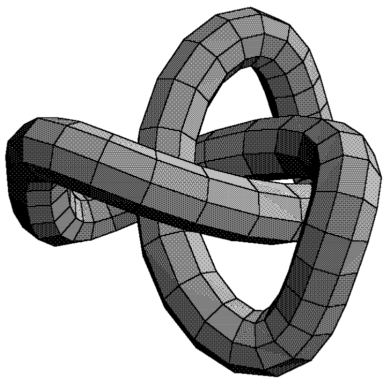
 ✠ ZPRAVODAJ CSTUG ✠ ZPRAVODAJ CSTUG ✠

TEX



1

ZPRAVODAJ 93

ČESKOSLOVENSKÉEHO SDRUŽENÍ UŽIVATELŮ TEXU

 ✠ TEX BULLETIN ✠ TEX BULLETIN ✠ TEX BULLETIN ✠

OBSAH

Komponenty TEXu

Joachim Schrod

TEX vyžaduje pomerne veľa doplňujúcich komponentov (súborov i programov), ktorých zmysel a vzájomná súčinnosť nie sú často známe. Tento článok osvetľuje jednotlivé zložky a vzájomné väzby jadra systému TEX, ktoré sú viditeľné pre užívateľa TEXu.

O obsahu tohto článku

TEX je sadzačský systém, ktorý ponúka autorovi ľahké využitie sadzačských vymožeností tlače na úrovni umenia sadzby pomocou počítačov. Toto však nie je len výsledkom programu TEX: značný počet doplňujúcich programov a súborov vytvára spolu kompletný sadzačský a autorský systém. Spolu s tými programami, ktoré patria priamo do TEXu, existujú ešte dva ďalšie veľké programy, ktoré boli vytvorené Donaldom Knuthom v spojitosti s TEXom a ktoré musia byť zahrnuté do popisu celého systému: METAFONT pre tvorbu fontov a `WEB`, dokumentačný a vývojový ‚jazyk' pre programovanie. TEX a METAFONT sú písané vo `WEB`e.

V tomto texte popíšeme toto ‚jadro' TEXu z hľadiska užívateľa: na konci čítania by ste mali mať prehľad o súčastiach systému TEX a o súboroch a podporných programoch, ktoré sú pre Vás ako užívateľa podstatné. Text však nie je úvodom do schopností TEXu alebo návodom, ako spustiť TEX na počítači.

Budem používať ‚rohové' poznámky na identifikáciu miest, kde budú po prvýkrát vysvetlené pojmy. Skratky pre typy súborov — obyčajne identifikované spoločnou príponou alebo rozšírením — budú vysádzané pomocou ‚`strojového`' písma. Uvedomte si, prosím, že tieto skratky sa niekedy nestotožňujú s príponou súboru (pozrite tiež tabuľku).

Tento článok je úvodným v sérii článkov, ktoré budú popisovať hore uvedené subsystémy a ich komponenty. V tejto sérii sa každý článok sústredí na jeden subsystém z istého pohľadu. Jeho výsledkom by nemal byť gigantický popis, ktorý povie všetko (a teda nič). Podľa môjho názoru nasledujúce témy by mohli byť zaujímavé:

- štruktúra a štandardná inštalácia TEXu

- `dvi` drajvery a fonty
- možnosti implementácie grafiky do TEXovských dokumentov
- komponenty METAFONTu
- štruktúra a štandardná inštalácia METAFONTu
- systém `WEB` — koncept *gramotného programovania*
- iné (aj keď ešte nenaplánované) témy prípadného záujmu
  - ▷ rozdiel medzi TEXom a DTP systémami
  - ▷ spôsob, ako pracuje TEX (na túto tému existuje niekoľko dobrých kníh)
  - ▷ hranice možností TEXu
  - ▷ TEX ako programovací jazyk

  Jednotlivé články budú publikované v uvedenom poradí.

Čo je to TEX

značkovanie, optické značkovanie, logické značkovanie

TEX je sádzací systém s veľkými možnosťami pre sadzbu formul. Jeho základným princípom je, že štruktúry sú vyznačené v texte, a potom transformované do sadzačského výstupu. Vyznačenie takejto informácie v štruktúre dokumentu sa nazýva *značkovanie*.[1]) Ak značky popisujú vzhľad dokumentu, tak sa nazývajú *optickými*, ak označujú štruktúru dokumentu, tak hovoríme o *logických značkách*. TEX podporuje obidva typy značiek. To znamená, exaktné riadenie vzhľadu častí dokumentu a ich umiestnenia, ako aj označkovanie štruktúry formul alebo zložiek dokumentu. Logické značky sú TEXom pretransformované do optických, takže vzhľad môže čitateľovi slúžiť na identifikáciu štruktúry. Vzhľad[2]) a vo všeobecnosti knižný dizajn nepredstavujú neužitočnú krásu. Dobrý knižný dizajn musí v prvom rade podporovať pochopenie obsahu v prospech čitateľnosti textu. Je *estetický* v jeho najlepšom slova zmysle, lebo spája formu a obsah a vytvára novú kvalitu.

sádzacia mašina, TEX82, balík makier

Jadrom sádzacieho systému TEX je formátovací program TEX82, ktorý sa často nazýva proste TEX. My tu prijmeme túto zvyklosť, pokiaľ rozdiel medzi celým systémom a formátovačom nie je dôležitý alebo je zrejmý. TEX82 je veľký monolitický program, ktorý je

[1]) v anglickom origináli markup

[2]) anglicky layout

publikovaný v knihe *TEX: The Program* od DONALDA KNUTHA. Jeho základné rysy môžu byť rozdelené do dvoch rovín:

1 TEX82 formátuje text, t.j. zalamuje odseky (včítane automatického rozdeľovania slov) a stránky.

2 Dáva k dispozícii programovací jazyk TEX, ktorý obsahuje mechanizmy pre tvorbu makier. To umožňuje zabudovanie nových príkazov na podporu značkovania na vyššej úrovni. DONALD KNUTH prezentuje jeden takýto príklad v TEXbooku: `plain` TEX. Súhrn makier, ktoré podporujú špecifický cieľ a majú (snáď) spoločnú filozofiu použitia, sa nazýva *balík makier*.

Rysy vyššej úrovne optického značkovania tak, ako sú reprezentované `plain` TEXom, umožňujú zabudovanie dodatočných úrovní pre úplné logické značkovanie. V súčasnosti sú široko rozšírené dva balíky makier pre logické značkovanie: $\mathcal{AMS}$-TEX a LATEX. Obidva systémy sú viac alebo menej postavené na `plain` TEXu a užívateľ môže v prípade potreby k logickému označkovaniu použiť aj optické z `plain` TEXu. To má za následok, že autor môže použiť zmes štrukturálnych informácií a explicitných informácií pre vzhľad — teda situáciu, ktorá pripúšťa veľkú škálu možností, ktoré môžu viesť (a aj vedú) k množstvu typografických nezmyslov.

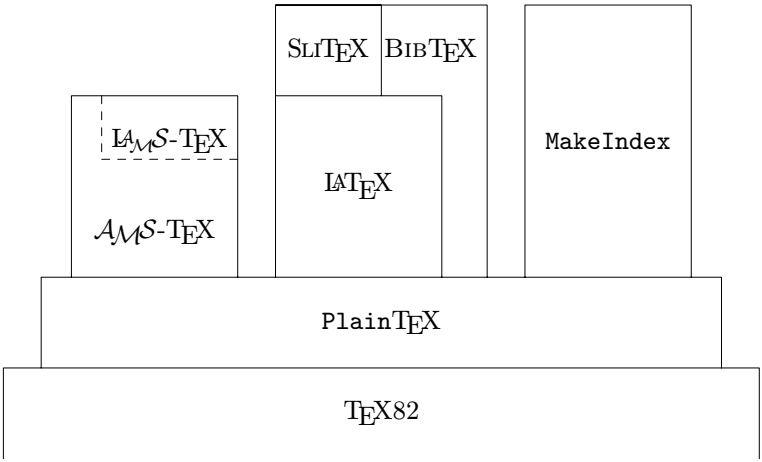
doplnkové programy

Pretože TEX bol vytvorený len pre sadzbu textov, a pre podporu realizácie nových značkovacích štruktúr, chýba mu veľa spôsobilostí, ktoré požadujú autori. Na podporu ďalších možností, ako zostavovanie indexov, zoznamov literatúry alebo na vtiahnutie grafiky, museli byť napísané dodatočné programy, ktoré využívajú informáciu získanú počas formátovania programom TEX82, spracujú ju a pripravia pre následné spracovanie programom TEX82. Veľmi rozšírené a dostupné pre mnoho typov a kombinácií počítač/operačný systém sú dva doplnkové programy: BIBTEX pre tvorbu zoznamov literatúry z bibliografických súborov a ***MakeIndex*** pre zostavovanie indexu.

Špeciálnym prípadom spracovania informácie, ktorú poskytuje TEX počas svojho behu, je zostavovanie obsahu a využitie krížových referencií v texte. K tomu potrebujeme len informáciu o čísle strany, čísle časti atď. Tieto údaje poskytuje TEX82, môže ich aj sám spracovať, takže TEX82 sa používa v tejto situácii aj ako vlastný post-procesor.

Videli sme, že sádzací systém TEX predstavuje súbor prostriedkov, ktoré pozostávajú zo sádzacej ‚mašiny' TEX82, balíkov makier (prí-

padne viacerých, ktoré využívajú ďalšie makrá) a doplnkových programov, ktoré sú používané spolu s týmito balíkmi makier. Tieto súvislosti sú ilustrované na obrázku 1.



Obr. 1: Komponenty TEXu

Formátovanie

Proces formátovania TEXom vyžaduje informáciu o rozmeroch znakov, ktorá je použitá pri zalamovaní odstavcov. Súbor znakov je zoskupený do *fontov*. (Toto je isté zjednodušenie, pretože pojem ‚font' by mal byť používaný pre realizáciu istých znakov v pevnej veľkosti pre špecifikované výstupné zariadenie.) Rozmery znakov fontu sa nazývajú ‚metrika fontu'.[3]) font

Formát, v ktorom je použitá metrika fontu TEXom, bol definovaný DONALDOM KNUTHOM a nazýva sa `TFM` formát (*TEX font metrics*). V tomto formáte je každý znak popísaný ako priehradka s istou výškou, hĺbkou a šírkou. TEX potrebuje len tieto rozmery a nezaujíma ho tvar znaku. Je dokonca možné, aby znak presahoval túto priehradku, čo môže viesť k prekrývaniu s inými znakmi. Rozmery znaku sú špecifikované v rozme-

[3]) anglicky *font metrics*

roch nezávislých od zariadenia,[4]) pretože zalamovací algoritmus TEXu je nezávislý od výstupného zariadenia.

vzory pre delenie

Počas zalamovania odseku TEX automaticky delí slová, a toto zalamovanie môže byť robené spôsobom, ktorý je skoro nezávislý od jazyka. Na jeho prispôsobenie k rôznym jazykom sú potrebné *deliace vzory*,[5]) ktoré parametrizujú deliaci algoritmus.

`dvi` drajver

Výsledkom TEXovského formátovacieho behu je `DVI` dokument, v ktorom je špecifikovaný typ a pozícia na stránke každého výstupného znaku. Rozlišovacia schopnosť, ktorá je použitá, je tak malá, že každé možné výstupné zariadenie bude mať hrubší raster, takže umiestnenie je naozaj nezávislé od zariadenia. `dvi` dokument špecifikuje len znak, ale nie fonty samotné, takže názov `dvi` (‚device independent') je presný. (Toto meno je problematické, lebo ‚DVI' je teraz chránená značka spoločnosti Intel, ale názov `dvi` v TEXu existoval skôr.) Na sprístupnenie výsledku formátovacieho procesu musí byť `dvi` súbor spracovaný tzv. `dvi` drajverom na požadovanom výstupnom zariadení.

Ak sa objavia problémy v procese formátovania, chybové hlásenia a varovania sa vypíšu na terminál. *Každá* správa, ktorá sa objaví na termináli, je tiež zapísaná do sprievodného protokolového súboru, ktorý sa nazýva `LOG` súbor. Do tohto `LOG` je možné uložiť aj ďalšie informácie, ktoré by boli príliš rozvláčne pre výstup na terminál. Ak sa tak stane, TEX to oznámi užívateľovi na konci formátovania. Hlásenia TEXu nie sú zabudované do programu, ale sú uložené v (reťazcovom) `POOL` súbore. Tieto hlásenia musia byť načítané na začiatku formátovacieho procesu.

Balíky makier

`plain` TEX

Základný balík makier je `plain` TEX, ktorý vyvinul DONALD KNUTH spolu s programom TEX82. Tento parametrizuje sádzaciu mašinu TEX82, takže táto môže sádzať anglicky text v sade znakov písma Computer Modern. `plain` TEX navyše poskytuje možnosti pre optické značkovanie. `plain` TEX je dostupný ako jeden zdrojový súbor `plain.tex`.

---

[4]) anglicky *device independent*

[5]) anglicky *hyphenation patterns*

Všetky iné autorovi známe balíky makier sú založené na `plain` TEXu, t.j. obsahujú zdrojový súbor `plain.tex`, či už celý alebo s modifikáciami menej dôležitých častí. K `plain` TEXu najbližšie najdôležitejšie (voľne) prístupné balíky makier sú $\mathcal{AMS}$-TEX od Michaela Spivaka a LaTeX od Leslie Lamporta. Ostatné voľné balíky makier majú často len lokálnu dôležitosť (napríklad BlueTEX, TEXT1 alebo TEXsis), alebo sa používajú len vo veľmi špeciálnych prostrediach (napríklad `texinfo` v projekte GNU alebo `webmac` pre `WEB`). Dôležité komerčné balíky makier sú MacroTEX autora Amy Hendrickson a LA$\mathcal{MS}$-TEX, ktorého autorom je tiež Michael Spivak.

$\mathcal{AMS}$-TEX, LaTeX, MacroTEX, LA$\mathcal{MS}$-TEX

štýly

Tieto balíky makier obyčajne pozostávajú z jadra, ktoré poskytuje dodatočné primitívne pojmy pre značkovanie. Z takýchto primitívnych pojmov môžu byť zostavené štýly dokumentov,[6]) ktoré realizujú logické značkovanie prostredníctvom príslušného vzhľadu. Tento vzhľad môže byť často upravovaný cez opcie *podštýlov* a štýlov, ktoré môžu poskytnúť ďalšie značkovanie.

doplnkové súbory

Balíky makier vytvárajú doplnkové súbory, ktoré obsahujú informáciu o zalamovaní stránok alebo o značkovaní dokumentov. Tieto informácie môžu byť využité podpornými programami — napríklad špecifikácia citácie z bibliografickej databázy alebo špecifikácia položky indexu so zodpovedajúcim číslom strany pre zostavenie indexu. Špeciálnym prípadom je informácia o krížových referenciách alebo nadpisoch pre zostavenie obsahu, lebo táto informácia môže byť zoskupená a opätovne použitá priamo TEXom.

SliTEX, LA$\mathcal{MS}$-TEX, MacroTEX

SliTEX je špeciálnou zložkou LaTeXu, ktorá je určená na prípravu fólií s prekryvmi. V TUGBoatu zväzok 10, číslo 3 (1989) bol ohlásený LA$\mathcal{MS}$-TEX, ktorý realizuje možnosti LaTeXu v $\mathcal{AMS}$-TEXu. MacroTEX je súhrn makro ‚modulov', ktoré môžu byť použité na realizáciu nových značkovaní, ale nakoľko sa stal dostupným len nedávno, nie je ešte veľmi rozšírený.

dodatočné fonty

Pri použití týchto (a ostatných) balíkov makier je potrebné sa presvedčiť, či nevyžadujú dodatočné fonty, ktoré nepatria do rodiny písma Computer Modern. Napríklad

[6]) document styles

pre LaTeX sú potrebné fonty pre dodatočné symboly a neviditeľné znaky (pre prekryvy fólií), zatiaľ čo $\mathcal{AMS}$-TeX potrebuje niekoľko dodatočných sád fontov pre matematické znaky a azbuku.

Podporné programy

BibTeX, ***MakeIndex***

Tu pojednáme len o dvoch podporných programoch: BibTeXe od Orena Patashnika pre prípravu bibliografií a ***MakeIndex*** od Pehonga Chena a Michaela Harrisona pre prípravu a triedenie indexu. Pre obidve úlohy existujú iné, funkčne ekvivalentné podporné programy. Ale spomínané dva sú dostupné pre veľa operačných systémov a maju ‚oficiálny' status, pretože Leslie Lamport povzbudzuje ich použitie s LaTeXom v jeho dokumentácii a TUG ich podporuje pre všeobecné použitie.

Neexistuje úplne prenosný mechanizmus pre zahrnutie všeobecnej grafiky do TeXovských dokumentov, a preto nie je k dispozícii strojovo nezávislý podporný program.

spôsob citovania, bibliografické štýly

BibTeX sa používa na spracovanie literatúry zhromaždenej v `BIB` súboroch. TeX vytvorí doplnkové súbory, ktoré obsahujú informáciu o požadovaných odkazoch na literatúru a BibTeX vygeneruje z nich utriedený zoznam literatúry v `BBL` súbore, ktorý vzápätí môže byť použitý TeXom. Usporiadane a spôsob citovania sú určené *bibliografickým štýlom*, ktorý je špecifikovaný v `BST` súboroch. Hlásenia z behu BibTeXu sú zapísané do `BLG` log-súboru.

indexový štýl

MakeIndex načíta `IDX` podporný súbor, ktorý obsahuje položky indexu a príslušné čísla strán, usporiada a zjednotí ich a zapíše ich do `IND` súboru, ktorý je vstupným súborom pre TeX. Štýl formátovania môže byť špecifikovaný indexovým štýlom. Hlásenia z behu ***MakeIndex***u sú zapísané do `ILG` súboru.

Zlepšenia výkonu

Veľa práce, ktorú vykoná TeX, je zhodnej pre každý dokument:

1. Celý text musí byť zalomený do riadkov. Slová častí textu v tom istom jazyku sú delené podľa tých istých vzorov.
2. Základné značkovanie príslušného balíka makier musí byť prístupné.

3 Požadované metriky fontov sú podobné pre veľa dokumentov, pretože použité sady fontov sa obyčajne nelíšia o veľa.

Kvôli zlepšeniu výkonu TEXu sú deliace vzory a popisy metrík fontov konvertované z externej reprezentácie do internej, ktorá je ľahko použiteľná programom TEX82. V prípade (1) a (2) je externá reprezentácia textová. Je rozumné previesť túto transformáciu len raz a nie pre každý dokument. Interná reprezentácia je uložená v `FMT` súbore. Uloženie sa uskutočňuje TEXovským príkazom `\dump`, a preto sa `FMT` súbory často nazývajú ‚dampované formáty'.[7]) `FMT` súbor môže byť načítaný na začiatku behu TEX82 a tak je dostupný pre spracovanie aktuálneho textu.

INITEX, produkčná verzia

Nakoľko `FMT` súbor sa vytvára nepravidelne — obyčajne pre zaktualizovanie balíka makier — formátovanie textu sa môže uskutočniť pomocou redukovanej verzie programu TEX82, ktorá neobsahuje uloženie a časti programu pre transformovanie vzorov pre delenie a pre dampovanie. Úplná verzia programu TEX82 je potrebná len v inicializačnej fáze, a preto sa volá INITEX. Dodatočné zlepšenia výkonu môžu byť dosiahnuté použitím produkčnej verzie programu TEX82, z ktorého sú odstránené časti pre štatistickú analýzu a odladovanie.

ViRTEX

Tie verzie TEXu, ktoré nemajú prednatiahnuté žiadne dampované formáty, majú schopnosť natiahnuť dampovaný formát (t.j. `FMT` súbor) a nemajú schopnosť dampovať `FMT` súbor (t.j. nie sú INITEX), sa často nazývajú VirTEX, čo je skratka z virgin TEX.

Typ Súboru	Vysvetlenie	Prípona, rozšírenie súboru, atď.
`TEX`	vstup textu	`tex,ltx`
`DVI`	výstup z TEX82, formátovaný text	`dvi`
`LOG`	log súbor z TEX82	`log, lis, list`
`HYP`	vzory delenia	`tex`
`TFM`	metriky fontov	`tfm`
`POOL`	reťazcový púl	`pool, poo, pol`
`FMT`	formátový súbor	`fmt`
`MAC`	súbor TEX makier	`tex, doc`
`STY`	štýlový súbor TEXu	`sty, tex, st, doc`

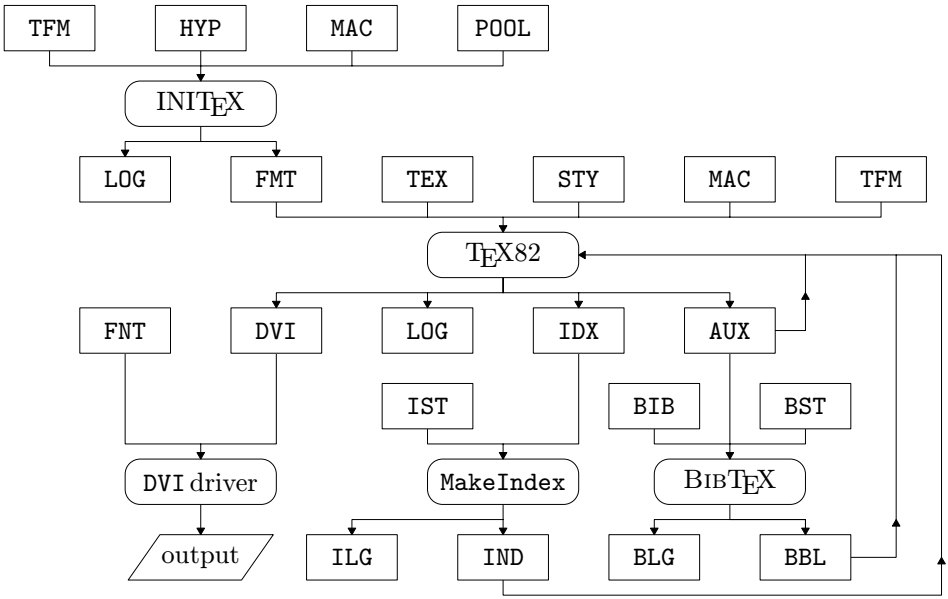
[7]) anglicky dumped formats

AUX	podporný súbor	aux, toc, lot, lof glo, tmp, tex
BIB	bibliografický súbor	bib
BBL	literatúra alebo bibliografia	bbl
BLG	log súbor z BibTeXu	blg
BST	štýlový súbor z BibTeXu	bst
IDX	neutriedený index	idx
IND	utriedený index	ind
IST	špecifikácia značkovania indexu	
ILG	log súbor z ***MakeIndex***u	ilg

Tab. 1: Typy súborov

Vzťahy medzi typmi súborov a komponentami

V predchádzajúcich častiach boli popísané komponenty systému TeX a spomenuté súbory, ktoré sú načítavané alebo písané týmito komponentami. Vzťahy medzi nimi sú graficky demonštrované na obrázku 2. Na



Obr. 2: Vzťahy medzi komponentami a typy súborov

tomto obrázku sú typy súborov reprezentované obdĺžnikmi a programy oválmi. Šípka znamená ‚je načítaný pomocou' alebo ‚vytvorený pomocou'. Skratky pre typy súborov sú vysvetlené v tabuľke, ktorá tiež uvádza identifikáciu súborov (prípony alebo rozšírenia), ktorú tieto súbory používajú (ale všimnite si, že sa používajú aj iné identifikácie súborov).

Poďakovanie.

Rád by som poďakoval Christine Detigovej, ktorá láskavo urobila anglický preklad. Nelson Beebeová navrhla urobiť preklad. Klaus Guntermann mal hodnotné pripomienky k prvej (nemeckej) verzii. Nico Poppelier prispel novou verziou obrázku 2, ktorá je lepšia než moja prvá. Publikovanie a distribúcia tohto článku je povolená len za podmienok analogických k GNU General Public Licence. Predchádzajúce revízie tohto článku boli publikované v ‚Die TEXnische Komödie' a v ‚Baskerville'.

Z anglického originálu publikovaného v TEXline č. 14
do slovenštiny převedl *Štefan Porubský*

Tabulky v plainTEXu

Jiří Veselý

Často se na naši hlavu snášejí výtky, že opomíjíme zařazovat do TEXbulletinu články pro ty, kteří s TEXem začínají nebo nedávno začali. Jim je určen následující přehledný popis tvorby tabulek v PlainTEXu. Jde spíše o seznam jednoduchých příkladů a ukázek než o teoretický výklad. Sazba tabulek v minulosti patřila v tiskárnách k nejnáročnějším operacím a ani při použití TEXu není triviální. Je ale spíše pracná než složitá. Pokud se budete muset sazbou tabulek zabývat častěji, vyplatí se poohlédnout po nějakém makru, které některé věci zjednoduší. Začneme „od Adama": použijme vstupní text

```
\halign{#&#&#\hfil\cr
Jméno    & Příjmení & Datum narození\cr
Adam     & Andrle   & 1.~1.~1911\cr
```

```
Barbora  & Blechatá & 2.~2.~1922\cr
Cyril    & Ceplecha & 3.~3.~1933\cr}
```
Ten dává po zpracování (spolu s vystředěním tabulky, které ve vzoru neuvádíme)

Jméno	Příjmení	Datum narození
Adam	Andrle	1. 1. 1911
Barbora	Blechatá	2. 2. 1922
Cyril	Ceplecha	3. 3. 1933

To ovšem jen přibližuje použitý princip, ale zdaleka to není uspokojivé. Můžete se například ptát, proč je v posledním sloupečku užito řídicí slovo `\hfil`. Sloupce jsou tak široké, aby se do nich vešla nejdelší položka sloupce; každá položka se uzavře do `\hbox` a pak se zpracuje. To, co se u prvních položek neprojeví, protože každá obsahuje jen jedno slovo, vypadá u poslední strašně. Odstranění `\hfil` v posledním sloupci tak vede k tabulce (tiskneme ale jen první řádek, při výsledné ošklivosti sloupce to plně postačí)
```
\halign{#&#&#\cr
Jméno    & Příjmení & Datum narození\cr
Adam     & Andrle   & 1.~1.~1911\cr...
```
Dostaneme tak následující „věc“

Jméno	Příjmení	Datum narození
Adam	Andrle	1. 1. 1911

Výsledek je jistě popudem k dalším experimentům. Přidáme proto nějaké mezery mezi sloupci a změníme typ písma v jednom sloupečku. Upravíme-li vstup na
```
\halign{\bf#&\qquad#&\qquad#\hfil\cr
...}
```
dostaneme snad o něco líbivější tabulku

Jméno	Příjmení	Datum narození
Adam	Andrle	1. 1. 1911
Barbora	Blechatá	2. 2. 1922
Cyril	Ceplecha	3. 3. 1933

Bude-li některý řádek tabulky delší (tj. bude obsahovat sloupeček navíc), nedojde ke kolizi, ponecháte-li ostatní řádky beze změny. Zkuste si takový pokus udělat (vše se řídí podle nejdelšího řádku).

Již jsme jednou použili „slabého péra“ `\hfil` k zarovnání posledního sloupce pomocí `#\hfil` na levou stranu. Obdobně lze použít `\hfil#` pro zarovnávání doprava a `\hfil#\hfil` pro centrování jednotlivých položek. Hned si to vyzkoušíme: dostaneme tak pomocí

```
\halign{\hfil#\hfil&\qquad\hfil\it#\hfil
&\qquad\hfil#\hfil\cr
\bf Příjmení &\bf Jméno  &\bf Datum narození\cr
...}
```

tabulku, která je snad ještě o něco přehlednější:

Jméno	**Příjmení**	**Datum narození**
Adam	*Andrle*	1. 1. 1911
Barbora	*Blechatá*	2. 2. 1922
Cyril	*Ceplecha*	3. 3. 1933

Takto tedy je možné vytvořit tabulku s centrovanými položkami a (polo) tučně vytištěným záhlavím. Připomínáme, že skupiny nesmí přesahovat „přes několik sloupců“ a že lze v některém sloupci použít např. i matematického módu, atp.

Chce-li TEX takovou tabulku lámat, učiní tak po kterémkoli řádku. To je však někdy nežádoucí. Zabráníme tomu např. tak, že použijeme konstrukce

```
\vbox{\halign{...\cr...}}
```

K „vystředění“ tabulek jsme od začátku používali tohoto jednoduchého triku: psali jsme

```
$$ \vbox{\halign{...\cr...}} $$
```

Tento často používaný trik v sobě skrývá i již zmíněný zákaz stránkového zlomu v tabulce.

Někdy je tabulka „velmi jednotvárná“ a přitom dlouhá. Existuje však způsob, jak si jednotvárné opakování ve vzorové řádce ušetřit. Způsob ukazuje následující příklad: budeme-li vytvářet tabulku popisující Fibonacciovu posloupnost, můžeme napsat pouze

```
$$\vbox{\halign{#\hfil\quad&&\hfil$#$\quad \cr
\bf Člen: &a_1& a_2&a_3&a_4&a_5&a_6&a_7&a_8&a_9\cr
Hodnota:  & 1& 1& 2& 3& 5&  8& 13& 21& 34\cr}}$$
```

a dostaneme

Člen:	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9
Hodnota:	1	1	2	3	5	8	13	21	34

Zde stojí za povšimnutí, že jakési „nastartování“ opakovacího mechanismu je možné v průběhu psaní vzorového řádku. Mechanismus je takový, že se vzorový údaj za `&&` opakuje tolikrát, kolikrát je (podle zadaných údajů pro obsazení tabulky) zapotřebí. Zde je na místě další upozornění důležité pro vyplňování vlastní tabulky: vynechávání „konců řádek“ tabulky před `\cr` **bez** nebo **s doplněním** zbývajících ampersandů odpovídajících vynechaným položkám **nevede** k témuž výsledku. Ve druhém případě se „vzorová řádka“ neignoruje. To v případě použití některých formátovacích příkazů hraje samozřejmě velkou roli.

Vraťme se však k našim experimentům s Adamem a spol. Někdy je důležité, aby se v tabulce nějaký údaj zpracoval individuálně. Pak stačí **na jeho začátku** uvést řídící slovo `\omit` a příslušná část (položka) vzorového řádku bude ignorována. To není podstatné např. při změně písma, kterou zvládneme lehce i bez tohoto řídicího slova, ale pro případ, že jde např. o speciální umístění (centrování, atp.). Jinak totiž bezprostředně uvedená instrukce o změně písma provede změnu na jiný font, než je uveden ve vzorové řádce. Uveďme příklad: ze vstupního textu

```
$$\vbox{\halign{\hfil#\hfil&\qquad\hfil\it#\hfil
&\qquad\hfil#\hfil\cr
\bf Jméno    & \bf Příjmení        &\bf Datum narození\cr
Adam         & Andrle              & 1.~1.~1911\cr
\it Barbora  & \bf Blechatá        & 2.~2.~1922\cr
Cyril        & \omit\quad Ceplecha & 3.~3.~1933\cr}}$$
```

dostaneme po zpracování TEXem (všimněte si zejména nestandardního ošklivého umístění prostřední položky posledního řádku)

Jméno	**Příjmení**	**Datum narození**
Adam	**Andrle**	1. 1. 1911
Barbora	**Blechatá**	2. 2. 1922
Cyril	Ceplecha	3. 3. 1933

S výhodou lze někdy použít i stabilního vkládaného údaje: tak dostaneme například pomocí následující úpravy vzorového řádku a standardního obsahu (položky v záhlaví a s Barborou jsme upravili zvlášť)

```
$$\vbox{\halign{\hfil Pan \it #\hfil
&\qquad\hfil\bf #\hfil
&\qquad\hfil nar.: #\hfil\cr
\omit\hfil\bf Jméno \hfil & Příjmení
&\omit\qquad\hfil\bf Datum narození\hfil\cr
```

```
Adam          & Andrle         & 1.~1.~1911\cr
\omit\hfil Paní \it Barbora
& Blechatá  & 2.~2.~1922\cr
Cyril         & Ceplecha       & 3.~3.~1933\cr}}$$
```
tuto tabulku:

Jméno	Příjmení	Datum narození
Pan *Adam*	**Andrle**	nar.: 1. 1. 1911
Paní *Barbora*	**Blechatá**	nar.: 2. 2. 1922
Pan *Cyril*	**Ceplecha**	nar.: 3. 3. 1933

V tabulkách lze užívat i složitější makra. Tak například při vyplňování adresáře do tabulky můžeme použít následující trik:

```
\def\adr#1!#2!{, 1#1\ 00 Praha #2}
$$\vbox{\halign{#\hfil&\qquad #\hfil\cr
\omit\bf Jméno \hfil & \omit\qquad \bf Adresa \hfil\cr
Adam Andrle & Alpinková 1\adr 10! 1! \cr
Barbora Blechatá  & Bezinková 2\adr 20! 2! \cr
Cyril Ceplecha & Citronová 3\adr 30! 3!\cr}}$$
```
čímž po zpracování dostaneme

Jméno	Adresa
Adam Andrle	Alpinková 1, 110 00 Praha 1
Barbora Blechatá	Bezinková 2, 120 00 Praha 2
Cyril Ceplecha	Citronová 3, 130 00 Praha 3

Někdy musíme mít v položce rozsáhlejší informaci. K tomu se hodí úprava, umožňující vložit údaj o více řádcích. A tak můžete nabídnout personálním pracovníkům tabulku tvaru

```
$$
\vbox{\halign{\hfil\it#\hfil&\qquad\hfil\bf#\hfil
&\qquad\vtop{\hsize 50mm\noindent#\strut}\cr
\bf Jméno    & \bf Příjmení   &\bf Charakteristika\cr
Adam         & Andrle
& Neustále vyvolává na pracovišti národnostní spory
a~třenice. Má nezákonný poměr s~Blechatou.\cr
Barbora  & Blechatá
& Je poněkud lehkomyslnější povahy, avšak kromě
poměru s~Andrlem má i~kladný poměr k~ldz.\cr
Cyril        & Ceplecha
```

```
& Tento pracovník je povahy mírné a~je proto v~kolektivu
velmi oblíben. Nemá však poměr, a~to je podezřelé.\cr}}
$$
```

Po zpracování personalista s letitými zkušenostmi z aparátu příslušných oddělení zajisté zajásá: údaje má přehledně jak na dlani:

Jméno	**Příjmení**	**Charakteristika**
Adam	**Andrle**	Neustále vyvolává na pracovišti národnostní spory a třenice. Má nezákonný poměr s Blechatou.
Barbora	**Blechatá**	Je poněkud lehkomyslnější povahy, avšak kromě poměru s Andrlem má i kladný poměr k ldz.
Cyril	**Ceplecha**	Tento pracovník je povahy mírné a je proto v kolektivu velmi oblíben. Nemá však poměr, a to je podezřelé.

Mezi řádky tabulky můžeme vkládat například dělící čáry, či jiný materiál (akceptovatelný ve vertikálním módu). To děláme pomocí konstrukce

```
... \cr \noalign{...}...,
```

kde mezi svorkami se nalézá to, co má být **mezi řádky** vloženo. Vložit se dá příslušné `\noalign{...}` opakovaně, tj. i několikrát za sebou. Vložíme-li však pouze `\hrule`, výsledek není příliš uspokojivý: vstupní text

```
\halign{\hfil#\hfil&\qquad\hfil\bf#\hfil
&\qquad\hfil#\hfil\cr
\bf Jméno &\bf Příjmení  &\bf Datum narození\cr
\noalign{\hrule}
...tabulka...
\noalign{\hrule}}
```

dává v našem standardním příkladu tabulku, která je snad o něco přehlednější, ale rozhodně ne esteticky uspokojivá: to proto, že při použití `\hrule` TEX nepoužije běžný `\interlineskip`.

Jméno	**Příjmení**	**Datum narození**
Adam	**Andrle**	1. 1. 1911
Barbora	**Blechatá**	2. 2. 1922
Cyril	**Ceplecha**	3. 3. 1933

Se získaným výsledkem nejste jistě spokojeni. Proto přistoupíme k dalším úpravám: „obklopíme“ (z jedné či obou stran) dělící čáry pomocí vhodných skipů — následující výsledek je získán pomocí vložení `\smallskip\hrule...`:

Jméno	**Příjmení**	**Datum narození**
Adam	**Andrle**	1. 1. 1911
Barbora	**Blechatá**	2. 2. 1922
Cyril	**Ceplecha**	3. 3. 1933

Kdybychom místo vodorovných čar vkládali pomocí `\noalign` nějaký text, může být situace složitější (je nutno upravit mezery mezi řádky). S posledním získaným produktem nejsme opět spokojeni, chtěli bychom mít tabulku rozdělenu i svislými čarami na jednotlivá pole. Pro napsání vstupního souboru je vhodné si nejprve představit celou tabulku postiženou malou „vnitřní explozí“ a rozbitou na jisté prefabrikáty:

Jméno	**Příjmení**	**Datum narození**
Adam	Andrle	1. 1. 1911
Barbora	Blechatá	2. 2. 1922
Cyril	Ceplecha	3. 3. 1933

Na základě této představy snadno napíšeme příslušný vstupní text (řešení je — pochopitelně — více)

```
$$\vbox{\offinterlineskip\halign
{\strut\vrule\quad\hfil#\hfil\quad
&\vrule\quad\hfil\bf#\hfil\quad\vrule
&\quad\hfil#\hfil\quad\vrule\cr
\noalign{\hrule}
\bf Jméno &\bf Příjmení &\bf Datum narození\cr
\noalign{\hrule}
Adam     & Andrle   & 1.~1.~1911\cr
Barbora  & Blechatá & 2.~2.~1922\cr
Cyril    & Ceplecha & 3.~3.~1933\cr
\noalign{\hrule} }}$$
```

Ten po zpracování dá následující tabulku. (Nepřehlédli jste použití řídicího slova `\strut`? Zajišťuje nám rozteč řádků, která by v závislosti na obsahu řádků mohla být nepříjemně nerovnoměrná.)

Jméno	**Příjmení**	**Datum narození**
Adam	**Andrle**	1. 1. 1911
Barbora	**Blechatá**	2. 2. 1922
Cyril	**Ceplecha**	3. 3. 1933

(♠)

Všimněte si také role nového řídicího slova `\offinterlineskip`; toto slovo nám fakticky umožňuje spojení svislých čar v tabulce pohodlným způsobem. Pozor ale na to, kam ho vkládáte: nebude-li vztaženo pouze k tabulce (umístění ve skupině), porušíme si sazbu dalších částí textu. Někdy bývá vhodné dát kousky svislých úseček do samostatných položek. Takové řešení dává větší svobodu při modifikacích. Uvedeme opět podstatnou část vstupního textu:

```
$$\vbox{\offinterlineskip
\halign{\strut#&\vrule#\quad&\hfil#\hfil
&\quad\vrule#\quad&\hfil\bf#\hfil
&\quad\vrule#\quad&\hfil#\hfil
&\quad\vrule#\cr
\noalign{\hrule}
&& \bf Jméno && \bf Příjmení && \bf Datum narození &\cr
\noalign{\hrule}
&& Adam      && Andrle       && 1.~1.~1911 &\cr...}
```

Po jeho zpracování dostaneme tabulku nerozeznatelnou od té, která je označena (♠). Nebudeme ji proto reprodukovat, místo ní uvedeme stejnou tabulku, v níž však každý `\quad` nahradíme dvojnásobným `\qquad`:

Jméno	**Příjmení**	**Datum narození**
Adam	**Andrle**	1. 1. 1911
Barbora	**Blechatá**	2. 2. 1922
Cyril	**Ceplecha**	3. 3. 1933

Již dříve jsme však tabulku „rozvolnili“ tím, že jsme obklopili vodorovné čáry vhodným „skipem“. To už teď tak snadno nejde (roztrhli bychom svislé čáry), můžeme si však pomoci např. takto: zavedeme novou definici (uvědomte si, co dělá)

```
\def\mez{\omit& height2pt&&&&&\cr}
```
a do každého řádku s `\noalign` přidáme na konec `\mez`. Tak dostaneme

Jméno	**Příjmení**	**Datum narození**
Adam	**Andrle**	1. 1. 1911
Barbora	**Blechatá**	2. 2. 1922
Cyril	**Ceplecha**	3. 3. 1933

Tak jsme dospěli k jednoduché tabulce, v níž lze sice ještě dále něco vylepšovat, ale to už si každý musí zkusit sám. Ukažme si dále použití řídicího slova`\tabskip`; celou manipulaci při tvorbě tabulky „na šířku stránky“ nebudeme podrobně vysvětlovat:

```
\offinterlineskip\tabskip0pt\halign to \hsize{
\strut#&\vrule#\tabskip3mm plus20mm\quad&
\hfil#\hfil&\quad\vrule#\quad&\hfil\bf#\hfil&
\quad\vrule#\quad&\hfil#\hfil&\quad\vrule#\tabskip0pt\cr
\noalign{\hrule}
&& \bf Jméno && \bf Příjmení && \bf Datum narození &\cr
\noalign{\hrule}
&& Adam      && Andrle       && 1.~1.~1911 &\cr
\noalign{\hrule}
&& Barbora   && Blechatá     && 2.~2.~1922 &\cr
\noalign{\hrule}
&& Cyril     && Ceplecha     && 3.~3.~1933 &\cr
\noalign{\hrule}}
```

Pomocí tohoto vstupu dostaneme:

Jméno	**Příjmení**	**Datum narození**
Adam	**Andrle**	1. 1. 1911
Barbora	**Blechatá**	2. 2. 1922
Cyril	**Ceplecha**	3. 3. 1933

Často se můžeme setkat s potřebou psát v tabulce „přes sloupce“. I to je celkem snadno řešitelné: použijeme dalšího důležitého příkazu `\multispan`. Toto řídící slovo umožňuje část sloupcové struktury v tabulce „přeskočit“ a odpovídající přeskočené sloupce spojit. Použití ukazuje příklad

```
$$\vbox{\offinterlineskip\halign{
\strut\vrule\quad\hfil#\hfil&\vrule\quad\hfil\bf#
```

```
\hfil\quad\vrule&\quad\hfil#\hfil\quad\vrule\cr
\noalign{\hrule}
\multispan3\strut\vrule\hfil
\bf Naše tabulka \hfil\vrule\cr
\noalign{\hrule}
\bf Jméno &\bf Příjmení  &\bf Datum narození\cr
\noalign{\hrule}
Adam     & Andrle & 1.~1.~1911\cr
\noalign{\hrule}
Barbora  &\multispan2\vrule\quad\hfil
    \it Nemám ji rád \hfil\quad\vrule\cr
\noalign{\hrule}
\multispan3\strut\vrule\quad\dotfill
    \it Cyril dodá údaje do zítřka
    \dotfill\quad\vrule\cr \noalign{\hrule} }}$$,
```

který dává po zpracování

Naše tabulka		
Jméno	**Příjmení**	**Datum narození**
Adam	**Andrle**	1. 1. 1911
Barbora	*Nemám ji rád*	
..... *Cyril dodá údaje do zítřka*		

K tomu uveďme několik obecných poznámek: použitím `\multispan2` v řádku s Barborou jsme spojili **dva** sloupce, proto se vynechává **jeden** ampersand. Prvním použitím jsme spojili pro tvorbu záhlaví tabulky **všechny** tři sloupce a tak prakticky vzorový řádek vymazali — všimněte si však „ošetření“ rámečku a užití `\strut` z téhož důvodu. Pozor na umístění: `\multispan`x musíme umístit na začátek vstupního údaje v prvním „slučovaném sloupci“; pokud číslo x je dvoumístné, je třeba je zapsat jako skupinu. Nyní můžeme místo `\omit` napsat `\multispan1`. Toto patrně nepoužijeme, nicméně možnost této záměny ukazuje, co vlastně `\multispan` provádí.

Tvorbu tabulek „po sloupcích“ umožňuje řídící slovo `\valign`. Nebudeme se tomuto problému podrobněji věnovat, uvedeme pouze dva příklady:

```
\valign{&\hbox{#\strut\qquad}\cr
    \bf Jméno &\bf Příjmení  &\bf Datum narození\cr
    Adam     & Andrle & 1.~1.~1911\cr
```

```
    Barbora  & Blechatá & 2.~2.~1922\cr
    Cyril    & Ceplecha & 3.~3.~1933\cr}
```
dává „transponovanou“ tabulku

Jméno	Adam	Barbora	Cyril
Příjmení	Andrle	Blechatá	Ceplecha
Datum narození	1. 1. 1911	2. 2. 1922	3. 3. 1933

Jestliže se rozhodneme pro další kosmetické úpravy, postupujeme zcela analogicky jako v předcházejícím případě, ale pracujeme „transponovaně“, tj. co jsme dříve dělali se řádky provádíme nyní se sloupci:

```
\valign{&\hbox{\strut\quad\hfil#\hfil\quad}\cr
\noalign{\vrule}
    \bf Jméno &\bf Příjmení  &\bf Datum narození\cr
\noalign{\vrule}
    Adam     & Andrle & 1.~1.~1911\cr
\noalign{\vrule}
    Barbora  & Blechatá & 2.~2.~1922\cr
\noalign{\vrule}
    Cyril    & Ceplecha & 3.~3.~1933\cr
\noalign{\vrule}}
```
dává jinou „transponovanou“ tabulku

Jméno	Adam	Barbora	Cyril
Příjmení	Andrle	Blechatá	Ceplecha
Datum narození	1. 1. 1911	2. 2. 1922	3. 3. 1933

Jiným (a snad do jisté míry i jednodušším) způsobem tvorby tabulek je **tabbing**, silně připomínající užití tabulátorů na psacím stroji. Není však tak flexibilní — v některých TEXových učebnicích se s ním ani nesetkáte, podle autora je jeho použití projevem „špatných TEXových mravů“. Avšak i přes toto varování: nač tedy tabbing je? Řádka v „nastaveném“ tabbingu má tvar

```
\+ ... & ... & ... & ... \cr
```
a po provedeném nastavení lze mechanismu vytvořených zarážek použít prakticky kdykoli. Využíváme toho i při jiných příležitostech. Centrování se dá provést obvyklým trikem přes

```
$$\vbox{\+...\cr  \+...\cr}$$
```

Jak se však nastaví šíře sloupců? Jedna z možností je nastavit explicitně šířky sloupců. Provedeme s pomocí „vzorové šířky“: vstupní text

```
\settabs
\+\kern20mm & \kern40mm & ... \cr
... tabulka ...\cr
```

dává po spojení s naší standardní „příkladovou tabulkou“ po zpracování

Jméno	**Příjmení**	**Datum narození**
Adam	Andrle	1. 1. 1911
Barbora	Blechatá	2. 2. 1922
Cyril	Ceplecha	3. 3. 1933

Tím se nastaví prakticky **dva** tabulátory na pozicích 20 a 60 mm (zde však je třeba mít na paměti, že dokumenty se často zvětšují pomocí `\magnification` apod.). Jestliže chceme tabulátory rozmístit rovnoměrně, použijeme konstrukci typu

```
\settabs 5\columns
```

Tak bychom měli dostat tabulku o pěti sloupcích na šířku stránky — podobně jako v předcházejících případech se standardní náplní získáme (umístili jsme položky do druhého, třetího a čtvrtého sloupce, první a poslední jsme neobsadili):

Jméno	**Příjmení**	**Datum narození**
Adam	Andrle	1. 1. 1911
Barbora	Blechatá	2. 2. 1922
Cyril	Ceplecha	3. 3. 1933

Jedna věc je možná poněkud překvapující: TEXse vždy „vrací“ na předešlý tabulátor, při psaní však následující tabulátor „uvolňuje“ automaticky. Zvolíme-li exotičtější (a delší) jména, dostaneme očekávaný **špatný** výsledek:

Jméno	**Příjmení**	**Datum narození**
Xantipa	Nabuchonodozorová	před n.l. …
Josif Vissarionovič	Džugašvili	21. 12. 1879
Adolf	Hitler	20. 4. 1889

Tentokrát jsme začínali psát rovnou, bez nastavování tabulátorů

```
\+&\bf Jméno &\bf Příjmení
    &\bf Datum narození &\cr
\+&Xantipa  & Nabuchonodozorová
    & \dots před n.l. \dots &\cr ...,
```

protože je máme nastaveny z předchozího použití.

Známe-li nejdelší položky v jednotlivých sloupcích, můžeme postupovat také takto

```
\settabs \+ \bf Jméno & Blechatá & Datum narození\cr...
```

Tak je šířka sloupců zadávána pomocí obsahu. Často bývá dána přímo záhlavím tabulky; pozor, i v tomto případě je nutné záhlaví zopakovat (vzorová řádka se netiskne).

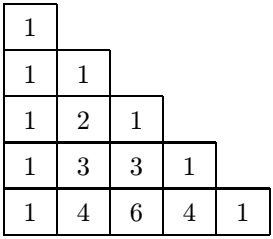
Nastavení tabulátorů se „uvolňují“ pomocí řídícího slova `\cleartabs`. Z důvodu bezpečí (nepamatujeme si, jak jsme co nastavili) je někdy doporučováno **pokaždé** začínat oběma řídicími slovy `\cleartabs\settabs`. Řídicí slovo `\cleartabs` užité v kterékoli vstupní řádce zruší tabulátory od svého umístění až do konce řádky. Vytváření místa kolem textu ve sloupcích je podobné jako v `\halign`, stejně tak i oddělování položek čarami. Oddělení řádku tabulky lze provést jednoduše uvedením `\hrule` mezi koncové `\cr` a po něm následující `\+`. I tímto mechanismem se můžeme propracovat k poměrně dokonalé tabulce. Připomínám, že centrování položek lze zařídit na základě stejného principu, který jsme již poznali, avšak **je nutno používat** důsledně `\hfill` („slabá pera“ typu `\hfil` jsou již použita k posunu obsahu položky k zarážce. V každém případě:

Pozor na horizontální rozměr tabulky !

Uveďme malou ukázku:

```
$$\vbox{\offinterlineskip\cleartabs
\def\hr{\vrule height .4pt width 2em}
\def\vr{\vrule height 12pt depth 5pt}
\def\cc#1{\hfill#1\hfill}
\+ \hr&\cr
\+ \vr\cc1&\vr\cr
\+ \hr&\hr&\cr
\+ \vr\cc1&\vr\cc1&\vr\cr
\+ \hr&\hr&\hr&\cr
...
\hrule
```

dává následující elegantní výsledek.



Na závěr alespoň heslovitě uveďme srovnání obou možností tvorby tabulek (připomínám, že jsme již zmínili, že pro snazší sazbu tabulek existuje celá řada maker; o jednom psal v TEXbulletinu 1/92 Mirek Dont):

Popis funkce:	**Tabbing**	`\halign`
Uchovávání vzoru zarovnání	*ano*	*ne*
Lze sloupce předefinovávat?	*ano*	*ne*
Lze dělat libovolně dlouhé tabulky?	*ano*	*ne*
(`\halign` ukládá celou tabulku před sazbou)		
Lze tabulky lámat (stránkový zlom)?	*ano*	*ano*
Lze položky centrovat nebo zarážet?	*ano*	*ano*
(u tabbingu musíme pera dávat do všech položek)		
Může položka přesahovat přes několik sloupců?	*ano*	*ano*
(u tabbingu jsou však potíže s péry)		
Lze přidávat do tabulek dělící čáry?	*ano*	*ano*
(u tabbingu je popis nutno opakovat)		
Určuje se šíře sloupců automaticky?	*někdy*	*ano*
Jak se zvládají výjimky?	*špatně*	*snadno*
Lze „společné věci vytknout“?	*ne*	*ano*
Lze sloupce oddělovat automaticky?	*ne*	*ano*
Lze šířku tabulky předepsat?	*ne*	*ano*

Jiří Veselý
`jvesely@cspguk11`

Automatizovaná tvorba formulářů pomocí LaTeXu

Jiří Rybička

V běžné kancelářské praxi je velmi často potřeba vyplňovat různé formuláře. Blankety formulářů se koupí ve větším množství (vždy musí být určitá provozní zásoba) v tiskárně, v dnešních cenách pohybujících se od 0,40 do 0,90 Kčs/kus. V okamžiku potřeby je blanket písařkou vložen do psacího stroje, kde nastává nepříjemný a vždy nerovný souboj mezi vkládaným textem a místem, které je v blanketu rezervováno. Stane-li se, že blanket poněkud zastará (dnes velmi často změna bankovního účtu, přečíslování ulic, změna `soudruh` ⟶ `pan` apod.), je navíc potřeba předepsané texty škrtnout nebo přepsat novým textem, v horším případě se zbytek zásob nepoužitelných blanketů zmaří na podružné účely.

Má-li být formulář alespoň trochu k světu, stojí to za těchto podmínek mnoho úsilí — vadí překlepy, nevhodně umístěný text apod. Chyby se musí často napravovat i přepisem na nový blanket.

Přirozeným řešením uvedených problémů je instalace systému formulářů na počítači. Přípravou textu „na nečisto“ na obrazovce se automaticky vyloučí obtíže s překlepy nebo nevhodnými formulacemi v textech. Je-li navíc systém obohacen o urychlení a zautomatizování správné lokalizace vpisovaných textů, je vyřešeno mnoho těžkostí.

Konkrétní programové řešení může být provedeno nepochybně mnoha způsoby. Je však vhodné využít běžně dostupného, rozšířeného a inteligentního vybavení, které se snadno doplní o jednoduché součástky.

Popis systému

Velmi vhodným inteligentním systémem je systém TeX, pro tento účel byl zvolen LaTeX, v němž jsou vysázeny všechny blankety. Uživatel do zvoleného blanketu doplní konkrétní texty, formulář se vysází a zobrazí na obrazovce. Po kontrole a případných opravách se provede výsledný tisk.

V tomto místě je nutno předeslat, že uživatel, pro nějž byl systém vyvinut, je počítačový laik, bylo tedy třeba všechny systémové záležitosti ukrýt tak, aby se musel učit co nejméně ovládacích kláves.

Činnost systému je řízena hlavní dávkou operačního systému MS DOS s názvem `Form`. Dávka má jeden parametr, který specifikuje vybraný formulář. Formuláře jsou očíslovány od 1 do 17, některé z nich ještě mají varianty — např. „Dohoda o ukončení pracovního poměru“ má jednu variantu pro ženu a druhou pro muže. Varianty jsou rozlišeny písmeny, např. `5a`, resp. `5b`.

Uživatel má k dispozici tabulku, v níž jsou uvedena čísla jednotlivých formulářů. Byla zkoušena varianta systému, v níž se na úvod spustil program zobrazující nabídku všech formulářů. Po provozních zkušenostech však byl tento program zrušen, neboť výběr z nabídky byl shledán daleko těžkopádnějším než zadání čísla formuláře do parametru dávky. Tato skutečnost je potvrzením všeobecně známého faktu, že programy „přeplácané“ různými nabídkami jsou spíše na obtíž než k ulehčení práce. Jedná-li se o víceméně rutinní činnost, je výběr z nabídky oproti zadání jednoho čísla skutečně unavující. Druhou (nepřímou) výhodou takto pojatého řešení je pružnost vzhledem ke změnám v repertoáru formulářů: není nutné měnit programy, stačí přidat nový blanket a říci obsluze, že existuje další číslo.

Text řídicí dávky `Form` je následující:[1])

```
1.  @echo off
2.  if not exist for\%1.txt goto chyba
3.  copy for\%1.txt prac.txt
4.  : znovu
5.  formedit prac.txt
6.  call latex prac
7.  if not exist prac.dvi goto znovu
8.  call view prac
9.  : rozhod
10. jakdal
11. if errorlevel 6 goto laser
12. if errorlevel 4 goto dotprn
13. if errorlevel 2 goto znovu
14. goto konec
15. : dotprn
```

[1]) V textu dávky jsou řádky pro lepší orientaci očíslovány.

```
16.  call dot prac
17.  goto rozhod
18.  : laser
19.  call las prac
20.  goto rozhod
21.  : chyba
22.  echo Číslo formuláře není udáno nebo je chybné.
23.  : konec
```

Úvodní příkazy dávky zjišťují, zda je korektně zadáno číslo formuláře. Test je proveden na základě existence blanketu zadaného formuláře. Blankety jsou uloženy v souborech s názvy `for*.txt`, kde hvězdička představuje příslušné číslo formuláře. V kladném případě je blanket zkopírován do pracovního souboru, v němž jsou prováděny všechny úpravy.

Klíčovou součástí systému je editor. V dávce je na řádku 5 volán editor `formedit`, který představuje obyčejný textový editor, u něhož jsou definována klávesová makra, usnadňující pohyb v textu. V tomto konkrétním případě se jedná o editor `dbFast`, u něhož je v konfiguraci definována klávesa `F7` pro pohyb o jedno vpisovací pole vpřed a klávesa `F8` pro pohyb o jedno pole vzad. Vpisovací pole je definováno textem `{\vstup }`, přičemž příkaz `vstup` s definicí `\def\vstup{\hskip 3ex \large \bf}` zajišťuje odstup vkládaného textu od „předtisku“ o 3ex a zvýrazňuje písmo tak, aby bylo na formuláři na první pohled patrné, co je text blanketu a co je vpisovaný text.

Stiskem klávesy `F7`, resp. `F8` se kurzor nastaví těsně za slovo `vstup`. Na řádku následujícím je komentář, vysvětlující, jaká položka je právě editována a jaký vstupní údaj se očekává, např.:

```
...
...
v souladu s \S~65 zákoníku práce
mu bude přiznána odměna ve výši
{\vstup } Kčs
%%%%%% ^ výše přiznané odměny
...
...
```

Po skončení editace je vyvolána dávka `latex` pro překlad pracovního souboru do tvaru `.dvi`. Dávka `latex.bat` obsahuje volání překladače TEXu v dávkovém režimu (s parametrem `/b`), čímž je zamezeno „zakousnutí“ překladu. V případě výskytu chyby (zjištěno testem `if errorlevel 2`) se smaže soubor `prac.dvi` a na obrazovku se vypíše

stručná zpráva o nalezených chybách (ze souboru `prac.log` jsou vyfiltrovány řádky se specifikací chyb, tj. řádky začínající vykřičníkem a řádky popisující umístění chyby). Je-li pak v dávce `form` zjištěno, že soubor `prac.dvi` neexistuje, pokračuje se skokem zpět na editaci, v opačném případě se vyvolá prohlížeč (dávka `view`).

Po prohlédnutí formuláře na obrazovce se aktivuje program `jakdal` (řádek 10), který na obrazovce zobrazí okno se čtyřmi možnostmi volby:

tisk Laserová tiskárna
tisk Jehlová tiskárna
Oprava formuláře
Konec práce

Podle uživatelovy volby vydá program `jakdal` příslušný výstupní kód, jehož hodnotu testují následující příkazy v dávce. Jestliže je zvolen tisk, aktivují se příslušné tiskové dávky (`laser`, resp. `dot`) a pak se znovu aktivuje tatáž nabídka. Při volbě `Oprava` se řízení vrátí na řádek 5 — formulář je možné opět editovat.

Program `jakdal` je programován v Turbo Pascalu. Nabídku je možné řešit mnoha různými způsoby, lze přijmout i velmi jednoduché řešení, které lze zapsat několika málo řádky. V tomto případě byl použit systém Turbo Vision, celý program sestává pouze z volání předdefinovaných procedur vytvářejících příslušnou nabídku.

Systém uchovává pouze jediný (poslední) vyplněný formulář. Z praktických zkušeností vyplývá, že uchovávání formulářů v elektronické podobě by nebylo přínosné — bylo by nutné vést agendu vyplněných formulářů a dělat spoustu jiných (servisních) činností, čemuž zdaleka neodpovídá potřeba návratu ke starším verzím.

Systém je již provozován zhruba rok, zkušenosti z provozu ukazují, že je tímto způsobem možné řešit zpracování standardních textů v kancelářské praxi k plné spokojenosti uživatelů i jejich nadřízených.

Dvojité hranaté závorky v matematice

Petr Olšák

> *V matematické sazbě rozeznáváme pět druhů závorek: oblé ( ), hranaté [ ], složené (svorky) { }, dvojité ⟦ ⟧ a lomené ⟨ ⟩.*
>
> Pavel Pop: *Ruční sazba,* učebnice pro stud. obor Polygrafie, 1983.

Před nedávnem jsem potřeboval zařadit do matematických textů dvojité hranaté závorky. Kolega Miroslav Dont sehnal na síti font `bbold` v METAFONTových zdrojových textech. Font obsahuje malá i velká písmena psaná dvojitým tahem, dále také řecká písmena, některé matematické operátory a zmíněné závorky. Autorem fontu je Alan Jeffrey.

U „dodávky“ jsem nenašel makro pro zavedení těchto fontů do TEXu. Přitom font obsahuje příslušné informace umožňující sazbu těchto závorek v libovolné velikosti a dále obsahuje plno zajímavých symbolů a operátorů. Myslím si, že by mohlo čtenáře zajímat, jak se takový font zařazuje do matematiky, aby vše fungovalo jak v základní, tak v indexové a podindexové pozici a aby byly využity všechny možnosti fontu.

Vytvořil jsem soubor `bb10.tex`, který obsahuje zavedení uvedeného fontu do „desetibodového“ dokumentu. Na obsahu tohoto souboru budeme ilustrovat problematiku zařazení matematického fontu.

```
% Macro loads the bbold fonts for mathematics in 10pt size

                                % Example for 12pt size:
\font\bbtext=bbold10            % \font\bbtext=bbold12
\font\bbscript=bbold7           % \font\bbscript=bbold9
\font\bbscriptscript=bbold5     % \font\bbscriptscript=bbold7
\font\bbex=cspex10              % \font\bbex=cspex10 scaled 1200

\newfam\bbfam
\newfam\bebfam

\textfont\bbfam=\bbtext
    \scriptfont\bbfam=\bbscript
    \scriptscriptfont\bbfam=\bbscriptscript
```

```
\textfont\bebfam=\bbex
    \scriptfont\bebfam=\bbex
    \scriptscriptfont\bebfam=\bbex

\def\sixt#1{\ifcase#1 0\or1\or2\or3\or4\or5\or6\or7\or8\or9\or
 A\or B\or C\or D\or E\or F\fi}

\def\[{\delimiter"4\sixt\bbfam5B\sixt\bebfam02 }
\def\]{\delimiter"5\sixt\bbfam5D\sixt\bebfam03 }

\mathchardef\squarcap="1\sixt\bebfam46
\mathchardef\circlevee="1\sixt\bebfam50
... atd...

\def\bbmathcodes{%
   \mathcode`\*="2\sixt\bbfam2A \mathcode`+="2\sixt\bbfam2B
   \mathcode`\(="4\sixt\bbfam28 \mathcode`\)="5\sixt\bbfam29
   ... atd...
   \mathchardef\alpha="0\sixt\bbfam0B
   \mathchardef\beta="0\sixt\bbfam0C
   ... atd...}

\def\bb{\fam\bbfam\bbmathcodes\bbtext}
```

Je vidět, že jsou vlastně použity čtyři fonty — `bbold10` pro základní velikost, `bbold7` pro indexovou velikost, `bbold5` pro podindexovou velikost a `cspex10` obsahující speciální symboly. Vpravo od příkazů `\font` za procenty je uveden příklad pro zavedení fontů pro dvanáctibodový dokument. Analogicky lze zavést i fonty pro základní velikost jedenácti bodů. Chceme-li udělat univerzálnější makro pro všechny velikosti (po vzoru LaTeXu), lze v místě příkazů `\font` zahrnout větvení podle přání uživatele. Všechny potřebné velikosti fontů jsou v dodávce `bbold` obsaženy.

Makro zavádí dvě nové „rodiny“ (`\fam`). Rodina `\bbfam` obsahuje velká a malá písmena abecedy latinské i řecké (𝔸 𝔹 ℂ 𝔞 𝔟 𝔠 𝔬 ϐ ...), některé speciální znaky (# $ ⅋ & ⟨ ⟩ \ ⫩ ∗ ⫽) a další. Tato rodina shrnuje uvedené znaky ve třech velikostech (základní, indexová a podindexová poloha). V této rodině jsou také „výchozí“ velikosti závorek ⟦ ⟧.

Druhá rodina `\bebfam` zahrnuje jedinou velikost fontu speciálních symbolů. Obsahuje „zvětšené“ formy závorek ⟦ ⟧ a sedm operátorů typu „suma“ ve velikosti pro textový styl i „display“. Jedná se o tyto operátory: ⨅ ⨆ ⫼ ⫼ ⨄ ⨁ ⨂.

Definice `\def\sixt#1` definuje příkaz `\sixt`, který vrací šestnáctkový zápis čísla příslušné rodiny. V dalších definicích se výsledek tohoto příkazu použije jako jedna cifra šestnáctkového zápisu kódu jednotlivých matematických objektů.

Následuje definice příkazů `\[` a `\]`. Tyto příkazy budou vracet příslušné dvojité hranaté závorky ve všech velikostech i libovolně „natahovatelné“. Kód příkazu `\delimiter` čtěte např. takto: čtverka – jedná se o matematický objekt typu „open“ (otevírací závorka), `\sixt\bbfam5B` – výchozí velikost se bere z fontu rodiny `\bbfam` z pozice 5B a konečně `\sixt\bebfam02` – zvětšená velikost se vezme z fontu rodiny `\bebfam` z pozice 02 (hexadecimálně). V posledně jmenovaném fontu je uložena informace tvaru „řetěz postupně se zvětšujících závorek“, přičemž tento řetěz je zakončen odkazem na tři znaky fontu: horní část, dolní část a střední část závorky. V případě požadavku na obří závorku si TEX vezme horní a dolní část a střed vyplní dostatečným počtem opakování střední části. Takové závorky vidíte po stranách tohoto odstavce. Stejný princip znáte z fontu `cmex10`, kde jsou uloženy „natahovací“ verze všech běžně používaných závorek.

Nyní například

```
$$  \left\[ {A+B\over C + \left\[ {D+E\over F} \right\]} \right\],
    \quad \[1\]_{\[2\]_{\[3\]}}.  $$
```

povede na

$$\left[\!\left[\frac{A+B}{C+\left[\!\left[\frac{D+E}{F}\right]\!\right]} \right]\!\right], \quad [\![1]\!]_{[\![2]\!]_{[\![3]\!]}}.$$

Upozorňuji uživatele LATEXu, že `{x\over y}` znamená totéž co `\frac{x}{y}` a že označení dvojitých závorek sekvencemi `\[` a `\]` jim může způsobit komplikace, protože tyto sekvence jsou v LATEXu použity pro vstup do matematického módu a výstup z něho (jakási náhrada za dolary). Pokud tyto sekvence nikdy v tomto smyslu neužíváte, můžete je s klidným svědomím používat pro dvojité závorky (předefinovat původní příkaz). Ke kolizím nedochází.

Pokud si budete chtít uvedenou definici opsat, je vhodné upozornit na důležitost mezery za kódem příkazu `\delimiter`. Kdyby tam nebyla, pak se například `$\[1+1\]$` expanduje na

```
$\delimiter"485B9021+1\delimiter"585D903$
```

což je určitě něco jiného, než

```
$\delimiter"485B902 1+1\delimiter"585D903 $
```

Dále v makru následují definice operátorů typu „suma“. Například příkaz

```
\mathchardef\squarcap="1\sixt\bebfam46
```

čtěte: sekvence `\squarcap` se definuje jako matematický objekt typu „suma“ (jednička v kódu), jehož menší (textová) verze se vezme z fontu rodiny `\bebfam` z místa 46 (hexadecimálně). Ve fontu existuje odkaz z menší verze na větší, takže se o to nemusíme v TEXu starat a například

```
\squarcap_{n=1}^\infty = \circlevee_0^\infty
```

povede uvnitř textu na $\bigsqcap_{n=1}^{\infty} = \bigcirc\!\!\!\!\bigvee_0^\infty$, zatímco v „display“ stylu dostaneme

$$\bigsqcap_{n=1}^{\infty} = \bigcirc\!\!\!\!\bigvee_0^\infty$$

Konečně příkaz `\def\bb` v makru definuje přepínač písma `\bb`, který pracuje podobně jako například známý přepínač `\bf`. Použití příkazu `\bb` v textovém režimu způsobí přepnutí fontu díky příkazu `\bbtext` (příkaz `\fam\bbfam` se neprojeví), zatímco v matematickém režimu naopak „zabere“ příkaz `\fam\bbfam` a příkaz `\bbtext` nemá vliv. Takže například 𝕥𝕖𝕩𝕥 𝕫 𝕟𝕒𝕤𝕖𝕙𝕠 𝕟𝕠𝕧𝕖𝕙𝕠 𝕗𝕠𝕟𝕥𝕦 byl vypsán sekvencí

```
například {\bb text z~naseho noveho fontu} byl ...
```

Původní font nemá háčky a čárky. Pozorný čtenář knížečky „The Pamphlet on TEX Fonts“ si jistě všiml, že font `bbold10` je demonstrován pouze na anglické citaci. Zatím jsem neměl potřebu ani chuť háčky do metafontových zdrojů dopsat. Jinak na tom byl Karel Horák, který tuto věc dodělal. 𝕡𝕣𝕠𝕥𝕠 𝕛𝕖 𝕞𝕠žné 𝕥𝕦𝕥𝕠 𝕦𝕜ázku 𝕧𝕪𝕤ázet 𝕠𝕡𝕣𝕒𝕧𝕕𝕦 česky.[1])

Mnohem užitečnější asi bude používat font v matematickém režimu. Proto přepínač `\bb` navíc spouští příkaz `\bbmathcodes`, v němž se (lokálně — v rámci skupiny) předefinují matematické objekty jako např. `+`, `\alpha` a podobně. Takže třeba

[1]) Na přání autora doplněno při závěrečné redakci bulletinu. Ve skutečnosti je ovšem nutno použít jiný (textový) font, protože v původním fontu nezbývá pro diakritická znaménka místo.

```
$$  {\bb \alpha (A + B) } = \alpha (A + B) $$
```

vede na

$$\mathbb{\alpha (A + B)} = \alpha (A + B)$$

V definici příkazu `\bbmathcodes` například

```
\mathcode‘\*="2\sixt\bbfam2A
```

lze číst takto: symbol `*` v matematickém módu bude znamenat binární operátor (dvojka v kódu) braný z fontu rodiny `\bbfam` z pozice 2A (hexadecimálně). Podobně se čtou příkazy `\mathchardef`, které definují kontrolní sekvence `\alpha`, `\beta` atd. Tyto definice nemám vypsané důsledně všechny ani ve vlastním souboru maker, protože se mi jeví výhodnější zařazovat nové matematické kontrolní sekvence až v případě potřeby. Vyžaduje to pracovat s tabulkou fontu `bbold10` a s rozmyslem.

Poznamenejme ještě, že příkaz `\mathcodes` v definici přepínače `\bb` způsobuje, že `\bb` není „ryzím“ přepínačem. Použití této sekvence způsobem:

```
obyč. text \bb jiny text \rm obyčejný text
```

způsobí totiž trvalé předefinování matematických kódů. Je proto nutné buď psát

```
obyč. text {\bb jiny text} obyčejný text
```

nebo zrušit příkaz `\bbmathcodes` z definice sekvence `\bb`.

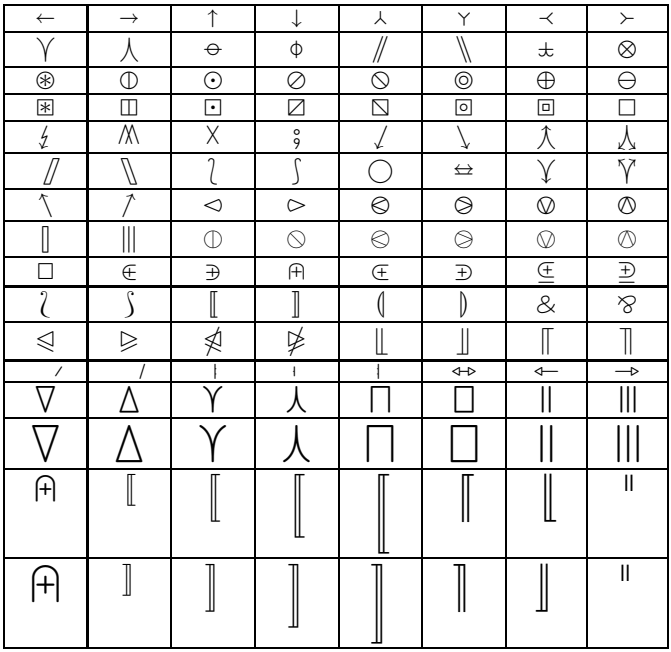
Na závěr se přiznám, že jsem ztratil hodně úsilí při vypátrání jedné „záludnosti“. Všimněte si, že v definici `\bbmathcodes` není u příkazu `\mathcode‘+=...` použit backslash před znakem plus, jak se sluší a patří (viz definice všech ostatních kódů). Totiž kontrolní sekvence `\+` je definována v Plainu pro účely tabbingu jako `\outer\def`. To způsobuje, že nelze sekvenci `\+` použít uvnitř těla definice v *žádném* smyslu. Tj. ani pro definici matematického kódu znaku `+` .

Uvedené makro `bb10.tex` není zdaleka univerzální. Na to přijdou velice rychle všichni uživatelé LaTeXu. Pokud například píšete poznámky pod čarou v petitu a v těchto poznámkách se vyskytne nějaký symbol z fontu `bbold`, je třeba použít font `bbold8`, přičemž ten není makrem `bb10.tex` zaveden. Podobně, pokud se speciální symbol vyskytne ve zvětšeném nadpise, je nutno zavést příslušně zvětšené fonty. Technika zavádění fontů automaticky podle požadované velikosti je vyřešena v LaTeXu (soubor `lfonts.tex`) a ještě lépe je řešena v NFFS. Kdo chce,

může se těmito makry inspirovat. Sám si myslím, že použití speciálního fontu v nadpise či zmenšené poznámce je poměrně málo časté. Proto je možné takový výskyt řešit individuálně. Pokud ale navrhujeme formát knihy (kde např. všechna cvičení budou sázena menším písmem), je nutno celou věc do makra zahrnout a vše pořádně domyslet. Makrojazyk TEXu nám neklade žádná omezení.

Článek jsem dal číst kolegovi Dontovi, který nedávno přinesl zajímavou zprávu pro doplnění. Na síti nalezl balík, zvaný `St Mary's Road`, v němž je zahrnut font se zmíněnými závorkami (tabulka dole, autory jsou Alan Jeffrey a Jeremy Gibbons). Navíc tam najdete LATEXovské styly pro zavedení fontu do dokumentu a font obsahuje podstatně větší množství nových matematických znaků. Stačí tedy pouze uvést příslušný název stylu a o příkazy typu `\mathcode` apod. se nemusíte starat. Chtěl jsem ale ve svém článku ukázat, že zavedení fontu na úrovni TEX-primitivů nemusí být zase tak komplikované, třebaže se jedná o matematický font. Výhodou je, že přesně víte, co máte zavedeno a jak.

17.1.1993 *Petr Olšák*



Jak dostat obrázky z programu *Mathematica* do TEXu

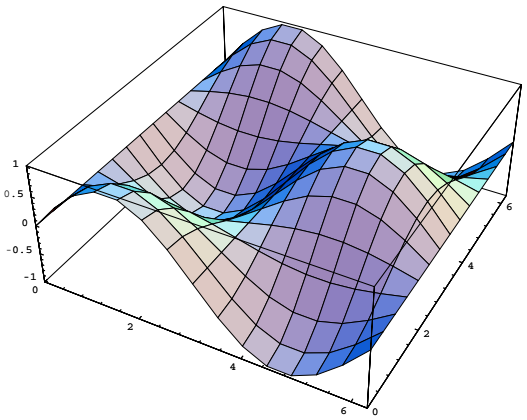
Petr Olšák

Program *Mathematica* Stephena Wolframa má charakteristický dovětek ke svému názvu: „System for doing Mathematics by computer“. Tím je zhruba řečeno, o co jde. Dále je známo, že TEX je „sazeč“ textů, zvláště technických, např. s komplikovanou matematikou. Z toho vyplývá, že uživatelé TEXu se často stávají i uživateli programu *Mathematica* a obráceně. Každý, kdo se stane uživatelem obou systémů, si zcela zákonitě položí otázku, formulovanou v názvu tohoto článku. Také já jsem si ji položil a musím přiznat, že odpověď se mi hledala nesnadno.

Jako příklad uvažujme reálnou funkci dvou proměnných $f(x) = \sin(x)\cos(y)$. Studentům většinou říkám, že představu o této funkci získají tím, že si prostudují formu na vajíčka. Druhá možnost je, nechat si tuto funkci znázornit programem *Mathematica*. Spusťme program *Mathematica*, který nám nabídne prompt ve tvaru `In[1]:=` . Napíšeme požadavek zobrazení naší funkce třeba takto:

```
In[1]:= Plot3D[Sin[x] Cos[y], {x,0,2Pi}, {y,0,2Pi}]
```

Po odeslání tohoto požadavku se nám na obrazovce objeví obrázek:



Chceme-li obrázek uložit do souboru, napíšeme v programu *Mathematica* příkaz

```
In[2]:= Display ["obraz.g", %1]
```

Příkaz způsobí, že se obrázek, vytvořený předchozím (prvním) příkazem uloží do souboru se jménem `obraz.g` . Pokud máme připojenou tiskárnu, můžeme ještě zkusit obrázek vytisknout. Provedeme to například příkazem

```
In[3]:= Hardcopy [%1]
```

K počítači jsem měl připojenu laserovou tiskárnu bez PostScriptu a po vytisknutí obrázku jsem se nestačil divit*). V tisku byly dosti podstatné chyby. Obrázek se rozdělil na jakési vodorovné pásy a ty na sebe vůbec nenavazovaly! Pokud jsem tentýž pokus udělal na maticové tiskárně (24 jehliček), dopadlo to ještě katastrofálněji.

Pro zařazení obrázku do TEXu potřebujeme spíš vědět, jak dopadlo uložení obrázku do souboru. V programu *Mathematica* proto ukončíme činnost pomocí příkazu

```
In[4]:= Quit
```

a podíváme se do souboru `obraz.g`. Jedná se o textový soubor, v němž je obrázek popsán jazykem PostScript. Tento soubor ale nelze poslat bez doplňujících údajů ani na tiskárnu vybavenou PostScriptem, protože mu chybí hlavička (soubor definic), které jazyk PostScriptu potřebuje načíst dřív, než bude zpracovávat tento obrázek.

Podívejme se proto podrobněji, jak funguje tisk obrázků z programu *Mathematica* pro DOS. Příkazem `Hardcopy` se nejdříve vytvoří přechodný soubor s podobným obsahem, jako je soubor `obraz.g`, a pak se z prostředí programu *Mathematica* spustí dávka s názvem `hardcopy.bat`, která může mít třeba tento obsah (je to závislé na instalaci)

```
@echo off
printps -printer hplj.300 -dev LPT1 -fonts c:\math\fonts %1
```

Je vidět, že se vlastně spustí program `printps.exe` s příslušnými parametry. V případě, že tiskárna nemá PostScript (parametr `hplj`), funguje

*) Program *Mathematica* jsem měl možnost zkoušet ve verzi 2.0 pro DOS a pro UNIX (Open Windows). Více jsem se zaměřil na verzi pro DOS, a proto všechny následující technické vlastnosti se týkají této verze.

program `printps.exe` jako interpret PostScriptu (jak se ukazuje z pokusů použití příkazu `Hardcopy`, funguje v dané verzi jako *špatný* interpret). Pokud je tiskárna vybavena PostScriptem, program `printps.exe` pouze připojí ke vstupnímu souboru hlavičku s definicemi příkazů PostScriptu speciálně používaných programem *Mathematica* a soubor pouze „překopíruje“ na tiskárnu.

My budeme pro zařazení obrázku do TEXu potřebovat vytvořit PostScriptový soubor, který hlavičku s definicemi obsahuje. Přirozeně, že k tomu použijeme program `printps.exe`, který zavoláme například tímto příkazem:

```
printps -eps obraz.eps -fonts c:\math\fonts  obraz.g
```

Tento příkaz vytvoří ze vstupního textového souboru `obraz.g` výstupní soubor `obraz.eps`, který již je opatřen příslušnými hlavičkami. Takovému formátu se říká „Encapsulated PostScript Form“. Samozřejmě, pokud zpracováváme více obrázků, je vhodné použít například dávku s obsahem:

```
printps -eps %1.eps -fonts c:\math\fonts  %1.g
```

Skutečnost, že soubor `obraz.eps` již obsahuje hlavičku s definicemi, poznáme podle začátku souboru, který vypadá takto:

```
%!PS-Adobe-2.0 EPSF-2.0
%%BoundingBox: 0 0 540 335
%%Creator: Mathematica
...
```

Všechny tyto úvodní řádky jsou komentáře jazyka PostScript, protože začínají procenty. (Kým byli asi programátoři firmy Adobe inspirováni, když navrhovali jazyk PostSriptu?) Zvláště charakteristický je druhý řádek s textem `BoundingBox`, kde je uveden rozměr výšky a šířky obrázku. Tuto informaci bude dále potřebovat software, který použijeme pro zařazení obrázku do textu.

Napišme nyní zdrojový text TEXu, který může vypadat třeba takto

```
\input epsf
...
... se na obrazovce objeví obrázek:

\centerline{\epsfysize=6cm \epsfbox{obraz.eps}}

\noindent Chceme-li ...
```

Makro `epsf.tex` obsahuje definice příkazů `\epsfbox`, `\epsfysise` a dalších. Toto makro načítá z úvodních řádků souboru `obraz.eps` parametry

příkazu `BoundingBox`, z nichž zjistí poměr výšky a šířky obrázku. Na základě těchto údajů a požadované výšky obrázku (v příkazu `\epsfysize`) vypočítá rozměry boxu `\epsfbox` a tento box uloží do `.dvi` souboru. Můžeme také zadat šířku boxu pomocí `\epsfxsize`, přičemž výška se dopočítá automaticky. Do souboru `.dvi` se také uloží prostřednictvím příkazu `\special` název souboru s obrázkem (v našem případě `obraz.eps`) a PostScriptová instrukce na zvětšení nebo zmenšení obrázku do požadovaných rozměrů. Zmíněné makro je součástí programového balíku `dvips`. Je tam zdokumentováno i s příkladem použití.

Po zpracování TEXem si výsledek můžeme prohlédnout Mattesovým prohlížečem. Místo obrázku ovšem uvidíme jen prázdné místo, protože příslušný příkaz `\special` tento prohlížeč neakceptuje.

Máme-li „odladěný“ text dokumentu, můžeme přikročit k vlastnímu zařazení obrázku. K tomu použijeme program `dvips.exe`, který načítá soubor `.dvi` a vytváří z něj soubor `.ps`, ve kterém jsou PostScriptové popisy jednotlivých stránek dokumentu. Tento program dále v místě výskytu příkazu `\special` vloží do výstupního souboru kopii souboru `obraz.eps`, případně překopíruje příslušné PostScriptové instrukce, uvedené v příkazu `\special`.

Program `dvips.exe` potřebuje navíc znát, kde jsou uloženy metriky a bitové mapy fontů, používaných v TEXu. Tyto informace převede do PostScriptového popisu stránek. Příslušné adresáře sdělíme programu `dvips.exe` prostřednictvím jeho konfiguračního souboru `config.ps`, v němž je například napsáno:

```
* Claim 180kbytes memory
m 180000
* Default resolution.
D 300
* Paths
T e:\emtex\tfm
P e:\emtex\pixel300\dpi%d\%f.%p
*V e:\texfonts\vf
*L c:\texfonts\;lj_0;lj_h;lj_1;lj_2;lj_3;lj_4;lj_5a;lj_5b;lj_sli
*S .;e:\emtex\texinput
*H .;e:\emtex\ps
...
```

Další údaje nás pro začátek nemusí zajímat. Hvězdičkou je označen komentářový řádek. Znamená to, že jsme prostřednictvím příkazů `T` a `P` označili adresáře pro metriky a soubory `.pk`, zatímco ostatní řádky jsou „odkomentovány“ a slouží jako příklad, co lze ještě v konfiguračním sou-

boru napsat. Jedná se například o adresáře k virtuálním fontům, knihovnám fontů apod.

Nyní bychom si chtěli prohlédnout na obrazovce, jak vypadá text i s obrázkem. K tomu lze použít public domain interpret PostScriptového popisu stránek, jehož výstupem může být jak obrazovka, tak tiskárna bez PostScriptu. Jmenuje se GhostScript. Pro prohlížení na obrazovce stačí napsat

```
gs386 soubor.ps > nul
```

kde `soubor` je název dokumentu. Přesměrování textových hlášení programu do `nul` je provedeno proto, aby se hlášení programu nemíchalo s grafikou (což se normálně děje).

Jsme-li rozmazleni komfortem Mattesova ovladače, asi nás to trošičku zklame. Lze skákat pouze po jednotlivých stránkách a zvětšení je pouze jediné — takové, aby na obrazovce byla vidět celá stránka textu. Všichni víme, že při takovém zvětšení se text nedá číst, a proto je nutné mít text odladěn už dříve. Zato obrázek vidíme dokonce barevně (máme-li barevnou grafiku).

V tuto chvíli jsem získal další negativní zkušenost. Obrázek jsem sice viděl v požadované velikosti společně s textem, ale nebyl na požadovaném místě. Byl poněkud výše. Tady se projevila další chyba drahého programu *Mathematica*. Konvertor obrázků programu *Mathematica* `printps.exe` nenastaví správné hodnoty parametru `BoundingBox`! Problém jsem vyřešil vytvořením krátkého testovacího „programu“ v TEXu s tímto zněním:

```
\input epsf
\vglue5cm
\vbox{\hrule\hbox{\vrule\epsfbox{obraz.eps}\vrule}\hrule}
\bye
```

Po spuštění programů `tex.exe`, `dvips.exe`, `gs386.exe` jsem na obrazovce viděl rámeček pro obrázek a navíc obrázek, který byl poněkud vpravo a nahoře. Nyní stačí editovat parametry `BoundingBox` v souboru `obraz.eps`. První dva parametry jsou souřadnice levého dolního rohu rámečku a druhé dva označují pravý horní roh. Zvětšováním souřadnice x se posunujeme doprava a zvětšováním y se posunujeme nahoru. Souřadnice jsou uvedeny v jednotkách `bp`, což je skoro totéž, co `pt`. Po změně parametrů znova použijeme výše uvedený testovací program pro kontrolu, zda jsme změnu provedli správně. Například pro obrázek z první stránky tohoto článku byly použity parametry:

```
%%BoundingBox: 50 200 560 590
```

Někdy ovšem není editace souboru `obraz.eps` tak snadná, protože soubory pro složitější obrázky dosahují délky několika megabytů. K tomu ovšem nabízí makro `epsf.tex` snadnou výpomoc. Parametry `BoundingBox` lze totiž v souboru `obraz.eps` ignorovat, a použít místo nich parametry explicite napsané v hranatých závorkách za příkazem `\epsfbox`, tedy například:

```
\centerline{\epsfysize=6cm
  \epsfbox[50 200 560 590]{obraz.eps}}
```

Na závěr si něco řekneme o fontech pro popisy obrázků. (Například popisy os apod.)

Pokud se spokojíme s fonty, které používá *Mathematica*, výrazně nám to usnadní práci. Tyto texty jsou v PostScriptovém popisu uloženy s odkazem na PostScriptový font `courier`, případně další fonty. Jestliže GhostScript tyto fonty v instalaci nenajde, nahradí je fontem `ugly`, což je jednoduché, poněkud hranaté písmo. V obou případech se tyto popisy zvětšují a zmenšují společně s obrázkem v libovolném poměru. Tento způsob bývá většinou dostačující a dokonce při tak drobném písmu pro popisy os, jaké vidíte v ukázkách, není příliš patrný rozdíl mezi použitím originálních PostScriptových fontů a písmem `ugly`. Zvláště při použití „hrubého“ rozlišení koncového zařízení (pouze 300dpi). Nicméně jsem přesvědčen, že Karel Horák se s tímto písmem nespokojí a instaluje pro účely tohoto článku příslušné PostScriptové fonty.

Chceme-li použít fonty z TEXu, musíme si rozmyslet jejich umístění na stránce a pomocí příkazů TEXu (např. v LATEXu je vhodné prostředí `picture` s příkazem `\put`) umístit texty přesně tam, kam potřebujeme. Zde je třeba upozornit na to, že texty popisů je nutno uvést *později*, než volání samotného obrázku příkazem `\epsfbox`. PostScript totiž překrývá dříve umístěné objekty na stránce novými. Výsledek je mnohem lepší, nicméně je to pracné a velikost popisů je pevná, tj. nemění se při dodatečné změně velikosti obrázku.

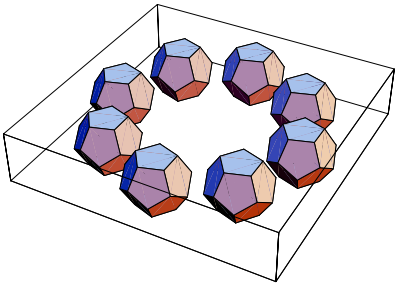
Pokud se nám už poloha obrázku včetně popisů líbí, přistoupíme k vlastnímu tisku. Pro tiskárnu, která není vybavena PostScriptem, stačí napsat například příkaz:

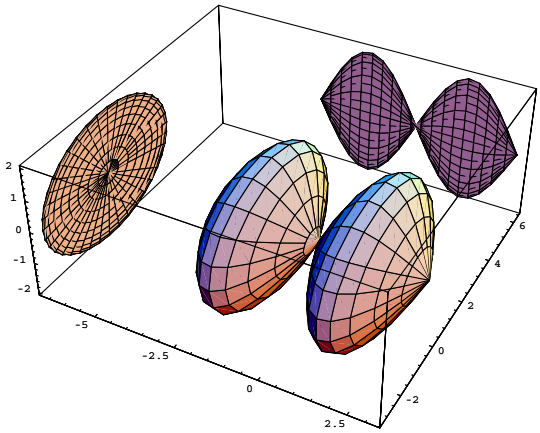
```
gs386 -sDEVICE=laserjet -r300 soubor.ps
```

Poznamenejme, že použití dávky `gslj.bat`, která je součástí instalace GhostScriptu nám na naší tiskárně nefungovala. Tiskárna začala vypisovat zmatené symboly stránku za stránkou v textovém režimu.

Při úspěšném tisku zmizí barva v obrázku a vytisknou se rastry s různými odstíny šedi. Máme-li ale barevnou laserovou tiskárnu, bude asi vybavena PostScriptem a pro závěrečný tisk nám stačí „poslat“ soubor `.ps` na tiskárnu, přičemž obrázky zůstanou barevné.

Protože příprava bulletinu neprobíhá na barevných tiskárnách (a také následující reprografické zařízení nepracuje s barvou), spokojíme se s černobílými ukázkami obrázků programu *Mathematica*.





3.1.1993 *Petr Olšák*

O dvou dalších možnostech začleňování obrázků do textu

Petr Vejda

Obě metody se opírají o možnosti ovladačů emTEXu začleňovat do textu grafické soubory ve formátu PCX. Připouštím, že tato metoda není úplně „TEXově čistá“, ale používáte-li emTEX a máte-li potíže s většími technickými obrázky, pak si následující příspěvek určitě přečtěte.

Důvody pro tento příspěvek

Náš ústav vydává již řadu let časopis Acta Technica. Před dvěma lety jsem byl pověřen, abych organizačně i technicky zajistil jeho sazbu v TEXu. Tento úkol jsem již dávno splnil, mimo jiné i díky panu Oldřichu Ulrychovi, který pro nás napsal nový styl. Stále však narážíme na problémy s obrázky. Zatím jsme tuto situaci řešili vždy za pomoci nůžek a lepidla. S velkým nadšením jsem si z EuroTEXu přinesl 386-kovou verzi METAFONTu. Tento program se mi sice velmi líbí a sám ho hojně používám, ale na kreslení vysloveně technických obrázků (a právě takové jsou v našem časopise) se vůbec nehodí. Důvody jsou tyto:

- Žádný náš přispěvatel se METAFONT nechce učit. (Pravdivost tohoto tvrzení pouze předpokládám.)
- S převodem z jiných grafických formátů do METAFONTu jsem neuspěl. (Program HP2XX je vysloveně slabý!)
- I když nakreslíme technický obrázek přímo v METAFONTu (např. graf funkce dvou proměnných), METAFONT nám tak veliký obrázek stejně obvykle nepřeloží. (A to ani 386-kový METAFONT spolu s mnou upraveným programem GFtoPK.)

Proto zde uvádím dva postupy, které nejsou založeny na použití METAFONTu a dávají dobrý výsledek. Navíc nadměrně nezatěžují nervy a čas uživatele.

Metoda prvá — program PrintGL

Nejrozšířenějším grafickým jazykem v našem ústavu je jazyk HP-GL, kterým jsou řízeny především plottery firmy Hewlett Packard. Ovladače pro výstupní zařízení, která jsou vybavena interpretem tohoto grafického jazyka, existují téměř ve všech grafických knihovnách. Obrázky jsou v tomto jazyce popsány vektorovým způsobem, což umožňuje libovolně měnit velikost obrázku beze ztráty na jeho kvalitě. Dlouho jsem se snažil sehnat program, který by uměl převádět obrázky z jazyka HP-GL do nějakého tvaru, který je zpracovatelný TEXem. Nakonec jsme koupili od americké firmy Ravitz Software, Inc., program **PrintGL**, který dokonale simuluje HP plotter pro velké množství tiskáren a umí převést obrázek i do formátu PCX, což je pro nás nejdůležitější.

Nyní se seznámíme s postupem začlenění obrázku. Předpokládejme, že obrázek popsaný jazykem HP-GL máme v souboru `GRAFY.PLT`, náš článek chceme prohlížet na obrazovce v základní velikosti (300 bodů na palec) a chceme ho pak vytisknout 1,44-krát zvětšený na tiskárně. (Tedy jakoby 432 bodů na palec.) Dále nechť má mít obrázek v textu rozměry 110×80 mm v základní velikosti. Zadejme tedy příkaz:

```
PRINTGL /DGRAFY.PCX /FZ128,125  /L10,7.5 /M1 /O2
        /PGRAFY.PLT /S0003 /WCEAC /Z5,2
```

Tím jsme vytvořili soubor `GRAFY.PCX`, kde je náš obrázek pro základní velikost. Program **PrintGL** má velké množství parametrů, které dovolují uživateli zvolit vše tak, jak si přeje. Použití parametrů je přehledně a srozumitelně popsáno v návodu k programu v souboru `PRINTGL.DOC` nebo v příručce, kterou obdržíme, když si program zakoupíme. Pro orientaci stručně uveďme význam parametrů, které jsme právě použili:

- `/D` Jméno výstupního souboru
- `/F` Zvolený grafický formát a údaje o hustotě v obou směrech
- `/L` Rozměry stránky
- `/M` Zvětšení
- `/O` Orientace obrázku
- `/P` Jméno vstupního souboru
- `/S` Nastavení různých odstínů šedi pro různá pera
- `/W` Nastavení šířek per
- `/Z` Míra hladkosti hladkých křivek

Soubor `GRAFY.PCX` přeneseme do adresáře, kam se dívají naše emTEXové ovladače, když hledají grafický soubor v základní velikosti. Tento adresář

je zadán parametrem výstupních zařízení, který se uvádí např. v souborech `SCR.CNF` a `HP.CNF`. Podrobný návod najdeme v adresáři `\TEX\DOC` v souboru `DVIDRV.DOC`. Například pro novou verzi obrazovkového prohlížeče používám specifikaci:

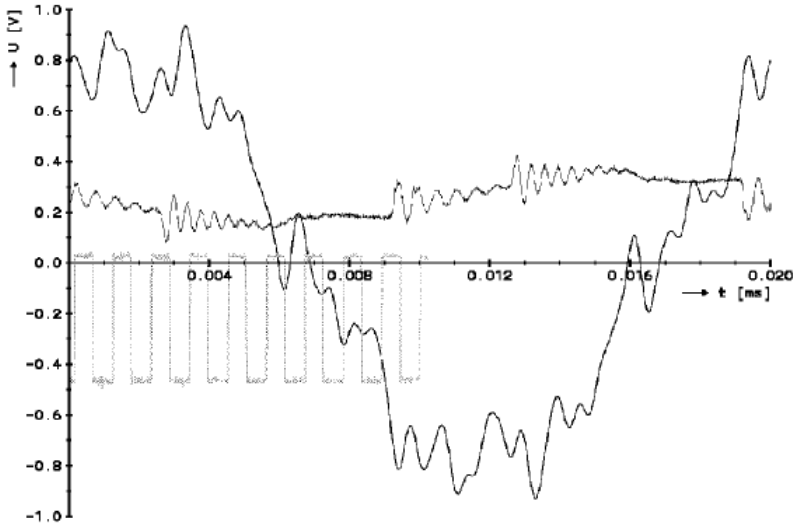
```
/pg={F:\TEX\300\@f,\TEX\GRAPH\300\@f}
```

Tím ovladači oznamuji, že má hledat grafický soubor nejprve v adresáři `F:\TEX\300`, a nenajde-li ho, pak v adresáři `\TEX\GRAPH\300`.

Dále zadáme příkaz:

```
PRINTGL /DGRAFY.PCX /FZ186,180  /L10,7.5 /M1 /O2
        /PGRAFY.PLT /S0003 /WCEAC /Z5,2
```

Tím jsme vytvořili soubor `GRAFY.PCX` 1,44-krát zvětšený a uložíme ho tedy do adresáře pro toto zvětšení. (Což je u mě adresář `F:\TEX\432`.) Na tomto příkladě si povšimněme, že jsme zvětšení neprovedli pomocí parametru `/M` ale pomocí parametru `/F`. Parametr `/M` by nám totiž neumožnil zvětšit celou pracovní plochu, ale na stejné pracovní ploše by se nám zobrazil pouze výřez z obrázku.



Obr. 1

Obrázek můžeme v PlainTEXu začlenit příkazy:

```
\midinsert
\special{em:graph GRAFY.PCX}
\vskip 82mm
\rm Obr.\ 1
\endinsert
```

Je třeba, abychom si uvědomili, že TEX nemá možnost zjistit, jak začleňovaný obrázek skutečně vypadá, tedy ani jaké má obrázek ve skutečnosti rozměry. Jedinou informací o velikosti obrázku je náš příkaz `\vskip 82mm`. Potřebujeme-li z nějakých důvodů sdělit TEXu informace o velikosti obrázku zcela přesně, chceme-li např. nechat TEXu obrázek zarámovat nebo přesně sesadit několik obrázků, můžeme použít program **PCXInfo**. O tomto programu bude zmínka v následující kapitole.

V případě, že jsme nikde neudělali chybu, dostaneme správný (a až na zvětšení totožný) výsledek jak pro základní velikost, tak při zvětšení 1,44-krát. Příkladem takto začleněného obrázku je obr. 1.

Metoda druhá – program DJtoPCX

V této kapitole se budeme zabývat problémem, jak začlenit do textu obrázky z některých programů, které pracují pod systémem MS WINDOWS. Náš ústav koupil před vánoci program *Mathematica 2.1* firmy Wolfram Research, Inc., který pracuje pod systémem MS WINDOWS 3.1. Je to pozoruhodný program, který skvěle kreslí na laserové tiskárně. Samozřejmě, že jsem se ihned snažil použít obrázky v TEXu. Systém WINDOWS umožňuje jednoduchým způsobem převést každý obrázek nakreslený na obrazovce do formátu PCX. Touto cestou bychom však daleko nedošli! Kdybychom takový soubor začlenily přímo do textu, byl by příliš malý. (Body na obrazovce jsou daleko větší než na tiskárně.) Neosvědčilo se mi použít ani jinak dobrý program **BM2FONT**. Tím sice můžeme obrázek zvětšit, ale obrázek zároveň „zhrubne“ a zdaleka nedosahuje kvality obrázku, který nakreslí *Mathematica 2.1* přímo na tiskárně. (Vnitřním grafickým jazykem „Matematiky“ je totiž PostScript a obrázky na obrazovce a na tiskárně jsou dvě zcela různé „bitmapové“ realizace téhož obrázku v jazyce PostScript.) Nabízí se tedy, abychom použili obrázek přímo ve tvaru, jak ho posíláme na tiskárnu. K tomuto účelu má sloužit program **PCLPIC**. Nemůžeme ho však použít, neboť tento program „nesnese“, aby byl obrázek ve skutečnosti „sestaven z několika kusů“, což je obvykle náš případ. Nakonec mi nezbylo jiné řešení,

než napsat podobný program, který by dokázal obrázek korektně „poslepovat“. Ukázalo se však, že kód, kterým jsou řízeny laserové tiskárny, je poměrně obtížný. Například některé části obrázku jsou posílány do tiskárny jako „nové fonty“ a podobně! Zvolil jsem tedy následující metodu. Obrázek necháme nakreslit do souboru (např. `KNOT.DJ`) v jazyce, který používá tiskárna HP DeskJet. Tato tiskárna má také rozlišení 300 bodů na palec, ale kód je poněkud jednodušší. Dále soubor `KNOT.DJ` zpracujeme programem **DJtoPCX**. Ještě před vytvořením souboru `KNOT.DJ` je nutno přesně nastavili parametry ovladače tiskárny HP DeskJet ve WINDOWS.

Pro MS WINDOWS 3.1

V okně HP DeskJet

Resolution:	`300 dots per inch`
Paper Size:	`A4 210 x 297 mm`
Paper Source:	`Auto Sheet Feeder`
Memory:	`None`
Orientation:	`Portrait`
Cartridges (max:2):	`None`

V okně Options

Dithering:	`Line Art`
Intensity Control:	„mírně doleva“
Print Quality:	`Letter Quality`

Pro MS WINDOWS 3.0

V okně HP DeskJet Family on FILE:

Printer:	`HP DeskJet`
Paper Source:	`Paper Tray`
Paper Size:	`A4 210 x 297mm`
Softfont RAM:	`None`
Orientation:	`Portrait`
Graphics Resolution:	`300 dots per inch`
Font Cartridges (2 max):	`None`

Program **DJtoPCX** nejprve kreslí obrázek v paměti počítače. Potom ho zašifruje do formátu PCX a v našem případě zapíše do souboru `KNOT.PCX`. Dále vytvoří TEXový „input“ soubor, v našem případě soubor `KNOT.TEX`, kde jsou příkazy pro přesné začlenění obrázku v PlainTEXu.

Program **DJtoPCX** má dva nebo čtyři parametry. Prvým parametrem je jméno vstupního souboru, které uvedeme bez cesty a bez přípony. Jako

příponu souboru program automaticky vezme `DJ`. Druhým parametrem je zvětšení vyjádřené v počtu bodů na palec. Tedy pro základní velikost uvedeme 300, pro zvětšení 1,2-krát 360, pro 1,44-krát 432 atd. Další dva parametry nejsou povinné. Udávají počet mikrosloupců a mikrořádků odleva resp. odshora, které má program ignorovat, aby bílý okraj zbytečně nezaplňoval pracovní paměť programu.

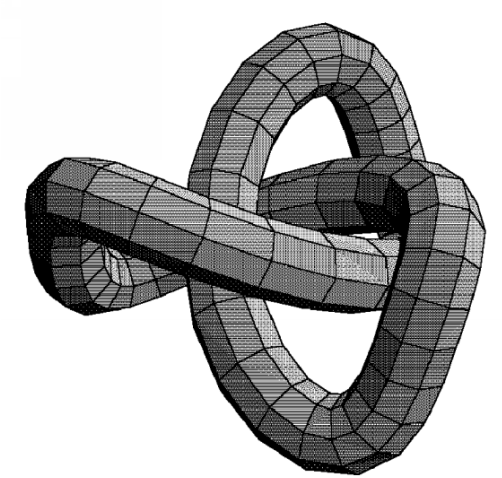
Chceme-li použít obrázek v základní velikosti, zadáme příkaz:

```
DJTOPCX KNOT 300
```

Tím jsme vytvořili dva nové soubory `KNOT.PCX` a `KNOT.TEX`, které musíme umístit do správných adresářů. Obrázek můžeme začlenit do textu např. příkazy:

```
\input KNOT
\setKNOT
```

Jak vypadá výsledek, si můžeme prohlédnout na obr. 2.



Obr. 2

Teoreticky by se mohlo stát, že by se nám příliš velký obrázek nevešel do pracovní oblasti programu **DJtoPCX** a že by ho tedy „kus chybělo“. Zde můžeme s výhodou využít toho, že program **DJtoPCX** dovoluje ignorovat určitý počet mikrosloupců odleva a určitý počet mikrořádků odshora, tedy vlastně posunout pracovní oblast. Jak toto posunutí zvolit

a další užitečné informace nám sdělí program **DJInfo**, doplněk k programu **DJtoPCX**.

Kdybychom si dodatečně rozmysleli, že budeme náš obrázek začleňovat do textu pro jiné zvětšení, bude mít táž „bitmapová“ realizace obrázku v souboru `KNOT.PCX` jakoby jiné rozměry, a tedy bychom potřebovali jiný soubor `KNOT.TEX`. Tento nový soubor můžeme vytvořit pomocným programem **PCXInfo**. Program **PCXInfo** má dva parametry. Prvým parametrem je jméno vstupního souboru, které uvedeme bez cesty a bez přípony. Jako příponu souboru program automaticky vezme `PCX`. Druhým parametrem je zvětšení vyjádřené v počtu bodů na palec. Chceme-li použít obrázek při zvětšení 1,2-krát, zadáme příkaz:

```
PCXInfo KNOT 360
```

Tím jsme vytvořili nový soubor `KNOT.TEX`. Je patrno, že má program **PCXInfo** své uplatnění i zcela nezávisle na programu **DJtoPCX**. Je možno jej použít jako pomůcku pro přesné začlenění libovolného souboru ve formátu PCX do textu.

Celý postup popsaný v této kapitole můžeme použít pro libovolný grafický program pod systémem WINDOWS, tedy např. i pro program **Paintbrush**. V tomto konkrétním případě však kvalita výsledného obrázku výrazně závisí na použité verzi WINDOWS.

Jak si programy opatřit

Starší verze sharewarového programu **PrintGL** se potulují po Praze a poslední verzi si můžete koupit u firmy Ravitz Software, Inc., P. O. Box 25068, Lexington, KY 40524-5068, USA za $50.

Programy **DJtoPCX**, **DJInfo** a **PCXInfo** včetně svých zdrojových tvarů v Turbo Pascalu 6.0 jsou v balíku `DJTOPCX.ZIP`, který bude pravděpodobně v Praze na MFF UK a v archivu CSTUGu na FEL. Upozorňuji, že se starší verzí systému WINDOWS umějí tyto programy pracovat až od své druhé verze. Autor prohlašuje, že nepřebírá odpovědnost za případné škody způsobené těmito nekomerčními programy.

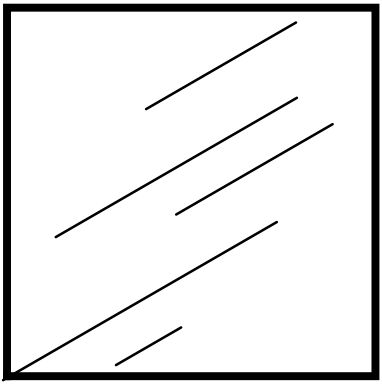
Petr Vejda
ÚE AV ČR
`uel@cspgas11.bitnet`

Nerovnoběžné rovnoběžky

Zdeněk Wagner

Tento příspěvek píšu na základě svých prvních zkušeností s METAFONTem. Přečetl jsem si pár článků v TEXbulletinech, zběžně jsem prolistoval METAFONTbook, udělal jsem několik úspěšných experimentů a pln elánu pustil jsem se do práce, která mě přivedla téměř k zoufalství. Tímto příspěvkem bych vás chtěl před takovým trápením ušetřit, neboť neúspěch může často být po důkladném prozkoumání úspěchem.

V učebnicích a příručkách o fotografii často najdeme porovnání zobrazení lidského oka a fotografického objektivu. Najdeme různé typy zrakových klamů, které jsou s oblibou označovány za vady oka. Tak tomu ovšem není. Nesmíme zapomínat, že oko bylo stvořeno k jiným účelům než fotografický objektiv a má proto jiné vlastnosti. Správně bychom měli říci, že oko vidí správně, ovšem jinak než objektiv kamery. Každý, kdo pracuje s výtvarnými objekty, by si měl být těchto zvláštností vědom, aby pak nebyl nepříjemně překvapen. Měl by je tudíž znát i ten, kdo pracuje s METAFONTem.



Obr. 1 Rovnoběžky vycházející z pěti zvolených bodů

Vraťme se ale zpět k mým pokusům. Ve svém obrázku jsem měl několik bodů, z nichž měly vycházet rovnoběžky obecně různé délky směrem vpravo šikmo nahoru. Prozatím si úhel rovnoběžek k vodorovné rovině označíme symbolem *alpha* a délku symbolem *len*. Příslušný vektor potom lze napsat jedním ze dvou způsobů: *right* **scaled** *len* **rotated** *alpha* nebo *right* **rotated** *alpha* **scaled** *len*. Rotace a „scaling“ jsou totiž komutativní. Předpokládejme, že již máme definovány body $z_1 \ldots z_5$[1])

[1]) Pro zlepšení čitelnosti budeme používat výhradně „matematickou notaci“ příkazů METAFONTu. Je ovšem důležité vědět, že do skutečného zdrojového souboru pro METAFONT musíme psát `z1` místo z_1 a `z[k+10]` místo z_{k+10}.

a délky $len_1 \ldots len_5$. Následující příkazy pak nakreslí rovnoběžky na obrázku 1.

$alpha = 30$;
for $k := 1$ **upto** 5 :
$z_{k+10} = right$ **scaled** len_k **rotated** $alpha$ **shifted** z_k;
draw z_k--z_{k+10};
endfor;

Vše jsme navíc orámovali, aby obrázek vypadal hezky. Sami se můžete přesvědčit, že přímky jsou skutečně rovnoběžné.

Nyní si připravíme makro, které využije výše uvedený koncept pro skutečný obrázek. Toto makro bude kreslit hranatý rám, který představuje kostru markýzy. Jeho definice je následující:

def $hranatyram =$
$z_1 = origin$; $z_2 = up$ **scaled** $15u$;
$z_3 = right$ **scaled** $13u$ **rotated** -10;
$z_4 = right$ **scaled** $35u$ **rotated** $slope$ **shifted** z_2;
$z_5 = right$ **scaled** $35u$ **rotated** $slope$ **shifted** z_3;
pickup $normalpen$;
draw z_4--z_2--z_1--z_3--z_5;
enddef;

přičemž již dříve jsme uvedli:

$slope = 3$;
pickup pencircle scaled $1pt$;
$normalpen := savepen$;

Obr. 2 Hranatý rám

Na obrázku 2 vidíme, jak to dopadlo. Přímky jsou stále ještě rovnoběžné. Můžeme tedy přistoupit k závěrečné etapě, k návrhu kompletního obrázku markýzy. Pro usnadnění práce si nejprve nadefinujeme parametrické proměnné, z nichž některé jsme již viděli:

pair $odstupvlnky$, $hloubkavlnky$;
$slope = 3$;
$odstupvlnky = down$ **scaled** $2.5u$;
$hloubkavlnky = down$ **scaled** $4.5u$;

$tv = 3$;

pickup pencircle scaled $1pt$;
$normalpen := savepen$;

pickup pencircle scaled $.4pt$;
$peronavlnky := savepen$;

Proměnné *peronavlnky*, *odstupvlnky* a *hloubkavlnky* signalizují, že budeme kreslit vlnky. Uděláme si proto makra. Možná se to zdá někomu zbytečné, ale ve skutečném souboru jsem kreslil více markýz, které měly určité znaky společné. Ostatně, definice maker je velmi užitečná praxe, neboť dobře navržené makro se může hodit i pro jiné fonty.

def $vlnkaodcary$(expr l, r, t, u) =
$(t_{l,r}$ **shifted** $odstupvlnky$)..tension tv and 1..((.5$[t, u]$)$[l, r]$ **shifted**
$hloubkavlnky$)..tension 1 and tv
enddef;

def $hranatavlnka$ (expr m) =
pickup $peronavlnky$;
draw $z1$--
for $i := 1$ **upto** m : $vlnkaodcary(z_1, z_3, (i-1)/m, i/m)$.. **endfor**
for $i := 1$ **upto** 8 : $vlnkaodcary(z_3, z_5, (i-1)/8, i/8)$.. **endfor**
z_5 **shifted** $odstupvlnky$--z_5;
enddef;

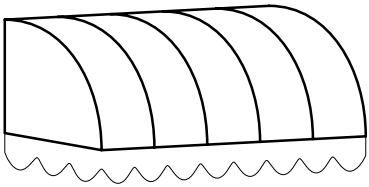
Program pro markýzu je potom:

beginchar („C“,50u#,18u#,8u#);
$hranatyram$;
for $t := 0$ **step** .2 **until** .9:
draw $t[z_2, z_4]\{right\}..\{down\}t[z_3, z_5]$;
endfor
draw $z_4\{right\}..\{down\}z_5$;
$hranatavlnka(3)$;
endchar;

Nyní se podíváme na obrázek 3. S hrůzou zjistíme, že se nám rovnoběžky rozbíhají. Příčina je jednoduchá, ale přiznám se, že bez rady kolegy Karla Horáka bych na to asi nepřišel.

Vlastně jsem to vysvětlil už v úvodu tohoto příspěvku. Lidské oko totiž vidí rovnoběžky tak, jako by se zdánlivě protínaly v úběžném bodě. Přidáme-li do obrazu perspektivu, pak všechny rovnoběžné linie musí směřovat do jednoho bodu. Tím ošidíme oko, takže se do dvojrozměrného obrazu dostane hloubka. Vizi hloubky vytvořila v obrázku markýzy čtvrtkruhová žebra, která byla nakreslena jako části elipsy. Geometrické rovnoběžky však oko vnímá jako rozbíhající se linie — a to je vlastně jádro problému.

Až tedy váš obrázek nebude vypadat tak, jak jste si představovali, uvědomte si, že příčina nemusí nutně spočívat ve špatném pochopení či použití příkazů METAFONTu. Než propadnete zoufalství, zkuste se zamyslet nad tím, zda porucha není způsobena nesprávnou interpretací perspektivy.



Obr. 3 Markýza s nerovnoběžnými rovnoběžkami

Zdeněk Wagner
wagner@csearn

Obsah nejnovějšího ročníku TUGboatu

TUGBOAT 13 (1) April 1992

TUGBOAT 13 (2) July 1992

TUGBOAT 13 (3) October 1992 1992 Annual Meeting Proceedings

TEX-Tagung DANTE '93 in Chemnitz (9. – 12.3. 1993)

Využili jsme s kolegou Petrem Sojkou z Brna laskavého pozvání německých přátel z DANTE a blízkosti východoněmeckého města Chemnitz (dříve Karl-Marx-Stadt, ještě dříve Chemnitz) a navštívili čtyřdenní setkání této nejsilnější evropské organizace uživatelů TEXu.

Program celého výtečně organizovaného setkání neskýtal příliš prostoru pro odpočinek (pro nás jedinou přestávku představovala dopolední členská schůze ve středu). K nejzajímavějším bodům bych zařadil tutoriál Jürgena Glöcknera o PostScriptových fontech a účastníkům pražského EuroTEXu dobře známý tutoriál pro „pokročilé“ Phila Taylora. Z přednášek bych vzpomněl zejména Glöcknerovo „Počítání s `\dimen` a `\count` registry při používání příkazu `\special`“, přednášku Friedhelma Sowy (autor programu BM2FONT „TGI — návrh standardu spolupráce TEXu s grafikou (TEX Graphics Interface)“, informaci Jense Pönische o používání TEXu pod operačním systémem Linux, což by měla být (znalci prominou) obdoba Unixu pro PC. Zajímavý byl i pohled Steffena Kernstocka na spolupráci TEXpertů s nakladateli „Nakladatelé

a TEX — o králících a hadech“. Phil Taylor zopakoval své vynikající expozé z Prahy na téma „Budoucnost TEXu“ a Rainer Schöpf se (jak jinak) zabýval projektem LATEX 3.

Ke společenskému programu patřila návštěva chemnitzké Nové radnice a podvečerní výlet do blízkého Augustusburgu, v jehož působivém hradu je rozsáhlá expozice historických motocyklů a kočárů.

Sdružení německy mluvících uživatelů TEXu má více než 2 000 členů a rozhodně netrpí finančními problémy jako americký TUG.

Karel Horák
horakk@csearn

Vydalo:	Československé sdružení uživatelů TEXu vlastním nákladem jako interní publikaci
Obálka:	Bohumil Bednář
Počet výtisků:	600
Tisk:	HBT-Jet PRESS Neratovice
Adresa:	CSTUG MÚ UK, Sokolovská 83, 186 00 Praha 8