Abecední řazení a sestavování rejstříků výhradně pomocí makrojazyka TEXu

Petr Březina

Ve světě TEXu se pro abecední řazení rejstříků obvykle užívá externích programů, jako je např. makeindex. TEX ovšem poskytuje silný makrojazyk, který umožňuje vytvořit mechanismus schopný kvalitně provádět řazení podle abecedy. Hlavní výhoda využití takového mechanismu pro tvorbu rejstříků by spočívala v tom, že dokument by bylo možno zpracovat včetně rejstříku na každém počítači, kde je nainstalován TEX, nezávisle na dalším programovém vybavení (externí programy nemusí být všude k dispozici, mohou být závislé na použitém operačním systému atd.). Přesto se v praxi této možnosti většinou nevyužívá. V jedenácté kapitole knihy *Typografický systém TEX* [3] autor vyjmenovává několik důvodů: „Algoritmus by byl velmi těžkopádný, zatěžoval by paměť TEXu, zdržoval by formátování a v neposlední řadě úpravy algoritmu pro třídění podle národních abeced by byly velmi obtížné.“ Tato nedůvěra ve schopnosti TEXu se pro mě stala výzvou, abych se pokusil v makrojazyku TEXu napsat prakticky využitelný program na abecední řazení.

Při tvorbě makra sort, jak jsem nakonec nazval výsledek svého úsilí, jsem se inspiroval programem csr Petra Olšáka [cf. 2], odkud jsem převzal myšlenku čtyřprůchodového abecedního řazení a myšlenku třídicí tabulky. Posléze jsem k tomuto makru udělal soubor obslužných maker na tvorbu rejstříku; při návrhu vzhledu rejstříku jsem nalezl inspiraci ve článku *An indexing facility for TEX* [4]. Celý makrobalíček, který dostal název index, si čtenář může stáhnout z mých internetových stránek na adrese http://www.volny.cz/petr-brezina/.

Poté co jsem dokončil makrobalíček index, jeví se mi výše citované argumenty jako nepřesvědčivé. Pokusím se proto v krátkosti zhodnotit výhody a nevýhody svého řešení v konkurenci s externími programy. Jedna významná přednost makrobalíčku index zazněla již v prvním odstavci tohoto článku; na tomto místě ji už opakovat nebudu a přejdu k dalším vlastnostem. Řadicí algoritmus lze pomocí třídicí tabulky snadno přizpůsobit pro různé jazyky; navíc je celé makro sort pouze TEXovský kód, takže si ho šikovný TEXista může vždycky přizpůsobit konkrétním potřebám (např. přidat pátý průchod, bude-li to nějaký jazyk vyžadovat). Takovou flexibilitou se standardně užívaný program pro rejstříky makeindex pochlubit nemůže; doslechl jsem se však o programu xindy, který prý umožňuje definovat pravidla abecedního řazení pro různé jazyky. Paměťové nároky makra sort nejsou nijak zvlášť velké a jsou v podstatě nezávislé na délce seznamu hesel, která mají být seřazena, takže lze pomocí tohoto makra třídit

mnohatisícové rejstříky i ve starém dobrém emTEXu. Největší slabinou makra sort je jeho rychlost, která je mnohonásobně menší než rychlost programu makeindex; na druhé straně rychlost makra sort je srovnatelná s rychlostí programu csr. I když činnost makra je poměrně pomalá, nebylo by správné příliš zdůrazňovat, že zdržuje formátování, neboť rejstřík se obyčejně negeneruje při každém formátování v průběhu vzniku dokumentu, nýbrž až nakonec, když už je celý dokument napsán. Je také zapotřebí uvědomit si, že při použití makrobalíčku index lze vytvořit rejstřík během jediného zpracování dokumentu TEXem, zatímco při použití externího programu je pro vytvoření rejstříku nutno spustit TEX dvakrát a jedenkrát spustit onen externí program. Zdá se mi, že makrobalíček index nabízí propracovanější způsob zpracování čísel stran v rejstříkových heslech než program makeindex. Na tomto místě se sluší upozornit na jedno nebezpečí, které číhá na uživatele československé mutace programu makeindex, známé pod jménem csindex: všiml jsem si, že linuxová verze programu csindex řadí chybně plankton < plaňka, zatímco v DOSu jsou obě tato slova řazena správně; chybu jsem nahlásil autorům linuxové verze programu csindex.

Technické souvislosti aneb Trocha historie

Myslím si, že nebude nezajímavé nahlédnout do „strojovny“ makra sort a povědět si, jakým způsobem pracuje. Od prvního návrhu po konečný výsledek prodělalo toto makro zásadní proměnu. Při tvorbě makra jsem věnoval obzvláštní péči jeho rychlosti; nejsem však matematik ani informatik a nevím, zda se mi podařilo nalézt optimální řešení.

Vstupní soubor je nejprve podle primární řadicí platnosti rozdělen na šestnáct pomocných souborů. Jejich velikost je různá v závislosti na tom, kolik položek začíná jakým písmenem; kdyby např. všechny položky začínaly písmenem a, jeden pomocný soubor by obsahoval všechny položky, zatímco zbylých patnáct souborů by bylo prázdných, takže bychom si tím dělením vůbec nepomohli. V praxi však bývá rozdělení položek do pomocných souborů mnohem vyrovnanější. Jednotlivé pomocné soubory jsou setříděny samostatně a nakonec sloučeny do jednoho výstupního souboru.

První verze třídicího mechanismu, neúnosně pomalá (nevyužívala ani rozdělení položek do pomocných souborů), fungovala tak, že první položka ve vstupním souboru byla z tohoto souboru vyjmuta, uložena do paměti a postupně porovnávána s ostatními položkami v tomto souboru. Byla-li položka v paměti z hlediska třídění shledána větší než ta, s níž byla porovnávána, byly obě položky prohozeny. Po projití celého vstupního souboru byl obsah paměti zapsán na konec výstupního souboru setříděných položek. To se opakovalo, než byl vstupní soubor zcela vyprázdněn. Vstupní soubor čítající n položek bylo tedy třeba projít $(n-1)$krát a celkově bylo provedeno $(n^2-n)/2$ porovnání položek.

Druhá verze byla chytřejší. Položka načtená do paměti se neporovnávala s netříděnými položkami vstupního souboru, nýbrž se setříděnými položkami výstupního souboru. Tím se počet porovnání položek stal závislý nejen na celkovém počtu položek, ale i na jejich původním pořadí; tento počet porovnání se pohyboval v rozmezí od $n-1$ do $(n^2-n)/2$, přičemž druhá krajní hodnota odpovídala setřídění vstupního souboru s n již abecedně seřazenými položkami, kdežto první krajní hodnota odpovídala setřídění vstupního souboru s n pozpátku seřazenými položkami.

Protože operace zápisu a čtení do/ze souboru jsou obecně pomalé, začal jsem experimentovat s tříděním pouze v paměti TEXu. Tato „bufferová“ metoda třídění byla opravdu rychlejší než původní „souborová“ metoda, ale pouze do určitého počtu položek; za touto hranicí se rozdíl rychlosti obou metod zvětšoval ve prospěch metody „souborové“, neboť TEX zpracovává dlouhé řetězce pomalu. V TEXu je totiž jistý exponenciální vztah mezi délkou řetězce a dobou jeho zpracování.

Aby bylo dosaženo co možná nejvyšší rychlosti třídění, výsledná verze třídicího mechanismu je kombinací obou dvou třídicích metod („bufferové“ a „souborové“). Nejdříve se v paměti TEXu setřídí skupina dosud nesetříděných položek, jejichž počet je dán registrem `\bufsize`; potom se tato skupina porovná s již setříděným souborem. A tak pořád dokolečka. Registr `\bufsize` je přednastaven na hodnotu 16.

Použití makra sort

Abecední řazení pomocí makra sort je velice snadné; stačí TEXem zpracovat soubor obsahující následující příkazy:

```
\input sort
\sort {třídicí tabulka} {vstupní soubor} {výstupní soubor}
\bye
```

Uživatel může experimentovat s nastavením registru `\bufsize`, který jistým způsobem ovlivňuje rychlost třídění; minimální hodnota je 2, maximální hodnota je omezena dostupnou pamětí TEXu. Příkazu `\sort` je možno v jednom dokumentu užít vícekrát.

Vstupní soubor se skládá z libovolného počtu příkazů

`\abc{`*položka*`}{`*text položky*`}`

Třídění se provádí podle parametru *položka*. Je-li v seznamu více příkazů `\abc` se stejnou *položkou*, ve výstupním souboru zůstane jen jeden; parametry *text položky* budou sloučeny a odděleny oddělovačem `\sep`. Každá *položka* může být rozdělena příkazy `\endoffield` do několika polí, což umožňuje implementovat rejstříky s podpoložkami, podpodpoložkami, podpodpodpoložkami

atd. (tj. makeindexovský příkaz vykřičník). V *položce* se může vyskytnou příkaz `\endofentry`; v tom případě bude položka tříděna jen podle textu předcházejícího tomuto příkazu, ostatní text bude ignorován; to umožňuje implementovat makeindexovský příkaz zavináč. Všechny řídicí sekvence s výjimkou `\endofentry` a `\endoffield` jsou při abecedním řazení ignorovány.

Pravidla abecedního řazení jsou definována ve třídicí tabulce, která je uložena v samostatném souboru. Komentovaný příklad třídicí tabulky, vhodné pro češtinu, nalezneme v souboru `stab-cz.tex`. Třídicí tabulka obsahuje příkazy `\sorttab` a `\sorttabx`, které jednotlivým znakům přiřazují číslo abecedního pořádku. Příkaz `\sorttab` přiřazuje číslo pro každý průchod, zatímco příkaz `\sorttabx` přiřazuje číslo jen pro čtvrtý průchod. Řídicí sekvence s identifikátorem `p`*číslo průchodu*`:`*znak* se za pomoci příkazu `\chardef` stane číselnou konstantou. Znaky, jimž nebylo přiřazeno žádné číslo (tj. řídicí sekvence s identifikátorem `p`*číslo průchodu*`:`*znak* má význam příkazu `\relax`), jsou při třídění ignorovány. Všechny znaky uvedené společně jako parametr makra `\sorttab` mají pro první průchod přiděleno stejné číslo. Všechny znaky uvedené společně v jedné skupině, v níž jsou vzájemně oddělené čárkou, mají pro druhý průchod přiděleno stejné číslo; jednotlivé skupiny se od sebe oddělují středníkem. Pro třetí a čtvrtý průchod mají všechny znaky, které se vyskytují v parametru makra `\sorttab`, přidělena různá čísla. Například `\sorttab{e,E;é,É;ě,Ě}` znamená, že
v 1. průchodu e = E = é = É = ě = Ě,
ve 2. průchodu e = E < é = É < ě = Ě,
ve 3. průchodu e < E < é < É < ě < Ě,
ve 4. průchodu e < E < é < É < ě < Ě.
Znaky uvedené jako parametr makra `\sorttabx` mají přiděleno číslo jen pro čtvrtý průchod; vzájemně jsou odděleny čárkou. Za jeden znak je výjimečně možno pokládat skupiny písmen ch, Ch, CH, cH. Čárka a středník se do třídicí tabulky vkládají pomocí zástupných řídicích sekvencí `\comma` a `\semicolon`.

Cizí jazyky

Domnívám se, že při vytvoření vhodných třídicích tabulek bude čtyřprůchodový třídicí algoritmus makra sort dostatečný pro většinu evropských jazyků, včetně polytonické řečtiny. Čeština má tu zvláštnost, že při abecedním řazení pokládá skupinu znaků ch za jediné písmeno. V jiných jazycích se vyskytuje opačný jev: jeden znak odpovídá dvěma písmenům, např. ß v němčině nebo œ ve francouzštině. To se dá lehce vyřešit prostřednictvím speciálních znaků na úrovni zapisovacích a čtecích maker: Například zařídíme, aby se do vstupního souboru pro rejstřík přepsalo každé ostré es jako dvojice znaků `^^19^^19`, do třídicí tabulky dáme řádek `\sorttab{s,^^19,S}` a při sazbě výstupního souboru za-

jistíme, aby dvojice znaků `^^19^^19` vytiskla jedno ostré es: `\catcode‘^^19=13 \def^^19^^19{\ss}`. Pak budeme mít hezky setříděno: Masse < Maße < Mast. Jisté starosti nám udělá také to, že v různých jazycích se stavějí různě k otázce víceslovných položek. Má být (Petr a Jan) < (Petra a Jana), nebo (Petra a Jana) < (Petr a Jan)? Ignorování mezer (např. ve všech průchodech kromě posledního) se dá snadno zařídit ve třídicí tabulce. Nastavení třídění nad jednotlivými slovy versus nad celou položkou v různých průchodech by však vyžadovalo zásah do maker `\firstpass`, `\secondpass`, `\thirdpass` a `\fourthpass`. Tolik o perspektivě využití makra pro cizojazyčnou sazbu.

Tvorba rejstříku

Chceme-li vytvářet rejstřík, musíme pomocí příkazu `\input index` načíst soubor maker `index.tex`; spolu s tímto souborem bude automaticky načten i soubor `sort.tex`. Pak pomocí příkazu `\openindex` otevřeme soubor `\jobname.idx`, do nějž budou v průběhu zpracování dokumentu ukládány podklady pro vytvoření rejstříku. Na konci dokumentu tento soubor zavřeme pomocí příkazu `\closeindex`. Stránka obsahující poslední odkaz do rejstříku musí být uložena do souboru `\jobname.dvi` dříve, než přijde ke slovu příkaz `\closeindex`; proto před tímto příkazem napíšeme něco jako `\vfill\supereject`. Poté co je soubor `\jobname.idx` uzavřen, můžeme pomocí příkazu `\sortindex` rejstřík setřídit (výsledek této činnosti bude uložen do souboru `\jobname.ind`) a pak už nám jen zbývá použít příkaz `\printindex` k vytištění rejstříku. Nečekejme však, že makro `\printindex` vysází nadpis „Rejstřík“, zahájí vícesloupcovou sazbu, změní velikost písma atd.; to jsou věci, které si uživatel musí podle konkrétního typografického požadavku zařídit sám.

Tvorbu rejstříku lze ovlivnit prostřednictvím několika parametrů. Pokud ještě před načtením souboru `index.tex` přiřadíme řídicí sekvenci `\makeindex` pomocí `\chardef` hodnotu 0 (`\chardef\makeindex=0`), rejstřík vytvořen nebude; této vlastnosti s prospěchem využijeme při vzniku dokumentu nebo při jeho dolaďování, kdy nám o sestavení rejstříku vůbec nejde. Pokud řídicí sekvenci `\makeindex` přiřadíme hodnotu 1, do souboru `\jobname.idx` budou uloženy podklady pro tvorbu rejstříku, ale rejstřík nebude setříděn ani vytištěn, neboť příkazy `\sortindex` a `\printindex` neprovedou příslušnou činnost. K úplnému vytvoření rejstříku vede hodnota 2; tu však řídicí sekvenci `\makeindex` přiřazovat nemusíme, neboť se jedná o implicitní nastavení.

Pro základní vzhled rejstříku jsou připraveny tři styly, označené čísly 0, 1 a 2. Styl 2 je vhodný pro rejstřík s maximálně dvěma úrovněmi hesel (tj. heslo a podheslo). Výběr stylu se provádí ještě před načtením souboru `index.tex` tím, že pomocí `\chardef` přiřadíme řídicí sekvenci `\indexstyle` příslušné číslo.

Pokud tak neučiníme, použije se implicitně styl 0. Úpravou některých maker si uživatel může vytvořit svůj vlastní styl.

Jméno souboru s třídicí tabulkou, která se má použít pro abecední řazení rejstříku, se uvádí v těle definice makra `\stabfile`. Implicitně je definováno `\def\stabfile{stab-cz}`; to znamená, že se použije třídicí tabulka vhodná pro vytváření českých rejstříků.

V těle definice makra `\itabfile` se uvádí jméno souboru s inverzní tabulkou. Jak název napovídá, inverzní tabulka je, zhruba řečeno, opakem třídicí tabulky (v třídicí tabulce jsou znakům přiřazena čísla, zatímco v inverzní tabulce jsou číslům přiřazeny znaky) a využívá se jí při sazbě záhlaví skupiny hesel začínajících stejným písmenem. Zvolená inverzní tabulka by měla korespondovat se zvolenou třídicí tabulkou. Implicitně je definováno `\def\itabfile{itab-cz}`.

Text dokumentu je umístěn mezi příkazy `\openindex` a `\closeindex`. V textu můžeme pomocí různých příkazů vyznačit místa, na něž se má v rejstříku odkazovat.

Chceme-li do rejstříku zanést odkaz na jedno slovo, zapíšeme před to slovo příkaz `\idx` a do jeho parametru uvedeme heslo, pod nímž má být daný odkaz v rejstříku zařazen; například:

```
Zvláště \idx{Plutarch}Plutarch se stal mojí oblíbenou četbou.
```

Často ale nechceme, aby čísla stránek v rejstříku odkazovala jen na výskyt pouhého jednoho slova v textu, nýbrž na pasáž textu, kde se pojednává o pojmu vyjádřeném příslušným rejstříkovým heslem. V tom případě místo užití příkazu `\idx` uzavřeme mezi dvojici příkazů `\idxb` a `\idxe` danou pasáž textu; například:

```
Mrhání časem je velké zlo. Jiná, ještě horší zla provázejí vědy
a~umění. \idxb{přepych}Takový je přepych, zrozený jako ony ze
zahálky a~marnivosti lidí. Přepych jde zřídka bez věd a~umění
a~ony nikdy nejdou bez něj.\idxe{přepych}
```

Někdy chceme v rejstříku zdůraznit některé číslo stránky (např. tučným písmem), abychom čtenáře upozornili, že příslušná pasáž či příslušný výskyt daného slova je důležitější než ostatní odkazy. Pro vytvoření rejstříkového odkazu se zvýrazněným číslem stránky máme k dispozici následující příkazy:

`\idxit` — odkaz na jedno slovo, kurzivně;
`\idxbit` — začátek odkazu na pasáž textu, kurzivně;
`\idxeit` — konec odkazu na pasáž textu, kurzivně;
`\idxbf` — odkaz na jedno slovo, tučně;
`\idxbbf` — začátek odkazu na pasáž textu, tučně;
`\idxebf` — konec odkazu na pasáž textu, tučně;
`\idxul` — odkaz na jedno slovo, podtrženě;
`\idxbul` — začátek odkazu na pasáž textu, podtrženě;
`\idxeul` — konec odkazu na pasáž textu, podtrženě.

Odkaz na jiné rejstříkové heslo („viz“, „viz též“) se do zdrojového textu zapisuje pomocí příkazu `\idxsee`; například:

```
\idxsee{Istanbul}{Konstantinopol}
```

Pro oddělení hesel, podhesel, podpodhesel atd. se v parametru výše uvedených příkazů užívá vykřičníku; například:

```
\idx{jazyk!indoevropské jazyky!slovanské jazyky!čeština}
```

Pokud si přejeme řadit některé heslo jinak, než by odpovídalo jeho psané podobě, můžeme použít zavináč k oddělení podoby hesla, která se má vytisknout, od podoby, podle níž se má řadit; například:

```
\idx{Ludvík 08@Ludvík~VIII.}
\idx{Ludvík 09@Ludvík~IX.}
\idx{Ludvík 10@Ludvík~X.}
```

V jednom parametru příkazu pro vytvoření rejstříkového odkazu se může vyskytnout zavináč i vykřičník; například:

```
\idx{král!Ludvík 14@král!Ludvík~XIV.}
```

Všechny výše uvedené příkazy pro tvorbu rejstříkových odkazů, vyskytnou-li se ve vertikálním módu, způsobí přechod do odstavcového módu. To je důležité pro to, aby do rejstříku bylo zaneseno správné číslo strany, když se těchto příkazů užije na samém začátku odstavce. Potřebujeme-li výjimečně potlačit přechod do odstavcového módu, můžeme využít například následujícího triku: `\hbox{\idx{`*heslo*`}}`.

Pro zpracování čísel stran platí tyto čtyři zásady:

1. Na uživatelem zadaný rozsah stran (např. pomocí dvojice příkazů `\idxb` a `\idxe`) se pohlíží, jako by do rejstříku byly zaneseny jednotlivé strany z tohoto rozsahu.

2. Každé číslo strany se v rejstříku vyskytne pouze jednou; způsoby zvýraznění jednotlivých čísel se kumulují.

3. Posloupnost dvou či více těsně za sebou jdoucích čísel, která jsou popřípadě tímtéž způsobem zvýrazněna, se převádí na rozsah (např. posloupnost čísel 1, 2, 3, 4, 5, 6, 7, *8*, *9* bude zpracována takto: 1–3, 4–6, 7, *8*–*9*).

4. Záporná čísla stran (tj. strany předmluvy), jejichž pořadí musí být sestupné, se zpracovávají stejně jako kladná čísla s tím rozdílem, že budou vytištěna římskými číslicemi (malými písmeny).

Pokud v jednom rejstříku zvýrazňujeme čísla odkazovaných stran kurzivně i tučně, musíme definovat řídicí sekvenci `\bi` jako přepínač do tučné kurzivy pro případ, že by některé číslo mělo být zvýrazněno oběma způsoby současně.

Pro sazbu rozsahové pomlčky můžeme využít makra `\až`, popisovaného v článku *Hrstka tipů pro TEXovskou sazbu* [1]. V tom případě musíme náležitým způsobem předefinovat příkaz `\spandash`.

Při zápisu do souboru `\jobname.idx` je zapotřebí změnit kategorii aktivních znaků na neaktivní, tak aby neexpandovaly nežádoucím způsobem; to může uživatel zařídit v definici makra `\idxdefs`. Podobnou úlohu při načítání třídicí a inverzní tabulky plní makro `\tabdefs`; zde je však nutné definovat návrat k původnímu nastavení prostřednictvím makra `\canceltabdefs`.

Bibliografie

[1] Březina, Petr: Hrstka tipů pro TEXovskou sazbu. Zpravodaj Československého sdružení uživatelů TEXu, 16/2006, 1, str. 10—25.

[2] Olšák, Petr: Program csr (Czech SoRt) — abecední řazení podle normy. Zpravodaj Československého sdružení uživatelů TEXu, 94, 3, str. 126—139.

[3] Olšák, Petr: Typografický systém TEX. 2. vyd., Brno, Konvoj, 2000.

[4] Winograd, Terry — Paxton, Bill: An indexing facility for TEX. TUGboat, 1, 1, Appendix A, str. 1—12.

Summary: Alphabetical sorting and creation of indexes exclusively by means of TEX language

The article presents the macro package "index" for creation of indexes. An important part of this package is the TEX macro "sort" for alphabetical sorting. It uses a four-pass sorting algorithme which can be accommodated to different languages through a sorting table. Its memory requirements are independent of the length of the list of entries to be sorted. The treatment of page numbers in index entries is sophisticated. The macro package "index" is available including French documentation on author's home page http://www.volny.cz/petr-brezina/.