\begingroup`. Pamätajte si, že odstavec musíte ukončiť príkazom `\par` alebo prázdnym riadkom *pred* zátvorkou `\endgroup`, ak majú zmeny parametrov priniesť nejaký efekt.

Pokračujte v lokálnej optimalizácii bez úprav parametrov `\tolerance` alebo `\hbadness`, až kým ďalšie zlepšenie už nebude môcť byť dosiahnuté; ak ostávajú nenaplnené rámčeky, je potrebné postupovať taktne: prezrite chybný odstavec (odstavce) a pozvite autora poopraviť ich len v prípade, ak na to oprávňujú estetické dôvody.

Pamätajte, že parameter `\tolerance` bol zmenšený zo svojej „optimálnej“ hodnoty na dizajnérovu dolnú hranicu; ak autor odmieta chybný odstavec prepracovať, uzavrite odstavec medzi dvojicu `\begingroup`/`\endgroup` rovnako ako vyššie a len na trvanie tohoto odstavca parameter `\tolerance` nastavte na jeho „optimálnu“ hodnotu (a oznámte dizajnérovi, že to bolo nevyhnutné).

Teraz je lokálna optimalizácia skončená. Zmeňte preambulu, aby ste začlenili experimentálne určené hodnoty pre `\tolerance` a `\hbadness`. Vyššie sme nezdôvodnili, prečo je potrebné nastaviť `\hbadness` na hodnotu o jedna nižšiu ako `\tolerance`: je to jednoducho preto, že ak ho nastavíme na doporučenú hodnotu, je možné preveriť, že experimentálne určená hodnota `\tolerance` bola v skutočnosti nevyhnutná, zatiaľ čo ostatné varovania TEXu potláčame; ak TEX zaradí do správy `underfull \hbox` so špatnosťou rovnou `\tolerance`, bola spravená nejaká chyba. Samozrejme, pre použitie musí byť `\hbadness` nastavené rovnaké ako `\tolerance`, aby boli eliminované falošné varovné oznamy.

Tento postup sa môže zdať zložitý, ale v skutočnosti je veľmi priamočiary a je určite intuitívny, ako raz plne oceníme jemnosť opakovaných zmenšení `\tolerance`. Zahrnuté kroky si vyžadujú úplné minimum edi-

tovania, maximálne možno využiť toho, že správanie TEXu môže byť ovplyvnené parametrami príkazového riadku. Postup bol dôsledne použitý pri písaní tohoto[2]) článku (a mnohých predchádzajúcich) a je pravidelne dennodenne používaný na mojej fakulte, kde sa produktivita cení dokonca viac ako estetická dokonalosť.

*Philip Taylor*
The Computer Centre, RHBNC,
University of London, U.K.
<P.Taylor@Vax.Rhbncc.Ac.Uk>

**Dodatok A:**
**Prehľad parametrov súvisiacich s odstavcom**
**Celočíselné parametre**:
`\adjdemerits`
`\doublehyphendemerits`
`\exhyphenpenalty`
`\finalhyphendemerits`
`\hbadness`
`\hyphenpenalty`
`\linepenalty`
`\looseness`
`\pretolerance`
`\tolerance`
**Rozmerové parametre**:
`\emergencystretch`
`\hangindent`
`\hfuzz`
`\hsize`
`\parindent`
**Glejové parametre**:
`\leftskip`
`\parfillskip`
`\rightskip`
`\spaceskip`
`\xspaceskip`
**Rôzne parametre**:
`\fontdimen 2`

[2]) Pri písaní originálu tohoto článku. (Pozn. prekl.)

```
\fontdimen 3
\fontdimen 4
\fontdimen 7
\parshape
```

## Ghostscript versus ps2pk

Karel Horák

V poslední době se stále častěji potkáváme s PostScriptem, a to i na stránkách tohoto občasníku. Minulé číslo našeho zpravodaje se ho dotklo hned třemi příspěvky a mně už před časem napadlo porovnat (nemaje přístup na žádnou obyčejnou PostScriptovou tiskárnu), jak se se zobrazením PostScriptových písmen vyrovnají běžně dostupné emulátory PostScriptu.

V horní části prvního obrázku vidíte v prvém sloupci TimesRoman tištěný z `.pk` souborů získaných programem `ps2pk` 1.4 z `.pfb` souboru ze sady písem pro Corel! 3.0, a to postupně ve velikosti 24.88 pt, 10 pt a 5 pt. Druhý sloupec pro srovnání obsahuje stejné písmo z české verze Adobe Type Manageru zpracované stejným způsobem.

Spodní část obrázku obsahuje stejnou ukázku zpracovanou verzí Ghostscriptu 3.12.

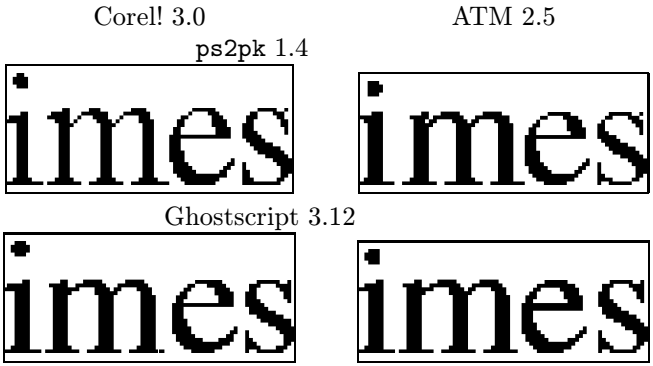

Je jasné, že čím větší rozlišení, tím kvalitnější je kresba bez ohledu na způsob emulace PostScriptu (toto číslo je tištěno z předloh v rozlišení 600 dpi). Proto nám lepší obrázek poskytnou čtyřnásobně zvětšené bitové mapy pětibodových písmenek, jak je vidíte v další tabulce.

TimesRoman TimesRoman
TimesRoman TimesRoman

A protože dokud jsem ještě používal třísetbodou laserovku, měl jsem k dispozici i komerční emulátor PostScriptu jménem UltraScript, vyzkoušel jsem tehdy i tuto možnost. Tentýž nápis je postupně vyrastrován starší verzí `ps2pk`, který jsem tehdy navíc provozoval bez koprocesoru (což opravdu dává trochu jiné výsledky, jak jsem později zjistil), verzí Ghostscriptu 2.5.2 a Ultrascriptem 3.01.

TimesRoman
TimesRoman
TimesRoman

Při dalším zvětšení už by snad mohly být patrny i rozdíly v kresbě patek (serifů), přičemž existují i drobné rozdíly v mezi různými `.pfb` soubory (zvláště stojí za povšimnutí umístění tečky nad i):



# Databáze bez databázového programu

Petr Olšák

Před nedávnem jsem se nabídl Karlu Horákovi, že budu spravovat databázi členů našeho sdružení. Po jisté „jednací době“ jsem od něj obdržel příslušnou databázi ve formátu DBase. Ukázalo se, že struktura této databáze není příliš dobře dořešena a že bude potřeba změnit a doplnit plno údajů.

Sám jsem nechtěl mít nic společného s žádným databázovým systémem. Po rozvaze, co budu od databáze očekávat, jsem dospěl k jednoznačnému závěru, že na doplňování a změny údajů vystačím s obyčejným editorem, rešerše a tisky mi udělá TEX a řazení podle české abecedy program `csr`. Řazení podle jednodušších položek (např. podle směrovacího čísla) zvládne funkce „Sort“ v editoru.

Databázi jsem transformoval z vnitřní datové reprezentace příslušného databázového programu do textového souboru, kde jeden řádek odpovídal jednomu „formuláři“ a jednotlivé položky lícovaly pěkně pod sebou. Položky jsem dále oddělil celkem snadno pomocí blokových operací editoru jistými oddělovači (smluvenými znaky), aby je mohl správně rozlišit TEX. K manipulaci s databází jsem použil editor Qedit.

Neznám vymoženosti databázových programů. Už v jednom svém článku jsem přiznal, že databázové systémy vůbec neumím. Proto nemohu srovnávat výhody těchto systémů s navrženou koncepcí. Přesto se pokusím o uvedení několika důvodů, proč mi práce v editoru připadá přijatelnější.

- Není potřeba učit se ovládat nové uživatelské prostředí.
- S databází je možné bez problémů pracovat v UNIXu i v DOSu.
- Textový soubor mohu snadno posílat např. ostatním členům výboru a nenutit je při čtení údajů používat nějaký databázový systém. Když jim navíc pošlu makra pro zpracování TEXem, je vše zcela bez problémů.
- Je-li výjimečně v jednom formuláři nějaká položka delší (například výjimečně dlouhá e-mailovská adresa), nemusí se předělávat struktura databáze. Stačí dát této položce prostor na úkor sousední položky,

protože při zpracování pro tisk mají stejně rozhodující roli oddělovací znaky a nikoli poloha údaje na řádku.

- Vyhledávání libovolného údaje probíhá snadno pomocí funkce „find“ v editoru.
- Prostým posunováním kurzoru ve vertikálním nebo horizontálním směru lze sledovat určitý údaj buď v kontextu s údaji stejného typu (např. platby za rok 1994) nebo v rámci jednoho řádku (jeden člen sdružení). Na obrazovce jsou vidět a snadno dosažitelné údaje v obou těchto směrech.
- Doplňování údajů (např. údaje o platbách členských příspěvků) lze provádět s minimálním počtem úhozů na klávesnici, protože pro poskočení kurzoru v řádku na požadované místo a pro doplnění stále se opakujících údajů (např. rok platby) lze udělat makro editoru.
- Kdykoli lze pomocí blokové operace třídění seřadit databázi tak, že je přehledně vidět sledovaný údaj. Například seřazení podle údaje „je člen výboru“ oddělí členy výboru od ostatních členů.

Přiznávám, že koncepce bude asi pro rozsáhlé databázové balíky údajů nedostačující. Napadají mě tyto nevýhody:

- Může se narazit na kapacitní možnosti editoru (např. pro Qedit je maximální délka řádku 512 znaků a maximální velikost souboru je omezena velikostí konvenční paměti DOSu).
- Může se jedinou neopatrnou operací v editoru poškodit struktura databáze tak, že náprava je těžko dosažitelná. Zvláště pravděpodobné to je v případě, kdy s databází pracují nepoučené osoby, které neumějí správně zacházet s editorem.
- Je otázka, zda může posloužit strukturovaný textový soubor jako vstup do databázového systému, ve kterém bude chtít pracovat můj následovník, který bude pokračovat v obhospodařování databáze členů našeho sdružení.

První nevýhoda nehrála v případě databáze členů CSTUGu velkou roli, protože celá databáze má objem pouze 130 kB a délka jednoho řádku je zhruba 370 znaků. Ani druhou nevýhodu nepovažuji za důležitou, protože zálohovat jakoukoli práci je nutnost. Kdo nezálohuje, může na to tvrdě doplatit, i když používá kvalitní a „blbuvzdorný“ databázový systém. Třetí nevýhoda se asi vyskytuje obecně při přechodu od jednoho databázového systému k jinému. Pokud ovšem databázový systém neumí import dobře strukturovaných textových souborů, pak to je dosti špatný databázový systém.

Nyní se zastavím u dalších „součástí“ této koncepce udržování databáze. Kromě editoru totiž pracuji ještě s dalšími dvěma softwarovými prostředky. Jedním z nich je program `csr` a druhým z nich je TEX. Oba programy jsou nezávislé na použitém systému (UNIX nebo DOS), a proto se takto koncipovaná databáze snadno udržuje na obou systémech zároveň.

Pokud jde o řazení podle abecedy, učinil jsem toto pozorování. Původní databáze, kterou jsem obdržel, měla snahu být seřazena podle jakési pseudoabecedy, kde všechny akcentované znaky byly později než Z. Že by existovalo něco jako dvojhláska CH, nebo dokonce víceprůchodový systém porovnávání, o tom nemohlo být ani řeči. Podobně zmršené databázové výstupy dostávám každoročně jako seznamy studentů ze studijního oddělení. Dospěl jsem tedy (přiznám, že bez důkladnějšího průzkumu) k závěru, že existuje plno databázových systémů, které českou normu abecedního řazení slov neuznávají. Řekl jsem si, že takto zrůdně pokračovat nebudu a napsal jsem si program, který dodržuje při řazení českou a slovenskou normu. Vlastnosti programu jsou popsány v článku *Program csr – abecední řazení podle normy.* Článek najdete v tomto čísle zpravodaje.

Nyní rozebereme TEXovská makra, která jsem pro účely obhospodařování databáze členů vytvořil. Především jsem definitivně oddělil databázi individuálních členů a kolektivních, protože údaje, které potřebujeme mít zaneseny, mají v obou případech dosti odlišnou strukturu. Snaha po sjednocení struktury obou databází, která při vzniku počítačové evidence členů existovala, vedla k naprosto neúčelným, nepochopeným a vesměs nevyplňovaným položkám (jako např. položka s názvem meziřádek1 a meziřádek2).

Udržuji tedy dva databázové soubory `individ.tb` a `kolekt.tb`, které obsahují vlastní databázové údaje v textové podobě. Ke každému z těchto souborů je napsán soubor maker `individ.tex` a `kolekt.tex`. TEX se spouští na soubor `*.tex`, v němž se pomocí příkazu `\input` přejde na odpovídající databázový soubor. Soubory `*.tex` obsahují tato makra:

- Makra na „scannování“, tj. načtení obsahu jednotlivých položek pomocí smluvených oddělovačů.
- Makra na vytváření rešeršních tisků (např. vytiskneme jen členy, kteří mají neprázdnou položku „e-mail“ a zároveň nezaplatili za rok 1993).
- Makra na formáty výstupu. V současné době jsou vytvořeny tři formáty: Tisk všech údajů, tisk přehledu plateb (samozřejmě se zabudo-

vaným sčítáním a vyplněním řádku „celkem“) a konečně tisk adres na samolepky.

- Za zmínku stojí ještě některá „pomocná“ makra. Například makro `\scandatum` přečte datum ve formátu `19940621` a vysází 21. 6. 1994. Makro `\anone` zase vytiskne slovo „ano“ nebo „ne“ v závislosti na hodnotě položky „logického typu“. Nebo makro `\pan` přečte rodné číslo a na základě toho vysází buď „pan“ nebo „paní“. Omlouvám se všem slečnám; rozdíl mezi „paní“ a „slečnou“ není z databázových údajů zjistitelný.

Před zpracováním databáze TEXem je potřeba odkomentovat na konci souboru `*.tex` jeden z příkazů `\printall`, `\printmoney` nebo `\printaddress`. Tímto způsobem se stanoví jeden ze tří formátů výstupu. Pokud bychom chtěli vytvořit rešeršní tisk, odkomentujeme navíc jeden z příkazů `\reserse`, nebo vytvoříme nový. Například

```
\reserse{\ifstudent\else\ifx\nezaplatil\moneyTHREE}{\fi\fi}
```

způsobí tisk jen těch členů, kteří nejsou studenty (`\ifstudent\else`) a zároveň nezaplatili za rok 1993 (`\ifx\nezaplatil\moneyTHREE`). Druhý parametr příkazu `\reserse` musí obsahovat odpovídající počet kontrolních sekvencí `\fi`. Vyjadřovací možnosti pomocí `\if` a `\fi` jsou lépe patrné při pohledu do souboru maker. Je totiž potřeba vědět, jak se jednotlivé položky jmenují.

Pokud by bylo potřeba přidat položku nebo změnit strukturu databáze, je nutné pozměnit tu část maker, která se věnuje načítání položek (tzv. scannovací algoritmy). Pokud by bylo potřeba upravit formát výstupu nebo přidat další typ výstupu, pak se provede jednoduchý zásah do té části maker, která se stará o vzhled výstupu (viz definice tisku). Tyto dvě části maker vzájemně spolupracují, ale je možné je upravovat myšlenkově odděleně.

Uvedená koncepce zpracování databázových údajů se dá použít například při organizování konferencí. Pokud by si to někdo chtěl vyzkoušet pro svou aplikaci, uvádím na závěr pro inspiraci výpis souboru maker `individ.tex`. Aby bylo lépe rozumět makrům z tohoto souboru, předchází ukázka jednoho řádku ze souboru `individ.tb`. Každý řádek v souboru `individ.tb` začíná příkazem `\p`. Za ním následují položky týkající se jednoho člena oddělené symbolem `:` nebo `|` podle smluvené struktury. Ukázka řádku týkající se jednoho člena se nevejde na jeden řádek v bulletinu. Proto je rozdělena do více řádků. Vězte ale, že ve skutečnosti se jedná o jeden řádek obsahující zhruba 370 znaků.

```
  \p Ferdinand    Mravenec   :010101/1234
      |Ondřeje Sekory 3  :Praha 13   :11300:     :mravenec@brouk.les.cz
      |Práce všeho druhu :RNDr.   :CSc. :     :                       :
      |     :        :     :     |FFTT: 1993
      |-:     :        : 150:19930709: 150:19940511|
```

Soubor maker `individ.tex` vypadá takto:

```
% Makro na zpracování databáze členů CSTUGu
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 9.11.1994                       Petr Olšák
% Plain

\newcount\globnum  \newcount\num
\newcount\sumTWO \newcount\sumTHREE \newcount\sumFOUR
\newif\ifstudent \newif\ifvybor \newif\ifrevizor
\newif\ifprihlas \newif\iftisk

% Scannovací algoritmy:
%%%%%%%%%%%%%%%%%%%%%%

\def\p#1|#2|#3|#4|#5|#6| {\advance\globnum by1
  \begingroup
  \scanname #1|\scanaddr #2|\scanoccup #3|\scanother #4|%
  \scanlogic #5|\scanmoney #6|%
  \expandafter\test\expandafter\print\endtest
  \endgroup}
\def\scanname #1 #2:#3|{%
  \def\jmeno{#1}%                   % jméno
  \def\prijmeni{#2\unskip}%         % příjmení
  \def\rc{#3\unskip}}%              % rodné číslo
\def\scanaddr #1:#2:#3:#4:#5|{%
  \def\uliceD{#1\unskip}%           % ulice \
  \def\mestoD{#2\unskip}%           % město  -- adresa pro korespondenci
  \def\pscD{#3\unskip}%             % PSČ   /
  \def\telD{#4\unskip}%             % telefon domů
  \def\email{#5\unskip}}%           % email
\def\scanoccup #1:#2:#3:#4:#5:#6|{%
  \def\zamestnani{#1\unskip}%       % zaměstnání
  \def\titul{#2\unskip}%            % titul před jménem
  \def\csc{#3\unskip}%              % titul za jménem
  \def\pscZ{#4\unskip}%             % PSČ (zam.)
  \def\uliceZ{#5\unskip}%           % ulice (zam.)
  \def\mestoZ{#6\unskip}}%          % město (zam.)
\def\scanother #1:#2:#3:#4|{%
  \def\telZ{#1\unskip}%             % telefon do zam.
  \def\fax{#2\unskip}%              % fax
```

```
  \def\telex{#3\unskip}%              % telex
  \def\poznamka{#4\unskip}}%          % poznámka
\def\scanlogic #1#2#3#4:#5|{%
  \def\jestudent{\anone #1}\if #1T\studenttrue\else\studentfalse\fi
  \def\jevybor{\anone #2}\if #2T\vybortrue\else\vyborfalse\fi
  \def\jeprihlas{\anone #3}\if #3T\prihlastrue\else\prihlasfalse\fi
  \def\jetiskni{\anone #4}\if #4T\tisktrue\else\tiskfalse\fi
  \def\rokclenstvi{#5\unskip}}%     % člen od roku
\def\scanmoney #1:#2:#3:#4:#5:#6:#7|{%
  \def\moneyONE{\anone #1}%         % zaplatil za r. 91 (ano/ne)
  \def\moneyTWO{#2}%                % příspěvek za r. 92
  \def\datumTWO{#3}%                % datum příspěvku 92
  \def\moneyTHREE{#4}%              % příspěvek za r. 93
  \def\datumTHREE{#5}%              % datum příspěvku 93
  \def\moneyFOUR{#6}%               % příspěvek za r. 94
  \def\datumFOUR{#7}}%              % datum příspěvku 94

\def\anone #1{\if#1Tano\else\if#1Fne\else nic\fi\fi}
\let\test=\relax  \let\endtest=\relax
\def\nezaplatil{ 0}
\def\emptyM{ }
\def\emptyP{ \unskip}
\def\reserse#1#2{\def\test{#1}\def\endtest{#2}}
\reserse{\relax}{\relax}
\let\finalaction=\relax

% Definice tisku
%%%%%%%%%%%%%%%

\def\hb{\hfil\break}
\def\scandatum#1{\if#1:\let\next=\eatcolon
   \else\let\next=\makedatum\fi\next#1}
\def\zerox{0}
\def\makedatum#1#2#3#4#5#6#7#8:{%
  \def\tempa{#7}\def\tempb{#5}%
  \ifx\tempa\zerox\def\tempa{}\fi\ifx\tempb\zerox\def\tempb{}\fi
  \tempa#8.\kern.2em \tempb#6.\kern.2em #1#2#3#4}
\def\eatcolon:{}

\def\printall {%
  \let\pt=\sevenrm
  \hoffset=-1.5cm \advance\hsize by3cm
  \voffset=-1.5cm \advance\vsize by3cm
  \def\beforeprint{\tisktrue}
  \parskip=\smallskipamount
  \baselineskip=10pt \lineskiplimit=-5pt
  \def\print {%
    \global\advance\num by1
```

```
    \vbox{\leftskip=\parindent
    \noindent\llap{\bf \the\globnum. }%
    {\bf \prijmeni\ \jmeno}, {\pt RČ:} \rc,
    {\pt Ulice:} \uliceD, {\pt Město:} \mestoD, {\pt PSČ:} \pscD,
    {\pt telD:} \telD.\hb
    {\pt Zam:} \zamestnani, {\pt titul:} \titul, {\pt za jménem:} \csc.
    {\pt email:} {\tt \email},
    {\pt telZ:} \telZ, {\pt fax:} \fax, {\pt telex:} \telex.\hb
    {\pt UliceZ:} \uliceZ, {\pt MěstoZ:} \mestoZ, {\pt pscZ:} \pscZ,
    {\pt Poznámka:} \poznamka.\hb
    {\pt Student:} \jestudent, {\pt Člen výboru:} \jevybor,
    {\pt Přihláška:} \jeprihlas, {\pt Tisknout:} \jetiskni.\hb
    {\pt Od roku:} \rokclenstvi, {\pt příspěvky: }{\pt 91:} \moneyONE,
    {\pt 92:}\moneyTWO\ (\expandafter\scandatum\datumTWO:),
    {\pt 93:}\moneyTHREE\ (\expandafter\scandatum\datumTHREE:),
    {\pt 94:}\moneyFOUR\ (\expandafter\scandatum\datumFOUR:).\par
    }\medskip}
}

\def\l#1#2{\hbox to#1{#2\hss}}
\def\r#1#2{\hbox to#1{\hss#2}}
\def\c#1#2{\hbox to#1{\hss#2\hss}}

\def\printmoney {%
  \advance\hsize by25pt
  \advance\vsize by2\baselineskip
  \def\beforeprint{\tisktrue}
  \headline{\vbox to0pt{\kern-1mm\hbox to\hsize{%
    \hskip3em\l{4cm}{Jméno a příjmení}%
    \l{3em}{členství}\r{3em}{1991}%
    \c{9em}{1992}\c{9em}{1993}\c{9em}{1994}\hss
    }\kern2pt\hrule\vss}}
  \def\print {%
    \global\advance\num by1
    \ifx\moneyTWO\emptyM\else\global\advance\sumTWO by\moneyTWO\fi
    \ifx\moneyTHREE\emptyM\else\global\advance\sumTHREE by\moneyTHREE\fi
    \ifx\moneyFOUR\emptyM\else\global\advance\sumFOUR by\moneyFOUR\fi
    \noindent
    \r{3em}{\the\num.\ \ }\l{4cm}{\prijmeni\ \jmeno}%
    \l{3em}{\rokclenstvi\ifstudent*\fi}%
    \r{3em}{\moneyONE}%
    \r{3em}{\moneyTWO}\r{6em}{\expandafter\scandatum\datumTWO:}%
    \r{3em}{\moneyTHREE}\r{6em}{\expandafter\scandatum\datumTHREE:}%
    \r{3em}{\moneyFOUR}\r{6em}{\expandafter\scandatum\datumFOUR:}%
    \par
    }
  \def\finalaction {%
    \medskip\hrule\medskip
```

```
    \noindent\hskip3em
    \l{4cm}{\bf Celkem}\hskip6em\r{3em}{\the\sumTWO}\hskip6em
    \r{3em}{\the\sumTHREE}\hskip6em\r{3em}{\the\sumFOUR}\par
    }
}

\def\pan{\expandafter\scanrc\rc}
\def\scanrc#1{\if#1\unskip pan\let\next=\relax
  \else\let\next=\scanrcnum\fi \next#1}
\def\scanrcnum#1#2#3#4\unskip{\ifnum#3>4 paní\else pan\fi}

\def\mkpsc{\expandafter\scanpsc\pscD}
\def\scanpsc#1{\if#1\unskip\let\next=\relax\else\let\next=\scanpscnum\fi
  \next#1}
\def\scanpscnum#1#2#3#4\unskip{#1#2#3\thinspace#4\unskip}

\def\printaddress{
  \baselineskip=1.1\baselineskip
  \nopagenumbers
  \hoffset=-1in  \hsize=210mm
  \voffset=-1in  \vsize=297mm \raggedbottom
  \lineskip=0pt
  \def\print {%
     \iftisk
     \global\advance\num by1
     \ifnum\num=16\par\global\num=1\fi
     \noindent\vbox to 37mm{\hsize=70mm \leftskip=18mm\vss
     \noindent \pan \hb
     \jmeno\ \expandafter\uppercase\expandafter{\prijmeni}\hb
     \hbox{}\uliceD\par
     \noindent\llap{\mkpsc\enspace}{\bf\mestoD}
     \vss}\hskip0pt plus2pt
     \fi}
}

% Vlastní zpracování (rešerše a deklarace formátu tisku)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Rešerše se zadávají pomocí příkazu \reserse se dvěma parametry.
% První parametr obsahuje vlastní podmínku a druhý parametr odpovídající
% počet \fi-ů.
% Je možno použít: \ifstudent \ifvybor \ifrevizor \ifprihlas \iftisk.
% Testy na položky typu \ifx, přičemž prázdá položka se testuje na
% \emptyP, prázdná položka v údaji o zaplacení na \emptyM a nulová
% hodnota v údaji o zaplacení má reprezentaci \nezaplatil.
% Následující příklad vybere všechny členy výboru, kteří mají neprázdnou
% položku email:
```

```
% \reserse{\ifvybor\ifx\email\emptyP\else}{\fi\fi}

% Jiný příklad vybere všechny členy, kteří nezaplatili za rok 93:

% \reserse{\ifx\nezaplatil\moneyTHREE}{\fi}

% Další případy:
% \reserse{\ifnum\num>30\endgroup\end\else}{\fi}
% \reserse{\ifstudent}{\fi}

% Formát tisku se vybere z následujících příkazů. Lze definovat
% analogicky další formáty (viz část definice tisku).

%\printall     % vytiskne všechny údaje v dosti zhuštěném tvaru.
\printmoney   % vytiskne prehled plateb.
%\printaddress % vytiskne adresy na lejblíky

\input individ.tb
\finalaction
\end
```

8. 12. 1994 *Petr Olšák*

# Program csr (Czech SoRt) – abecední řazení podle normy

Petr Olšák

Nejprve jsem si podle dokumentace pana Wagnera k programu `csindex` udělal soukromou verzi programu na abecední řazení. Pak mi kolega Josef Tkadlec půjčil normu ČSN 01 0181 (Abecední řazení). Jakmile jsem ji otevřel, zhrozil jsem se. Když jsem se vzpamatoval ze šoku, pokusil jsem se svůj program co nejvíce přiblížit této normě a hotový produkt věnovat veřejnosti. Následuje dokumentace k programu. Zajímavý je zřejmě odst. 5, kde na mnoha místech doslova cituji normu, upozorňuji na rozpory a uvádím případné návrhy, jak problémy řešit.

Program `csr` používá speciální čtyřprůchodový algoritmus abecedního řazení. První dva průchody probíhají nad jednotlivými slovy a případné

další dva průchody nad obsahem celé položky. Podrobněji viz odst. 3. Na rozdíl od české verze programu `makeindex` (`csindex`) program `csr` řadí pouze řádky souboru bez dalšího zpracování pro formátovací účely. Pracuje s externí tabulkou znaků, kterou lze konvertovat do libovolného kódování, a tím provádět řazení v libovolném kódování. Tuto tabulku lze také editorem upravit tak, aby řadicí algoritmus lépe vyhovoval představám uživatele. Dále, na rozdíl od programu `csindex`, program řeší řazení podle více slov (např. příjmení a jméno) a řazení podle více položek (heslo, doplněk, podheslo atd.).

Program se volá se třemi parametry:

```
csr input sorttab output
```

kde `input` je název vstupního souboru, `sorttab` je název souboru s tabulkou znaků, podle kterých se má řadit, a `output` je název výstupního souboru. Výstupní soubor může být shodný se vstupním.

Výstup se bude od vstupu lišit v pořadí řádků, jinak všechny texty zůstanou beze změny zachovány. V hlavičce vstupního souboru je možné specifikovat, podle kterých sloupců se má provádět řazení (viz odst. 2).

Program `csr` najdete např. v brněnském archívu `ftp.muni.cz` v adresáři `pub/tex/local/cstug/olsak/csr`. Kromě DOSovské (zkompilované) verze `csr.exe` tam je i zdrojový text v jazyce C, který umožňuje snadno instalovat program pro nejrůznější UNIXy. V UNIXu stačí provést toto:

```
dos2unix csr.c csr.c ... konverze na UNIXový formát,
cc csr.c -o csr      ... vytvoření běhové verze programu.
```

Protože program vyžaduje tři parametry, je užitečné vyrobit script, například s názvem `csort`, obsahující řádek:

```
csr $1 /uplna/specifikace/souboru/sorttab.iso $2
```

Pak se dá program volat z příkazové řádky bez specifikace tabulky řazení, což je určitě uživatelsky přijatelnější. Dále pomocí pracovních souborů v `/tmp` a příkazu `cat` je možné vylepšit script tak, aby se choval jako filtr. Samozřejmě je nutné přesměrovat `stdout` programu do `stderr` nebo do `/dev/null`. Program samotný se jako filtr nechová. Šikovní administrátoři mohou script dále rozšířit o možnosti přepínání mezi různými kódovacími tabulkami podle přepínačů napsaných v příkazovém řádku.

**1. Pevné a volné řádky vstupního souboru**

Ve vstupním souboru rozlišujeme řádky pevné a volné. Pevné řádky zůstanou na stejném místě i ve výstupním souboru, zatímco volné řádky

podléhají abecednímu řazení. Zhruba řečeno, pevné řádky mohou být na začátku a na konci souboru, zatímco „uprostřed“ jsou řádky volné.

Pevné řádky jsou všechny komentované řádky na začátku souboru. Komentovaný řádek je řádek, který má v první pozici komentářový znak, například procento (%) (viz odst. 4). Tyto řádky se bez změny pořadí přepíší do výstupního souboru. Počet takových řádků může být libovolný, tj. i nulový.

První nekomentovaný řádek souboru zahajuje pole volných řádků. Pole volných řádků končí buď koncem souboru, nebo komentovaným řádkem. Od tohoto komentovaného řádku až do konce souboru jsou všechny řádky (i nekomentované) pevné.

Poznamenejme, že komentářový znak má speciální funkci pouze v první pozici řádku. Kdekoli jinde se interpretuje jako obyčejný znak, tj. jako každý jiný a může například podléhat i abecednímu řazení.

**2. Parametry položek**

Pokud v poli pevných řádků (na začátku nebo na konci souboru) bezprostředně za komentářovým znakem následuje slovo `csr`, program přečte z tohoto řádku parametry položek. Například:

```
%csr: 30-50, 10-29, 51-61
```

znamená, že se bude nejprve řadit podle podřetězce (položky) v každém řádku od 30. do 50. sloupce (například tam je v každém řádku napsáno příjmení), pak podle položky vymezené 10. až 29. sloupcem (třeba tam je křestní jméno) a nakonec podle položky vymezené sloupci 51 až 61 (je tam třeba rodné číslo).

Symboly `-` (minus) a `,` (čárka) jsou ve formátu parametrů položek nutné. Za posledním parametrem se nesmí napsat znak `,` (čárka). Počet položek je maximálně 10.

Nenapíše-li se symbol `-` (minus), pak je daná položka ohraničena koncem prvního slova (tj. prvním výskytem mezery, tabulátoru nebo konce řádku). Například víme-li, že všechna příjmení v naší databázi se skládají jen z jednoho slova, pak stejný požadavek jako v předchozím příkladu můžeme napsat takto:

```
%csr: 30, 10-29, 51
```

Zde pouze v poli křestních jmen předpokládáme výskyt více slov, a proto jsme specifikovali i konec položky pro křestní jména.

Pozor na tabulátory! Program interpretuje tabulátor jako mezeru, tj. jediný znak. Program nemůže mít představu, jaká interpretace tabulátoru je v daném případě zvolena. Doporučuje se proto nejprve nahradit všechny tabulátory příslušným počtem mezer a pak teprve určovat parametry položek. Jinak to nebude řadit správně!

V parametrech položek může být bezprostředně za znakem - místo číslice znak `$`. Tím je řečeno, že položka končí až koncem řádku. Například

```
%csr: 1-$
```

znamená, že každý řádek od začátku do konce bude brán jako jediná položka. Toto nastavení je implicitní a použije se tehdy, pokud chybí v poli pevných řádků specifikace parametrů položek.

### 3. Algoritmus řazení

**3.1.** Algoritmus porovnává jednotlivé řádky nejprve podle položek. Položka je souvislý podřetězec řádku, který se obvykle specifikuje pomocí zadání počátečního a koncového sloupce (viz odst. 2). Položek může být více (maximálně deset). Pokud se shledá obsah první položky z hlediska řazení shodný, přejde se na porovnání druhé položky, pak třetí atd.

**3.2.** Algoritmus porovnává jednotlivé položky postupně po jednotlivých slovech (zleva doprava). Slovo je skupina znaků, která je od dalšího slova oddělena mezerami nebo tabulátory. Mezer a tabulátorů může být mezi slovy libovolné množství (počet mezer a tabulátorů mezi slovy na řazení nemá vliv). Pokud se např. první slovo shledá z hlediska řazení stejné, přejde se na porovnávání druhého slova v rámci položky atd.

**3.3.** Algoritmus porovnává jednotlivá slova nejvýše ve dvou průchodech. V každém průchodu prochází slovo postupně zleva doprava. V prvním průchodu rozlišuje pouze podle primární řadicí platnosti (tj. nerozlišuje např. ý od y ani od Y. Přesně je to specifikováno v tabulce `sorttab`). Pokud jsou z tohoto pohledu slova stejná, provede se druhý průchod, kde se rozlišují i znaky se sekundární řadicí platností (např. rozlišuje mezi ý a y, ale nerozlišuje mezi y a Y). V tabulce `sorttab` je (podle normy) uvedeno, že v prvním průchodu se nerozlišují písmena s diakritikou (až na čtyři výjimky), zatímco v druhém se diakritika rozlišuje. Ani v jednom průchodu se nerozlišuje mezi velkými a malými písmeny.

**3.4.** Pokud se všechna slova v jedné položce jeví podle odst. 3.3 jako stejná, provedou se ještě nejvýše dva další průchody, ovšem globálně nad celým obsahem položky, nikoli po jednotlivých slovech. Průchody probíhají zleva doprava a ignorují mezery a tabulátory mezi slovy. V prvním (celkově třetím) průchodu se rozlišují všechny znaky uvedené v tabulce `sorttab` s výjimkou tzv. skoro-prázdných znaků. Prakticky to znamená, že se rozlišují malá a velká písmena. V posledním průchodu se navíc rozlišují skoro-prázdné znaky, což prakticky znamená rozlišování interpunkčních znamének.

**3.5.** Mezera (nebo tabulátor) jako první znak za slovem je řazená abecedně dříve, než jakýkoli jiný znak. V souboru `sorttab` nelze pozici mezery ani tabulátoru specifikovat. Např. slovo gram bude abecedně dříve, než slovo gramáž. Jinak řečeno: jsou-li slova shodná až na to, že jedno je kratší, pak to kratší slovo bude řazeno dříve.

**3.6.** Pojem slovo (viz odst. 3.2) je zaveden pro libovolnou skupinu znaků s výjimkou mezery a tabulátoru. Pokud slovo obsahuje jenom prázdné znaky (tj. znaky neuvedené v řadicí tabulce `sorttab` a pro průchod 1 až 3 též skoro-prázdné znaky), interpretuje se jako tzv. prázdné slovo, které je abecedně dříve než jakékoli neprázdné slovo. Mezi různými prázdnými slovy řadicí algoritmus nerozlišuje (u skoro-prázdných znaků se rozlišuje ve čtvrtém průchodu). Příklad je uveden v odst. 5.11.

Příklady správného řazení slov (symbol „$\prec$“ označuje abecední pořadí):

| | | |
|---|---|---|
| plagiát | $\prec$ plachta | (protože g $\prec$ ch v prvním průchodu) |
| pláně | $\prec$ plánička | (protože ě = e $\prec$ i, v prvním průchodu) |
| plánička | $\prec$ Plánička | (protože p $\prec$ P ve třetím průchodu) |
| Plánička | $\prec$ plaňka | (protože i $\prec$ k v prvním průchodu) |
| plaňka | $\prec$ plankton | (protože a $\prec$ t v prvním průchodu) |
| platno | $\prec$ plátno | (protože a $\prec$ á v druhém průchodu) |
| plátno | $\prec$ platnost | (protože platno $\prec$ platnost v 1. průchodu) |
| plášť | $\prec$ plat | (protože š $\prec$ t v prvním průchodu) |
| plat | $\prec$ plát | (protože a $\prec$ á v druhém průchodu) |

Celkové řazení:

plagiát $\prec$ plachta $\prec$ pláně $\prec$ plánička $\prec$ Plánička $\prec$ plaňka $\prec$ plankton $\prec$ plášť $\prec$ plat $\prec$ plát $\prec$ platno $\prec$ plátno $\prec$ platnost

Jiný příklad, potvrzující skutečnost, že velká a malá písmena se rozlišují nikoli po jednotlivých slovech, ale v celé položce:

a ≺ abc ≺ ABC ≺ abc frézaře ≺ abc nástrojáře ≺ ABC nástrojáře

Vysvětlení:
ABC ≺ abc frézaře, protože druhé slovo je v položce „ABC“ prázdné a je tedy abecedně dříve, než slovo „frézaře“. K rozlišení dojde už v prvním průchodu nad jednotlivými slovy, kdy se ještě nerozlišují velká a malá písmena.

**4. Tabulka sorttab**

V této tabulce jsou uvedeny všechny znaky, podle kterých se provádí řazení (včetně speciální notace pro dvojhlásku CH). Do řádků jsou seskupeny znaky, které se nerozlišují v prvním průchodu, do skupin (oddělených mezerou) jsou zapsány znaky, které se nerozlišují v druhém průchodu, a všechny znaky se rozlišují v průchodu třetím (s výjimkou skoro-prázdných znaků). Skoro-prázdné znaky jsou uvedeny na zvláštních řádcích uvozených mezerou. Ty se rozlišují ve čtvrtém průchodu. V tabulce `sorttab` se také specifikuje komentářový znak.

Znaky, které nejsou vůbec uvedeny v tabulce `sorttab`, jsou tzv. prázdné znaky. Řadicí algoritmus je ignoruje.

Podrobnější informaci o formátu tabulky `sorttab` najde čtenář v dokumentaci k programu (`csr.doc`). Domnívám se, že tyto technické záležitosti nepatří na stránky tohoto časopisu.

Dodávaná tabulka `sorttab` má následující vlastnosti.

- Dvojhlásky ch, Ch a CH jsou mezi h a i.
- V prvním průchodu se rozlišují jen písmena neakcentovaná a písmena č, ř, š a ž.
- V prvním průchodu se též rozlišují číslice 0 až 9, které jsou uvedeny až za písmenem ž.
- V druhém průchodu se rozlišují všechna akcentovaná písmena, ale nerozlišuje se mezi velkými a malými písmeny.
- Ve třetím průchodu se rozlišuje mezi malými a velkými písmeny. Velké písmeno je později než malé.
- Skoro-prázdné znaky (čtvrtý průchod) jsou uvedeny až za číslicemi, a to v tomto pořadí: `.,;?!:"‘’|/()[]<>@&%=+*`

## 5. Odchylnosti chování programu od normy ČSN 01 0181 a návrhy na řešení

**5.1.** Norma člení tzv. prvky řazení nikoli na položky, ale na tzv. heslo (např. Novák Jan), doplněk hesla (např. senior), první podheslo (např. název díla), druhé podheslo (např. rok vydání) apod. Heslo, doplněk apod. se může skládat z více slov. Pro různé struktury dat se používá různé členění. Při použití programu můžeme provést např. toto ztotožnění:

heslo = 1. položka
doplněk = 2. položka
1. podheslo = 3. položka
2. podheslo = 4. položka
atd.

Ve výjimečných případech, kdy doplněk hesla má speciální způsob řazení nebo speciální grafickou úpravu, nastávají problémy.

**5.2.** V článku 7 normy je velmi nejasná formulace týkající se rozlišování slov, která se po prvním průchodu jeví jako stejná. Citujeme doslova (v hranatých závorkách je vysvětlení kontextu, které jsem doplnil):

„*Při jinak zcela stejném slově* [tj. po prvním průchodu] *se řadí nejprve písmena bez diakritických znamének a pak písmena s diakritickými znaménky neobsažená v abecedě podle čl. 5* [čl. 5 uvádí písmena bez diakritiky plus písmena č, ř, š, ž].

*Při diakritických znaménkách na různých místech v jinak zcela totožných slovech se řadí nejprve slovo s diakritickým znaménkem blíže konci slova. Vyskytnou-li se nad týmž písmenem slova různá diakritická znaménka, stanoví se pořadí podle čl. 8* [čl. 8 stanoví např. é $\prec$ ě, nebo ú $\prec$ ů $\prec$ ü]. *Vyskytnou-li se slova s diakritickým znaménkem nad týmž písmenem, avšak jedno slovo má ještě další diakritické znaménko, řadí se nejprve slovo s jedním diakritickým znaménkem.*“

Je vidět, že věda se dá dělat ze všeho. Méně průstřelnou formulaci by člověk těžko vymýšlel. Jedna interpretace této části normy velmi blízko odpovídá algoritmu implementovanému do programu (prochází se i v druhém průchodu zleva doprava). Například:

sténá $\prec$ stěna (é $\prec$ ě, podle algoritmu v programu `csr`)
sténá $\prec$ stěna (nejprve slovo s diakr. znaménkem blíže konci slova)
sténá $\prec$ stěna (dvě různá diakr. znaménka – čl. 8)

| | | |
|---|---|---|
| sténá | $\succ$ stěna | (nejprve slovo s jedním diakr. znaménkem) |
| nadívá | $\prec$ nádiva | (a $\prec$ á, podle algoritmu v programu `csr`) |
| nadívá | $\prec$ nádiva | (nejprve slovo s diakr. znaménkem blíže konci slova) |
| lalálá | $\prec$ lálalá | (a $\prec$ á, podle algoritmu v programu `csr`) |
| lalálá | ? lálalá | (není podle normy rozhodnutelné !?!) |

Důsledek: Přesné požadavky normy nejsou do programu implementovány, protože tyto požadavky jsou

a) nejasně formulovány a vedou ke sporným případům,

b) vedou k nerozhodnutelným případům,

c) vyhodnocování zleva doprava je vesměs každému člověku nejbližší.

**5.3.** Třetí průchod se podle normy dělá nikoli nad jednotlivými slovy, ale nad celými hesly. Tato vlastnost je do programu implementována. Algoritmus seřazování je ale opět trochu zvláštní. Cituji normu, čl. 9:

„*Velká a malá písmena mají stejnou řadicí platnost. Teprve tehdy, liší-li se v psaní velkých a malých písmen dvě HESLA (popřípadě další prvky řazení), shodná jak v písmenech standardizované abecedy, tak v diakritických znaménkách, řadí se nejprve heslo s malým písmenem. Různí-li se hesla velkým písmenem na různých místech, řadí se nejprve heslo s velkým písmenem blíže konci hesla. Při různém počtu velkých písmen se řadí nejprve heslo s menším počtem velkých písmen.*“

Můžeme učinit tyto důsledky:

| | | |
|---|---|---|
| aBCD | $\prec$ AbcD | (a $\prec$ A podle algoritmu v programu `csr`) |
| aBCD | $\succ$ AbcD | (tři velká písmena jsou více než dvě podle normy) |
| AbcD | ? aBcD | (není podle normy rozhodnutelné !?!) |

Požadavek normy na srovnávání podle počtu velkých písmen není do programu implementován z podobných důvodů jako v odst. 5.2.

**5.4.** V dokumentu [2] upozorňuje pan Wagner na potřebu čtvrtého průchodu pro rozlišení dvou variant velké alternativy písmene CH (tj. Ch a CH). Já jsem to v normě nenašel a považuji za postačující, rozlišit Ch $\prec$ CH už ve třetím průchodu.

Pan Wagner se podrobně rozepisuje o problému „když Ch není Ch“. Jeho příklad se slovem „mochnátý“ je jeden z nejpěknějších příkladů. Při použití programu `csr` je nutno vložit mezi c a h v takových případech prázdný znak, tj. např. `moc-hnátý` nebo `moc{}hnátý`. Tím se potlačí interpretace c a h jako dvojhlásky ch. Přitom řadicí algoritmus prázdné znaky ignoruje. Nebo jiný příklad. `C.~H.~CH.` bude při prázdných znacích `~` a `.` (vlnka a tečka) řazeno jako `CHCH`, ale první dvojice CH bude

interpretována jako Cé a Há a druhá dvojice jako CH (Cvičná Horská Chata).

**5.5.** Německé ostré ß se řadí jako `ss`.

Tato vlastnost (a mnoho podobných) se dá řešit zařazením preprocesoru a postprocesoru. Nechť např. znak `#` není uveden v tabulce `sorttab`, tj. jedná se o prázdný znak. Nahraďme preprocesorem výskyty ostrého ß skupinou tří znaků `ss#`. Pak provedeme řazení, které skutečně řadí `ss#` jako `ss`, protože `#` je prázdný znak. Nakonec postprocesorem nahradíme skupinu znaků `ss#` ostrým `s`. Takové pre- a postprocesory se např. v Unixu implementují jednoduše jako příkaz `sed` s určitými parametry.

V TEXu nemáme vůbec žádný problém, protože tam se ostré ß zapisuje jako `\ss{}`, tj. stačí mít znaky `\`, `{` a `}` jako prázdné.

Je třeba si uvědomit, že preprocesor prodlužuje texty položek, tj. může se stát, že položky už nebudou správně pod sebou zarovnané ve sloupcích. Pokud se jedná jen o pár pozic (mezer), většinou se nic neděje, protože program přeskakuje libovolné množství mezer na začátku položky.

**5.6.** Některé znaky (např. řecké $\gamma$) se řadí jako slovo (tj. „gamma“) a nikoli jako znak.

V takovém případě můžeme použít preprocesor podobně jako pro ostré ß. V TEXu opět nemáme problémy, protože tam se např. znak $\gamma$ píše jako `$\gamma$`. Stačí mít `$` a `\` jako prázdné znaky.

**5.7.** Problém s číslicemi a čísly komplikuje norma takto:

„*Číslice se řadí podle svého významu v hesle nebo v dalších prvcích řazení:*

*a) buď jako slovo (resp. sousloví), pokud jsou jako část delšího slovního výrazu chápány spíše jako slovní součást oficiálního nebo ustáleného označení určitého jevu, události, názvu apod.*

*b) anebo v číselném pořádku, jestliže výrazně označují pořadí něčeho, chronologii apod.*

*c) jestliže je výjimečně nutno přihlížet přednostně k číselné hodnotě číslic, řadí se ve vzestupném číselném pořadí až za abecedu, tj. za písmeno ž.*“

Příklady:

- zz $\prec$ z-2 $\prec$ ž $\prec$ 3 (podle c) – je implementováno v tabulce `sorttab`.
- 1 $\prec$ 10 $\prec$ 100 (podle b) – platí 001 $\prec$ 010 $\prec$ 100. Stačí nahradit vedoucí nuly před číslem nějakým znakem, který v dalším zpracování

nebudeme tisknout. Např. v řádku s nulou v tabulce `sorttab` stojí znak `_`. Pak využijeme toho, že: `__1` ≺ `_10` ≺ `100`.

- 10 pohádek ≺ 2. místo ≺ 25 let ≺ 1. výročí (podle a) – řadí se jako: deset pohádek ≺ druhé místo ≺ dvacet pět let ≺ první výročí.
- Václav III ≺ Václav IV (protože Václav 3 ≺ Václav 4).

Poslední dva příklady nelze snadno implementovat. Většinou jsou dosti výjimečné, a pokud je potřeba je řešit, pak jsou dvě možnosti:

1. Preprocesorem nahradíme všechny výskyty číslovek slovy (např. místo 10 bude `\deset`). Programovat takový preprocesor pro obecné použití by asi bylo dost složité.

2. Vytvoříme „stínovou položku“ s obsahem např. „deset pohádek“, podle které se řadí, ale která se nebude tisknout při dalším typografickém zpracování. Máme tedy vedle sebe položky „deset pohádek“ (podle které řadíme) a „10 pohádek“ (kterou tiskneme).

**5.8.** K interpunkčním znaménkům a značkám (tj.: `!`, `;` apod.) se obecně při řazení nepřihlíží. Teprve v případě, že jsou hesla stejná a liší se jen interpunkcí, pak se přistupuje k řazení podle interpunkce. Podle charakteru řazeného materiálu je možné některá interpunkční znaménka zařadit do tabulky `sorttab` například jako skoro-prázdné znaky. Pro orientaci citujme čl. 14 normy:

„*Jestliže se znaménka, značky, obrazce vyskytují samy o sobě ve funkci hesla, řadíme je až na konec abecedy, resp. až za číslice (pokud jsou řazeny podle čl. 11 c)* [viz bod c) v odst. 5.7]*, a to v tomto pořadí:*

. , ; ? ! : *všechny tvary uvozovek*

– | / \ ( ) [ ] < > { }

& £ § % ‰ $

= + × * # ∼ ≈ ≃

*další grafické značky*

*Stejného pořadí se používá při podřazování smíšených hesel se stejnou písemnou částí a se znaménkem (značkou, obrazcem) na stejném místě hesla.*

*Ve speciálních případech je nutno určit pořadí podle zásad příslušné vědní disciplíny.*“

Poznamenejme, že v dodávané tabulce `sorttab` nejsou uvedeny všechny tyto znaky. Pozměňte si tabulku `sorttab` tak, aby to nejlépe odpovídalo řazenému materiálu.

**5.9.** Existují skupiny slov, které se mají interpretovat jako jedno slovo. Např. mluvnické členy v románských jménech (la le) se berou dohromady se slovem:

lebeda ≺ le Bon ≺ lepra ≺ l'Herbier ≺ Li Pen ≺ liška

V těchto případech je možno slova spojovat např. nějakým prázdným znakem, třeba:

lebeda ≺ le~Bon ≺ lepra ≺ l'~Herbier ≺ Li~Pen ≺ liška

Postprocesor může tyto vlnky nahradit mezerami. V TEXu nemáme problémy.

**5.10.** U firemních názvů skládajících se ze dvou nebo více slov spojených symbolem `&` nebo `+` nebo `a` se nejprve řadí podle jednotlivých slov bez přihlédnutí ke spojce. Až v případě, že jsou obě (všechna) slova stejná, pak se ke spojce přihlíží.

Zeman + J. ≺ Zeman a Zeman ≺ Zeman & Zeman ≺ Zeman + Zeman ≺ Zeman & Ž.

Jak to implementovat? Spojku `a` nahradit např. znakem `@`. Deklarujeme-li v tabulce `sorttab` znaky `@`, `&` a `+` jako skoro-prázdné (v uvedeném pořadí), pak skutečně máme:

Zeman + J. ≺ Zeman @ Zeman ≺ Zeman & Zeman ≺ Zeman + Zeman ≺ Zeman & Ž.

Postprocesor pak nahradí znak `@` písmenem `a`.

**5.11.** Porovnávání prázdných slov (viz odst. 3.7). Příklad:

1. Novák
2. Novák – junior
3. Novák – senior
4. Novák Jan
5. Novák Jan – senior
6. Novák Karel

Pro pochopení řazení těchto údajů je vhodné interpretovat např. první řádek jako řádek s prvním slovem neprázdným (Novák) a s dalšími slovy prázdnými. Druhý řádek má první slovo neprázdné (Novák) a druhé prázdné (–), protože se skládá z prázdných znaků. Třetí slovo je neprázdné (junior). Nerovnost 1. ≺ 2. platí, protože prázdné třetí slovo prvního řádku ≺ neprázdné třetí slovo druhého řádku. Podobně např. 3. ≺ 4., protože prázdné druhé slovo třetího řádku ≺ neprázdné druhé slovo čtvrtého řádku. Výsledný efekt je přesně to, co od řazení očekáváme.

Máme-li různě oddělené doplňky hesel, případně z významového hlediska jsou stejná slova jinak interpretována, pak je potřeba být ve střehu. Citujme čl. 19 normy (včetně příkladu):

„*Jestliže se sejde v téže abecední sestavě ve funkci pořadatelů* [prvních slov hlavního hesla] *rodinné jméno se stejně psaným osobním jménem, řadí se nejprve osobní jméno.*“

Příklad: Adam – bakalář ≺ Adam de la Halle ≺ Adam, Alfred.

Program je správně seřadí, pokud prostřední prvek píšeme třeba ve tvaru `Adam ~ de la Halle`, tj. vnutíme za osobní jméno `Adam` prázdné slovo. V první položce už prázdné slovo prezentované znakem `~` máme.

**5.12.** Ještě větší problémy jsou s tzv. dvojitými jmény. Norma praví, že nejprve řadíme rodinná jména jednoduchá a potom teprve dvojitá. Dvojité rodinné jméno se nepovažuje za jedno slovo. Má platit:

Nová, Zdena ≺ Nová-Vlachová, Anna ≺ Nováček

Problém je možno řešit například tím, že pro rodinná jména (příjmení) sestavíme jinou položku než pro jména křestní. Takto je to ukázáno v příkladu v odst. 2. Dále pak musíme v rodinných jménech nahradit před zpracováním spojovník mezerou a po zpracování tam vrátit spojovník. To lze dělat automaticky pre- a postprocesorem. Jiným řešením je vložit do tabulky `sorttab` znak `-` dopředu před všechny abecední znaky. To ale předpokládá, že znak `-` nebude v řazeném souboru použit jiným způsobem. Pokud ale znak `-` je použit jako „klasická“ interpunkce, můžeme třeba nahradit všechny spojovníky dvojitých jmen znakem `^`, který bude uveden v čele tabulky `sorttab`. Postprocesor pak nahradí tento znak zpětně spojovníkem.

**5.13.** K mluvnickému členu na začátku hesla se nepřihlíží. Například „Der große Einbruch“ se řadí jako „große Einbruch“.

U německo-českého slovníku stačí psát členy do sloupce (položky), ke které se při řazení nepřihlíží, ale která se nakonec tiskne. Taková položka může být „vysunuta vlevo“ od hlavní položky. U samostatného německého slovníku je třeba postupovat podle příslušné normy DIN, kterou neznám. U již zpracovaných databází (například pro názvy děl v knižním katalogu) by asi bylo potřeba vytvořit preprocesor.

**5.14.** Nechtěl jsem nikoho odradit od entuziasmu algoritmizovat řazení podle české normy. Nicméně je vidět, že problematika je široká, a k různým typům datových vzorků je potřeba přistupovat individuálně. Norma

byla schválena v roce 1977, tj. před 17 lety. K výpočetní technice je tam napsáno tolik:

„*Norma je určena pro konvenční (ruční) řazení. Doporučuje se však, aby i při počítačovém řazení, zejména při sestavování programů a úpravě vstupních dat bylo k této normě přihlíženo (v mezích daných úrovní strojové techniky) za účelem snížení stupně diskompatibility mezi ručně a strojově řazenými abecedními sestavami na minimální, nevyhnutelnou míru.*“

Není vyloučeno, že v době, kdy píšu tyto řádky, existuje i norma pro strojové řazení. V každém případě se mezi strojovým a ručním řazením musíme snažit „snížit stupeň diskompatibility na minimální, nevyhnutelnou míru“. Ověřte si, jak ctí tuto skutečnost různé, často draze koupené, databázové systémy, které o sobě hlučně tvrdí, že obsahují národní „lokalizaci“. Mám zkušenosti, že mnohdy slavná lokalizace končí prachbídným přeložením manuálu do češtiny a počeštěním nabídek.

### 6. Omezení programu csr

**6.1.** Program dynamicky alokuje paměť pro obsah celého vstupního souboru. Tím je omezena velikost zpracování vstupního souboru. Například předkompilovaná verze programu `(csr.exe)` dodávaná v balíku umí pracovat jen s konvenční pamětí, tj. maximální rozměry souborů jsou cca 400 kB. Na druhé straně program byl testován na UNIXových pracovních stanicích (SUN) a bez problémů zpracoval i soubory velikosti několika desítek MB obsahující např. 250 000 řádků. Nepovažuji proto omezení dané velikostí alokovatelné paměti za příliš rozhodující. Určitě též bude možné kompilovat program pro DOS tak, aby pracoval s rozšířenou pamětí. Já to dělat nebudu, protože

a) pracuji s kompilátorem C pro DOS verze 1.0,
b) mám k dispozici zmíněné pracovní stanice.

**6.2.** Maximální délka jednoho řádku je stanovena parametrem MAXLINE ve zdrojovém textu programu na 1 000 znaků. V případě potřeby je možné uvedenou hodnotu zvětšit. Na nároky dynamické alokace paměti to nemá vliv.

**6.3.** Počet řazených řádků vstupního souboru je omezen prostorem dynamicky alokovatelné paměti a číslem 2 147 483 648.

**6.4.** V tabulce `sorttab` ani ve vstupním souboru nelze pracovat se znakem ASCII 0, protože tento znak má v programu speciální význam.

Také nelze pracovat se znaky ASCII 1 až 4, protože těmto kódům je přiřazena vnitřní reprezentace dvojhlásek ch, Ch, cH a CH. Se všemi ostatními znaky (ASCII 5 až 255) lze pracovat bez omezení. S výjimkou mezery, tabulátoru a znaku či znaků pro konec řádku v daném systému lze tedy pro libovolné jiné znaky v rozsahu 5 až 255 definovat jejich interpretaci v tabulce `sorttab` a pak podle nich provádět řazení.

**7. Literatura**


[1] ČSN 01 0181 ... Norma na abecední řazení slov.

[2] csindex.dvi ... český/slovenský index, dokumentace. Český překlad a implementace češtiny: Zdeněk Wagner. Dokumentace je součástí volně šířeného balíku CsIndex, který je např. přibalen do volně šířeného balíku CSTEX.


# Ještě jednou o programu WP2LaTeX


Václav Vopravil


Před časem jsem dostal příspěvky do lokálního sborníku v češtině, slovenštině, polštině a v lužické srbštině. Protože příspěvky byly napsány různými editory, hledal jsem možnost, jak příspěvky sjednotit. Při hledání po archívech jsem narazil na soubor[1]) `wp2latex.arj`, který obsahoval následující *pod*soubory:

```
leesme.bat
readme.bat
wp2latex.exe
wp2latex.msg
wp2latex.pas
wp2latex.sty
```

[1]) Uvedený soubor je zmiňován i v distribuci CD-ROM 4allTEX holandského TUGu.

```
wp2latex.tex
wp2ltx.tex
```

a který umožňoval konverzi formátu WordPerfektu do LaTeXu. Svoje zkušenosti s používáním těchto souborů popsal Zdeněk Wagner ve Zpravodaji 3/93.

Postup, který jsem použil já, vysvětlím na konverzi formátu Text602 do LaTeXu — analogicky je možné postupovat i pro editory WordStar, ChiWriter a pod., respektive pro jiné kódování. Předpokládejme dále, že zdrojový text `clanek.602` je napsán v kódování KeybCs2 editorem Text602. Příkazem

```
totex.bat clanek.602 clanek.wp clanek.tex
```

se spustí program:

```
pre602 /Z:5 %1 %2
wia %2 %3
cstocs -i0 %3 -o1 %3
```

který využívá převodníku `pre602.exe`, přefiltruje soubor `clanek.602` do formátu WordPerfektu na soubor `clanek.wp`, spustí program `wia.exe`, který převede soubor `clanek.wp` do LaTeXového `clanek.tex`. Nakonec program `cstocs` odstraní — pro lepší čitelnost — vnitřní kódování TeXu.

Zatímco programy `pre602.exe` a `cstocs.exe` jsou známy, program `wia.exe` vznikl úpravou programu `wp2latex.pas`. V tomto programu jsem, mimo několika nepodstatných změn, upravil tabulku odkazů znaků v proceduře

```
Ext_chr_init
```

na kódováni KeybCs2.

Uvedená úprava dobře převádí kódování češtiny (včetně horní tabulky znaků), zachovává typ písma (včetně dolních a horních indexů) a odstavce, ale nezachovává např. centrování řádků, nastavené tabulátory, matematický text aj., jinými slovy hodí se pouze na *hladké texty*. Vzhledem k tomu, že ani filtr `pre602.exe` není kvalitní, dávám plně za pravdu kolegu Wagnerovi, že nejlepší je psát text rovnou v TeXu. Zmiňované programy jsou volně k dispozici na adrese `vopravilv@pf.ujep.cz`.

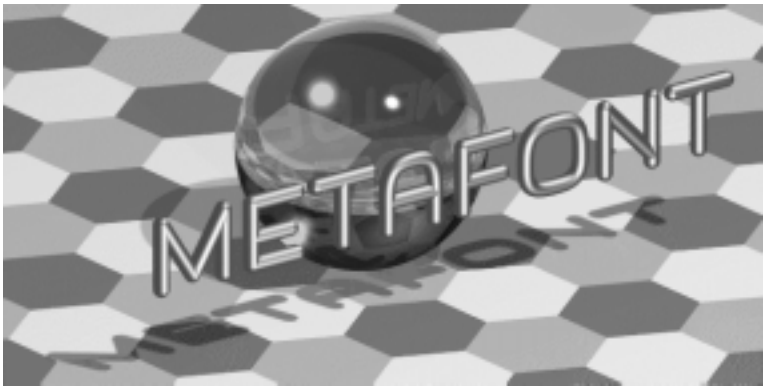
*Václav Vopravil*
`vopravilv@rek.ujep.cz`

# Z obsahu příštího čísla

Karel Horák: Jak se vypořádat s velkými METAFONTontovými obrázky při zrcadlovém tisku

---

Ze slibovaných ozvěn EuroTEXu zatím jen zlomek: TEXovský *Lion* na obálce vyvedený v barvách provázel uvedení nové verze 3.0 programu `bm2font` Friedhelma Sowy, která už umí i barevnou separaci. A následující obrázek byl původně rovněž barevný. Z METAFONTových dat jej pomocí dalšího softwaru připravili Bogusław Jackowski a Marek Ryćko, kteří podobnými obrázky zahájili svou skvělou přednášku. Stejně jako před pár lety v Praze byla hlasováním přítomných jejich přednáška oceněna jako nejlepší.




---


| | |
|---|---|
| Vydalo: | Československé sdružení uživatelů TEXu<br>vlastním nákladem jako interní publikaci |
| Obálka: | Bohumil Bednář |
| Počet výtisků: | 650 |
| Uzávěrka: | 19. prosince 1994 |
| Tisk a distribuce: | KOPP, Máchova 16, 370 01 České Budějovice |
| Adresa: | CSTUG, MÚ UK, Sokolovská 83, 186 00 Praha 8 |

Podávání novinových zásilek povoleno Oblastní správou pošt v Českých Budějovicích, j.zn. P 3.202/94 ze dne 19. července 1994