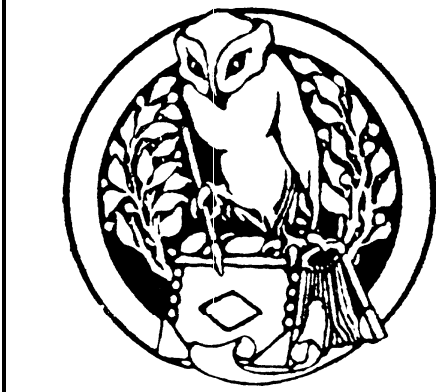
TEX METAFONT



4

ZPRAVODAJ 97

ČESKOSLOVENSKÉHO SDRUŽENÍ UŽIVATELŮ TEXU

ISSN 1211-6661 Ročník 7

OBSAH

Zpravodaj Československého sdružení uživatelů TEXu je vydáván v tištěné podobě a distribuován zdarma členům sdružení. Po uplynutí dvanácti měsíců od tištěného vydání je poskytován v elektronické podobě (PDF) ve veřejně přístupném archívu dostupném přes `http://www.cstug.cz`.

Své příspěvky do Zpravodaje můžete zasílat v elektronické podobě anonymním ftp na `ftp.icpf.cas.cz` do adresáře `/wagner/incoming/`, nejlépe jako jeden archivní soubor (`.zip, .arj, .tar.gz`). Současně zašlete elektronickou poštou upozornění na `mailto:bulletin@cstug.cz`. Uvedený adresář je pro vás „write/only“. Pokud nemáte přístup na Internet, můžete zaslat příspěvek na disketě na adresu:

Zdeněk Wagner
Vinohradská 114
130 00 Praha 3

Disketu formátujte nejlépe pro DOS, formát Macintosh 1.44 MB je též přijatelný. Nezapomeňte přiložit všechny soubory, které dokument načítá (s výjimkou standardních součástí CSTEXu), zejména v případě, kdy vás nelze kontaktovat e-mailem.

ISSN 1211-6661

The inaugural meeting of TUG India

Sebastian Rahtz

First stirrings

Back in the summer when I first started corresponding with C. V. Radhakrishnan in India about TEX and SGML-related matters, I little thought that I would be escaping the English winter for a week in Southern India at the start of 1998. But something seemed to crystallize in the minds of some Indian TEXies, and events moved fast in the subcontinent during the autumn. By November 16th, Radhakrishnan was able to announce to the world that the newest TEX user group had been born:

> The Indian TEX Users Group has been informally launched today at the academic premises of Department of Mathematics of University of Kerala, Trivandrum. Prof. KSS. Nambooripad, a world-renowned mathematician and an ace TEX programmer chaired the session. He was unanimously elected as the Chairman of the Indian TEX Users Group. Following are the office-bearers of the TUGIndia.

Chairman:	Prof. (Dr.) K. S. S. Nambooripad
Secretary:	C. V. Radhakrishnan
Treasurer:	Dr. R. Rajendra
Executive:	Dr. A. R. Rajan (University of Kerala)
	Dr. E. Krishnan (University College, Trivandrum)
	Dr. V. N. Krishnachandran (Vikram Sarabhai Space Center)
	Dr. R. K. Chettiar (Department of Education, Govt. of Kerala)
	Mr. C. V. Rajagopal (University Observatory)
	Mr. Deepak Tony Thomas, Oracle Corporation, Bangalore
	Dr. P. Rameshkumar (MG University, Kottayam)
	Dr. SRP. Nayar (Inter Univ. Center for Astronomy, Pune)

At the same time, they did me the great honour of inviting me to inaugurate the group, and I lost no time in accepting in principle. In the ordinary course of events it would have been beyond the finances of either TUG India or myself to pay for a trip there, but then a fairy godmother appeared, in the shape of the UK TEX Users Group. The committee considered my tentative suggestion, and agreed that support of such a potentially important group would be a reasonable

use of group funds. That just meant fixing a date, and finding a flight, and all was in train. By an amazing coincidence, another member of the UK TUG committee, Kaveh Bazargan, had already booked a holiday over Christmas and New Year at precisely the location in India chosen for the TUG India launch, so we were able to mount an even more impressive presence.

India, and the inauguration

The TUG India meeting took place in Trivandrum, the capital of Kerala state, which forms the southwest corner of India. It is a tropical area, which sometimes seems to be entirely covered in cocoanut palms and banana trees, and is famous for its Communist state government, its almost 100% literacy rate, and a general air of some prosperity and a good distribution of wealth.

The principle mistake I made before setting off was to contract a vile cold, which rendered me almost speechless during my first few days, and a poor picture of health for the whole stay. However, after a long flight from London, a sweaty wait in Mumbai, and then a short flight to Trivandrum, it was little hardship to be taken off after lunch to the excellent beach resort of Varkala. Talking TEX beneath the palms next to a sunny beach of the Arabian sea was a little disconcerting, but we managed....

On Monday January 5th, we gathered in the University Observatory (a purely courtesy title these days) for the opening ceremony of TUG India, at which Kaveh and I were joined as speakers by Professor Nambooribad, the group's chairman, the University Vice Chancellor, and the local member of Parliament (showing a healthy interest in IT matters). Kaveh and I tried to present the TEX world as place of dynamism, excitement and new possibilities for conventional and electronic publishing, and some at least of our audience seemed convinced.

Many of the delegates were from typesetting companies (some of them suppliers to my own employer, Elsevier), with the biggest contingent from Madras — Thomson's office seemed to have sent almost all their R&D team. But there was plenty of academic interest too, and of course a special concern with typesetting Indian scripts.

It was a pleasure to be able to hand over to Radhakrishnan a selection of TEX-related books and journals, donated by Jonathan Fine, Malcolm Clark and myself, and to confirm the imminent despatch of back issues of TUGboat and MAPS to India. NTG had already sent a generous batch of 4allTEX CD-ROMs, of which each delegate was given a copy.

The first TUG India courses

On January 6th the serious work started, four days of tutorials in the morning on 'advanced' topics, and introductory LaTeX in the afternoons. I managed to avoid teaching the latter (I always find myself *apologizing* too much for LaTeX), but had fun in the mornings.

We started by dealing with a subject dear to my heart, and to that of some of the delegates — LaTeX to SGML translation. I expounded the Elsevier system, based on four stages of transformation:

1. LaTeX to dvi, using a very specialized class file, which redefines almost everything to put SGML markup into the dvi file;
2. dvi to ASCII (using Tobin's dtl programs);
3. ASCII to SGML against an intermediate DTD;
4. SGML to SGML for the final DTD (using a Perl library with directly interfaces with the NSGMLS parser).

It turned out that at least two others present had also thought of similar methodologies, which was reassuring.

From LaTeX to SGML, I moved on next day to DSSSL (Document Style Semantics and Specification Language) and its relationship with TeX — perhaps not everyone present quite went along with me on that one. We were on safer ground discussing general aspects of electronic publishing using TeX, and I was glad to able to be describe pdfTeX in some detail, to publicize what I consider a much under-rated alternative to LaTeX2html (Eitan Gurari's TeX4ht), as well as give a puff for my own LaTeX hyperref package.

One the third day, we moved onto pictures, and I attempted to make a (rather shaky) case for MetaPost. Colour was a subject where it was easier to find common ground, albeit by agreeing that color separation specification in TeX was much too immature at present.

For the last day, I had decided that this was the moment where I would really make a first go at using Omega, and (somewhat to my surprise), I was able to write, compile and use a one-line Omega Transformation Process after some study of Omega examples. Since one of the Omega authors (Yannis Haralambous) is very actively working on the necessary OTPs, hyphenation and so on for typesetting Malayalam (the language spoken in Kerala), we can expect rapid deployment of Omega amongst those typesettings things like school textbooks.

. . . and some sightseeing

After talking TeX for 5 days, I was ready for some relaxation. We started with a shopping expedition, during which I bought some dresses for my daughters which are certain to lighten up wintry Oxford, and a selection of South

Indian classical music. Then on the Saturday we drove across the state line into Tamil Nadu to visit the Padmanabhapuram palace of the Maharajah of Travancore, the princely state which occupied much of what is now Kerala until Independence. In the late 18th century, a replacement palace was constructed in Trivandrum, and Padmanabhapuram was left untouched. With elements from the 16th century, it is an incredible structure built almost entirely of teak, often intricately carved, and all ingeniously designed to keep the rooms cool with natural air-conditioning. Whether it was the ladies bathing tank, the audience chamber, or the hall where 2000 Brahmins could be entertained to dinner, the whole place was a marvel of design — and preservation by the State Archaeological Service! Perhaps the best moment was when we were granted special access to view the Maharajah's private meditation apartment whose plaster walls were covered in marvellous paintings, and where a pair of cocoanut-oil lamps had been burning non-stop for 200 years.

From the past to the present, as we drove to Cape Comorin, the southern-most tip of India, where you can see both sunrise and sunset across the sea from the same spot, and where three oceans meet. Here, in the late 19th century, Swami Vivekananda (a very influential religious reformer) swam out to a bare rock in the sea, meditated for five days, and achieved a state of enlightenment to accord him the status of a saint. Now there is a modern memorial on the rock, and we joined hundreds of pilgrims in the boat ride to examine the spot. Thence back north, trying to visit a Jain temple set deep in a cave, but sadly the gates were locked, and some monkeys laughed from the rock.

On the Sunday, to Kerala's secret paradise, the long salt waterway that runs for 200 km parallel to the sea, sometimes as a wide as a lake, at other times turning into quiet green tunnels with barely enough depth for the boat. A vista of endless cocoanut palms, half-hidden houses, and small fishing boats provided a very relaxing boat trip.

Conclusion

This was a worthwhile, if exhausting, trip, and I hope it gave a good start to TUG India. When I left, they already had 79 members signed up, just from word of mouth, so the group looks set to be active. It is hoped to cycle the meetings around the different parts of India, as well as publishing a newsletter, so the current bias towards the south should soon be corrected.

I must, of course, take this space to extend the heartfelt thanks of Kaveh and myself to C. V. Radhakrishnan and the many others who looked after us so magnifcently during our stay in Kerala. They were very worthy ambassadors

of a lovely part of India. I look foward to working with them, and hopefully to visiting India again soon.

TUGIndia can be contacted as follows:

C K Radhakrishnan
Secretary
TUGIndia
Kripa, TC 24/548, Sastha Gardens
Thycaud, Trivandrum 695014, India

Tel. +91 471 324341
Fax. +91 471 333186

Email: tugindia@mailexcite.com

Sebastian Rahtz
Elsevier Science Ltd
Oxford
s.rahtz@elsevier.co.uk

TEX a sazba zrcadlových překladů

Arnošt Štědrý

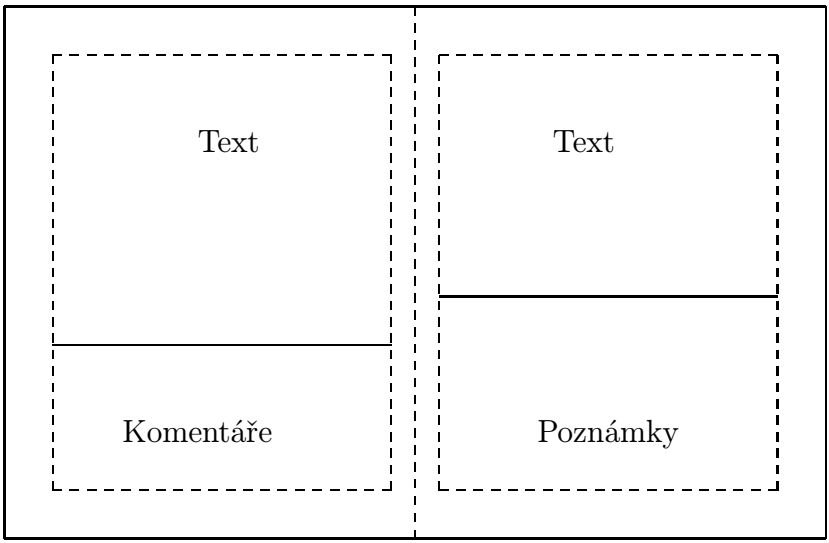
> *`\read`. Tento přístup k souborům se v TEXovských úlohách používá velmi zřídka.* Petr Olšák: TEXbook naruby

Článek vychází z osobních zkušeností autora při návrhu a realizaci TEXovských maker pro tzv. zrcadlové překlady. Navržená makra kopírují francouzské vzory lingvistických edic a byla použita pro sazby řecko-české bilingvy.

Vymezení problému

Z hlediska náročnosti je sazba takzvaných zrcadlových překladů jedna z nejtěžších. Podíváme-li se na takovýto dokument, zjistíme, že je většinou strukturován do protilehlých dvojstran, kde každá stránka je navíc rozdělena na textovou a poznámkovou část. Hlavní požadavek na textovou část je, aby zdrojový text i jeho

překlad byly synchronizovány, tj. zastoupeny na dvojstránce významově stejnou měrou. Poznámková část je obvykle rozdělena na komentáře k původnímu textu a překladatelské poznámky. Důraz je kladen na konzistentnost, tj. aby se poznámky a komentáře zmiňované v textové části vyskytovaly v plném znění na příslušné dvojstraně. Oproti tomu je dovoleno přelévat poznámky a komentáře mezi sebou, tj. je-li více komentářů, je možné jejich část přesunout k poznámkám a obráceně. Pro větší přehlednost zavádíme rozdílné číslování komentářů a poznámek (např. arabskými čísly a písmeny). Lingvistický uzus navíc vyžaduje vynulování číslování po každé dvojstraně. Obě textové části jsou vybaveny marginálním číslováním většinou odkazujícím na řádek v původním (paleografickém) vydání.



Obrázek 1: Dvojstránka a její rozdělění

Hlavním problémem, který bude nutno řešit, je rozdělení do dvojstran, což je ovšem věc, kterou lze jen obtížně provádět strojově. Lze totiž jen obtížně stanovit odpovídající si místa v původním textu a v překladu, zvláště proto, že místo vhodného zlomu se může nacházet uvnitř odstavce. Sazbu bude tedy nutno zhotovovat nejlépe přímo s autorem překladu (žije-li), případně s vlastní velkou erudicí. Osobně doporučuji první řešení, poněvadž zásahy do textu[1] jsou mnohdy nezbytné.

Text samotný je nejvhodnější rozčlenit do dvou souborů obsahujících původní

[1] Zde je míněn zejména text poznámkového aparátu.

text a překlad. Poznámky, komentáře a marginální číslování do nich včleníme pomocí vhodných značek.[2]

Vlastní stránkové zlomy vyznačíme v obou souborech nějakou vhodnou posloupností písmen která se určitě v žádném z obou textů neobjeví (používám qqq).

Protože následující navržená makra byla použita k sazbě řecko-české bilingvy, budu bez újmy na obecnosti používat označení „řecký text“ a „český text“. Stejně tak komentáře a poznámky budeme dále většinou souhrnně nazývat poznámkami pod čarou. Patřičně pozměněná makra byla použita i pro sazbu trilingvy hebrejsko-řecko-české.

TEXnické provedení

Je jasné, že hlavní roli zde bude hrát makro `\read`, pomocí kterého načteme stránky řeckého a českého textu do příslušných token registrů, odkud následně vysázíme dvojstranu. Je třeba si uvědomit několik omezujících vlastností makra `\read`, s nimiž musíme počítat při návrhu našeho balíku i při pořizování textu. Syntaxe příkazu je:

```
\read<number> to <control sequence>
```

kde `<number>` je číslo souboru deklarované příkazy:

```
\newread\soubor
\openin\soubor=jmeno.souboru
```

Primitiv `\read` se chová poněkud komplikovaně. Nejprve načte z požadovaného souboru řádek. Pokud je řádek již balancován (tj. obsahuje odpovídající si { a }) převede ho do posloupnosti tokenů. Pak definuje `<control sequence>` jako tuto posloupnost. Pokud řádek nebyl balancován, čte další řádky do té doby, než takovýto stav nastane.

Naším úmyslem bude číst řádky souboru s řeckým textem dokud nenarazíme na ukončovací posloupnost písmen. Načtené řádky uchováme v token registru `\reckytxt` a totéž provedeme s českým textem, pouze k jeho uložení použijeme token registr `\ceskytxt`. Nejprve tedy nezbytné deklarace:

```
1    \newif\ifdalsi \dalsitrue
2    \newtoks\pomocny
3    \newtoks\reckytxt
4    \newtoks\ceskytxt
5    \newread\reckytext
6    \newread\ceskytext
7    \def\konecstr{qqq }
```

[2] Protože existuje mnoho maker na tvorbu marginálního číslování, nezahrnul jsem z důvodu přehlednosti kódu patřičné definice.

Dále definujme makra na pohodlné otvírání a zavírání souborů. Pro řecký text použijeme file handler `\reckytext`, obdobně `\ceskytext` pro soubor s českým překladem.

```
8   \def\openrecko#1{\openin\reckytext=#1}
9   \def\opencesko#1{\openin\ceskytext=#1}
10  \def\zavri{\closein\reckytext\closein\ceskytext}
```

Nyní definujme makra, která přečtou část souboru vymezenou proměnnou `\konecstr` do příslušných token registrů:

```
11  \def\cticesko{\dalsitrue
12  \loop\global\read\ceskytext to \pomocna
13  \ifx\pomocna\konecstr\dalsifalse
14  \else
15  \pomocny\expandafter{\pomocna}
16  \edef\pomoc{\the\ceskytxt\the\pomocny}
17  \ceskytxt\expandafter{\pomoc}
18  \fi\ifdalsi\repeat}
19  \def\ctirecko{\dalsitrue
20  \loop\global\read\reckytext to \pomocna
21  \ifx\pomocna\konecstr\dalsifalse
22  \else
23  \pomocny\expandafter{\pomocna}
24  \edef\pomoc{\the\reckytxt\the\pomocny}
25  \reckytxt\expandafter{\pomoc}
26  \fi\ifdalsi\repeat}
```

Všimněme si řádků 15 až 17. Ve starším návrhu místo nich bylo pouze:

```
\edef\pomoc{\the\ceskytxt \pomocna}
\ceskytxt\expandafter{\pomoc}
```

což způsobilo nevhodnou expanzi makra `\pomocna` (obsahující právě čtený řádek) už v době plnění token registru `\ceskytxt` a vedlo např. ke komplikovanému zacházení s poznámkami pod čarou. Takto můžeme expanzi použitých maker pozdržet až do doby vlastní sazby.

V otázce výstupní rutiny máme práci zjednodušenou, protože veškerý načtený materiál bude i vytištěn. Výstupní rutina nejprve expanduje do jednoho vboxu token registr `\reckytxt`, do druhého vboxu registr `\ceskytxt`. Během expanze obou tokenů se poznámky a komentáře vysází do třetího vboxu, který vhodným vsplitem rozdělíme na dva. Makro `\pozn` bude sloužit k označení poznámek i komentářů:[3]

```
27  \newcounter{pozn}
28  \setcounter{pozn}{1}
```

[3] Omlouvám se všem puristům za míchání LaTeXového kódu s čistým plainem, ale zkušení plainisté si nepatřičná makra jistě opraví.

```
29  \long\def\pozn#1{
30  \unskip$^{\rm \thepozn}}$\global\setbox\poznamky
31  \vbox{\hsize=11cm\unvbox\poznamky
32  {$^{\rm \thepozn}$  #1}
33  }\addtocounter{pozn}{1}}
```

Před vlastní expanzí makra `\pozn` bude nutno vhodně definovat `\thepozn`.

```
34  \newbox\leva
35  \newbox\prava
36  \newbox\poznamky
37  \newdimen\zbyva
38  \def\out{
39  \def\thepozn{\arabic{pozn}}\setcounter{pozn}{1}
40  \setbox\leva\vbox{\hsize=11cm \greek \the\reckytxt}
41  \def\thepozn{\roman{pozn}}
42  \setbox\prava\vbox{\hsize=11cm \the\ceskytxt}
43  \nastav
44  \vbox to 18.7cm{
45  \box\leva\vglue .5cm plus 1fill\vsplit\poznamky to
46  \zbyva}
47  \newpage
48  \vbox to 18.7cm{
49  \box\prava\vglue .5cm plus 1fill\box\poznamky}
50  \newpage
51  \ceskytxt{}
52  \reckytxt{}
53  \setcounter{pozn}{1}}
54  \def\nastav{\zbyva=18cm\advance\zbyva by
55  -\ht\leva\advance\zbyva by -.5cm}
56  \def\ctijeden{\ctirecko\cticesko\out}
```

V kódu je několik neobratností, které lze však snadno odstranit. Za prvé „magická“ čísla 18.7, .5 a 18 a 11, šla jistě nahradit registry. Rovněž primitivní volání stránkové rutiny pomocí `\newpage` na řádcích 48 a 51 je snadno napravitelné. Jedná se koneckonců o pouhý prototyp.

Použití

Předpokládejme, že vlastní text je rozčleněn do kapitol. Vhodným umístěním rozdělovacích sekvencí `qqq` nahrubo rozčleníme text do stránek a vzhled stran vyladíme. Přitom je nutno míti stále na paměti specifika primitivu `\read` a vyvarovat se nebalancovaných závorek {, }. To vede například k nutnosti v některých případech nepoužívat makro `\uv` k sazbě českých uvozovek (použijeme

῾Ερμοῦ πρὸς Τὰτ υἱόν·
ὅτι ἀφανὴς θεὸς φανερώτατός ἐστιν.

Καὶ τόνδε σοι τὸν λόγον, ὦ Τάτ, διεξελεύσομαι, ὅπως μὴ ἀμύητος ᾖς τοῦ κρείττονος θεοῦ ὀνόματος. σὺ δὲ νόει πῶς τὸ δοκοῦν τοῖς πολλοῖς ἀφανὲς φανερώτατόν σοι γενήσεται. οὐ γὰρ ἂν ἦν εἰ ἀφανὲς ἦν·[1] πᾶν γὰρ τὸ φαινόμενον γεννητόν· ἐφάνη γάρ· τὸ δὲ ἀφανὲς ἀεί ἐστι· τοῦ γὰρ φανῆναι οὐ χρήζει· ἀεὶ γάρ ἐστι καὶ τὰ ἄλλα πάντα φανερὰ ποιεῖ, αὐτὸς ἀφανὴς ὤν, ὡς ἀεὶ ὢν. φανερῶν αὐτὸς οὐ φανεροῦται, οὐκ αὐτὸς γεννώμενος, ἐν φαντασίᾳ δὲ πάντα φαντασιῶν. ἡ γὰρ φαντασία μόνων τῶν γεννητῶν ἐστίν. οὐδὲν γάρ ἐστιν ἡ φαντασία ἢ γένεσις.

ὁ δὲ εἷς ἀγέννητος δηλονότι καὶ ἀφαντασίαστος καὶ ἀφανής, τὰ δὲ πάντα φαντασιῶν διὰ πάντων φαίνεται, καὶ ἐν πᾶσι, καὶ μάλιστα οἷς ἂν αὐτὸς βουληθῇ φανῆναι. σὺ οὖν, ὦ τέκνον Τάτ, εὔξαι πρῶτον τῷ κυρίῳ καὶ πατρὶ καὶ μόνῳ καὶ οὐχ ἑνί, ἀλλ᾽ ἀφ᾽ οὗ ὁ εἷς, ἵλεω τυχεῖν, ἵνα δυνηθῇς τὸν τηλικοῦτον θεὸν νοῆσαι, κἂν ἀκτῖνά σοι κἂν μίαν αὐτοῦ τῇ σῇ διανοίᾳ ἐκλάμψαι. νόησις γὰρ μόνη ὁρᾷ τὸ ἀφανές, ὡς καὶ αὐτὴ ἀφανὴς οὖσα. εἰ δύνασαι, τοῖς τοῦ νοῦ ὀφθαλμοῖς φανήσεται, ὦ Τάτ· ἄφθονος γὰρ ὁ κύριος φαίνεται διὰ παντὸς τοῦ κόσμου. νόησιν ἰδεῖν καὶ λαβέσθαι αὐταῖς ταῖς χερσὶ δύνασαι καὶ τὴν εἰκόνα τοῦ θεοῦ θεάσασθαι; εἰ δὲ καὶ τὸ ἐν σοὶ ἀφανές ἐστί σοι, πῶς αὐτὸς ἐν ἑαυτῷ[2] διὰ τῶν ὀφθαλμῶν σοι φανήσεται;

[1] Nock opravuje na οὐ γὰρ ἂν ἦν ⟨ἀεὶ⟩ εἰ ⟨μὴ⟩ ἀφανὲς ἦν.

[2] Scottova a Nockova konjektura. Rukopisy mají ἑαυτὸν ἐν σαυτῷ, což je gramaticky nemožné, neboť věta by pak musela mít dva podměty. Turnebus navrhuje αὐτὸς ἐν σαυτῷ.

[a] Doslova: … *Boha, který je mocnější jména.*

[b] „Zjevování v představivosti“ je naším pokusem o překlad řeckého *fantasia*, které neznačí jen naši „fantazii“ a latinské *imaginatio*, nýbrž též (doslovně) „schopnost zjevovat“.

[c] „Mysl“ je zde *dianoia*, nikoli *nús*. Význam je však nejspíše podobný.

Hermés ke svému synu Tatovi o tom, že Nezjevný Bůh je nejzjevnější

Také tuto řeč přednesu tobě, Tate, abys nezůstal nezasvěcený do mystérií Boha, který je příliš mocný, než aby mohl být pojmenován.[a] Uvědom si tedy, jak se ti to, co se mnohým zdá nezjevné, stane nejzjevnější. Vždyť by to vůbec nebylo, kdyby to bylo nezjevné. Každý jev je totiž zplozený, neboť se zjevil. Nezjevné jest však vždy, nepotřebuje se zjevovat. Je stálé a činí všechno ostatní zjevným. On sám je nezjevný, neboť jest vždy. Ač zjevující, sám se nezjevuje. Sám je nezplozený, všechno však představuje v představivosti. Vždyť zjevování v představivosti se týká pouze zplozených věcí. Zjevování v představivosti[b] totiž není ničím jiným než zrozením.

Ten Jeden nezplozený je samozřejmě nepředstavitelný i nezjevný, všechno však představuje a skrze vše a ve všem se zjevuje — a zejména pak těm, kterým se rozhodl se zjevit. Ty se tedy, můj synu Tate, nejprve modli k Pánu a Otci, který je Jediný, avšak není Jeden, nýbrž Jeden je od Něho, aby se ti dostalo Jeho milosti a byl jsi schopen poznat tak velikého Boha, a aby ti dal ve tvé mysli[c] zazářit třeba i jedinému ze svých paprsků. Vždyť jedině myšlení spatřuje nezjevné, neboť je samo nezjevné. Pokud jsi toho schopen, zjeví se tvým očím Mysli. Pán není skoupý:[d] zjevuje se skrze všechno v Kosmu[e]. Můžeš snad spatřit své myšlení, chopit se jej svýma rukama a nazřít obraz Boha? Pokud je ti nezjevné i to, co je v tobě, jak se ti bude skrze tvé oči zjevovat On sám v sobě samém?[f]

[d] „Není skoupý“ = je *afthonos*, doslova tedy „je bez závisti (*fthonos*), není nepřejícný“.

[e] Doslova „skrze celý“ nebo „skrze veškerý Kosmos“.

[f] Rukopisný text je gramaticky nesmyslný a překládáme proto podle konjektury (viz aparát). S jinou a menší opravou by znění mohlo být: *... jak se ti bude v tobě samém skrze tvé oči zjevovat On sám?*

místo toho `\clqq` a`\crqq`). Stejně tak bude potřeba na některých místech upravit zvýrazňování jiným typem písma (s ohledem na párovost závorek). Též bude třeba pozorně zacházet s odstavci, a někdy použít `\noindent`. Přelévání poznámek a komentářů lze řídit vhodnými příkazy `\vglue` na konci stránky s řeckým textem. Zpracování obou souborů řídíme z hlavního souboru sekvencí příkazů:

```
\openrecko{ch3grk.tex}
\opencesko{ch3.tex}
\ctijeden
\ctijeden
\ctijeden
\zavri
```

Poznámky

V návrhu jsem vycházel z francouzské tradice zrcadlových překladů. Vynechal jsem pouze zvyk vyplňovat stránku zvětšováním mezi řádkových mezer (i když by to nebyl problém) a nahradil, dle mého názoru, ohavný font, použitý v edici, jiným. Zde předvedená makra jsou jen funkční podmnožinou balíku, který ve skutečnosti používám. Cílem bylo demonstrovat možnosti, které TEX přináší sazbě zrcadlových překladů. Každý si jistě navržený jednoduchý postup předělá k obrazu svému, pokud k problému nepřistoupí od základu jinak. Vlastní sazba asi sto dvaceti stran dokumentu trvala dvanáct hodin, spolu s kontrolou správného dělení a odstraňováním některých nepatřičností použitého řeckého fontu. Na některých místech bylo nutno pozměnit text poznámkového aparátu.

Vlastní realizace celého díla probíhala ve třech fázích. Nejprve se řecký originál převedl awkovským skriptem z betakodu (tradiční způsob zápisu řeckých textů) do RTF. Autor překladu a komentářů dále pracoval v MS Wordu (s patřičnou nadstavbou pro práci s řečtinou). Následoval převod z RTF do TEXu veškeré další úpravy se prováděly již v TEXovských souborech.

Ukázka na stránkách 182 a 183 je zde publikována s laskavým svolením pana Radka Chlupa.

Štrukturované editovanie

Janka Chlebíková

Elektronická forma dokumentu poskytuje priestor pre vytváranie rôznych aplikácií postavených nad novou formou dokumentu. Textové systémy,[1] ktoré okrem vytvárania dokumentov sú primárne určené pre výstup dokumentov na tlač (tlačiareň, osvit), sú jednou z najprirodzenejších aplikácií s elektronickými dokumentami. Inou aplikáciou nad elektronickými dokumentami môže byť použitie databázových systémov, ktoré budú spracovávať niektoré časti dokumentu. Príkladom môže byť vytváranie zoznamu autorov z nejakej kolekcie matematických článkov.

Pre vytváranie väčšiny takýchto aplikácií je dôležitá znalosť vnútornej štruktúry dokumentu a nie jeho prezentačných vlastnosti, ktoré sú dôležité len pre vizualizáciu dokumentu (na papieri alebo obrazovke počítača). Ak si ako príklad dokumentov zoberieme opäť matematické články, potom pre databázovú aplikáciu nie je podstatná prezentačná forma, akou je autor zobrazený (veľkosť a typ fontu, centrovanie, ...), ale značky označujúce v dokumente miesto, kde začína a končí autorove meno.

Špeciálne a všeobecné značkovanie

Problém v súčastnosti rozšírených WYSIWYG editorov (napr. Microsoft Word) je, že používajú na uchovávanie dokumentu formát, ktorý obsahuje len prezentačné značky. Tieto prezentačné značky sú navyše zrozumiteľné len textovému systému, v ktorom bol dokument vytvorený. Z tohto dôvodu sa takéto značkovanie nazýva *špeciálnym značkovaním* (special markup). Dokument v takomto formáte je prakticky nepoužiteľný pre inú aplikáciu, okrem jeho vytlačenia daným textovým systémom. Nevýhodou takéhoto prístupu je aj ťažko realizovateľný preklad do formátov používaných inými textovými systémami, resp. aplikáciami. A tak napríklad i dokumenty napísané v Microsoft Worde (obľúbenom zvlášť antiTEXistami) sú pre ďalšie aplikácie len ťažko použiteľné.

Ak chceme elektronický dokument používať aj iným spôsobom (ako len pre tlač), výhodnejšie je dokument vytvoriť v textovom systéme, ktorý využíva formát tzv. *všeobecného značkovania* (general markup). V tomto formáte si textový systém uchováva dokument spolu s označením logických prvkov, ktoré spolu vytvárajú štruktúru dokumentu. Prezentácia dokumentu je oddelená od obsahu

[1] Tento pojem zahŕňa WYSIWYG interaktívne textové editory, i neinteraktívne textové formátovacie systémy typu TEX.

dokumentu a je zabezpečovaná priamo textovým systémom (nie autorom!). Pre textový systém značky logických prvkov predstavujú volanie makier (preto niekedy aj procedurálne značkovanie), ktoré zabezpečia prezentáciu príslušného logického prvku v danom textovom systéme. Makrá na prezentáciu sú uložené nezávisle od dokumentu. Pre jeden typ dokumentu (zodpovedajúci nejakej štruktúre dokumentu, napr. 'letter') môže byť k dispozícii niekoľko súborov makier, ktoré určujú rôznu prezentáciu logických prvkov. Tieto podľa potreby umožňujú dokument vizualizovať v rôznych prezentáciach.

Pre autora dokumentu má oddelenie prezentácie od obsahu dokumentu veľký význam v tom, že sa nemusí starať o typografickú stránku dokumentu (tá patrí do rúk odborníkom), ale môže svoju energiu sústrediť na obsah dokumentu.

Pre takto jednotne označkované dokumenty daného typu nie je problém vytvoriť žiadanú aplikáciu.

Nespornou výhodou je aj otvorenosť takéhoto prístupu, t.j. ľahšia prenositeľnosť dokumentov medzi textovými systémami, resp. aplikáciami — v ideálnom prípade ide len o zmenu názvov logických prvkov.

Ako príklad uveďme spomínaný matematický článok, ktorý sa skladá z nasledujúcich logických prvkov: názov článku, autor, autorova adresa, klasifikácia, kľúčové slová, abstrakt, ... Systémom používajúcim všeobecné značkovanie je TEX, presnejšie LATEXovské štýly (resp. štýl „amsppt“ z AmSTEXu). Dokument napísaný v LATEXovskom štýle 'article' obsahuje jasne oddelené logické prvky pre názov článku, autora, adresu autora, kľúčové slová a ďalšie logické prvky.

Textové systémy, ktoré používajú všeobecné značkovanie, však nie vždy dôsledne „nútia“ autora dokumentu používať značky oddeľujúce logické prvky. Autor dokumentu môže dodávať vlastnú prezentáciu logickým prvkov a tým nie je nútený používať všeobecné značkovanie. Dokument teda „pekne vyzerá“ (aspoň podľa autora), ale je ďalej nepoužiteľný.

SGML

Prirodzeným vývojovým krokom vo všeobecnom značkovaní bola jeho ďalšia formalizácia. Formalizáciou sa dosiahla úplna nezávislosť všeobecného značkovania od formátu, ktorý používajú existujúce textové systémy na uchovávanie dokumentov. Táto formalizácia viedla k prijatiu ISO normy SGML (Standard Generalized Markup Language) definovanej v ISO 8879.

SGML je metajazyk poskytujúci prostriedky pre vytváranie DTD (Document Type Definition). DTD popisuje, z akých logických prvkov sa daný typ dokumentu skladá, v akom sú vzájomnom vzťahu (napríklad poradie prvkov v štruktúre) a čo je obsahom jednotlivých prvkov (iný logický prvok alebo samotná časť dokumentu). Formalizmus popisu DTD zaručuje jednoznačnosť zápisu takýchto značiek a tým ľahkú prenositeľnosť medzi rôznymi textovými systémami

alebo počítačovými architektúrami. SGML tiež poskytuje normalizované spôsoby reprezentácie špeciálnych znakov a symbolov použitím len ASCII znakov dostupných na bežných klávesniciach. Podstatné je, že SGML dokument neobsahuje žiadne[2] prezentačné značky.

Najrozšírenejším DTD je jazyk HTML (Hypertext Markup Language), známy z WWW (World Wide Web). Príkladom aplikácie postavenej nad HTML dokumentami sú Netscape alebo Mosaic na prezeranie takýchto dokumentov. Dôležité je uvedomiť si, že jeden a ten istý dokument nie je zobrazovaný úplne identicky v obidvoch aplikáciach preto, lebo každá zo spomínaných aplikácií priraďuje odlišné prezentačné vlastnosti logickým prvkom (nadpisom, zoznamom položiek, ...).

Niektoré veľké vydavateľstvá, napr. Elsevier prechádzajú na SGML formát pre uchovávanie svojich publikácií. Tiež Americká fyzikálna spoločnosť mení svoj publikačný systém na SGML. V súčasnosti všetky informácie a vývoj okolo SGML je možné sledovať aj na Internete, ako štartovný bod možno doporučiť: `http://www.sil.org/sgml/sgml.html`.

Výhody štrukturovaného prístupu

Na pohodlné vytváranie štrukturovaných dokumentov slúžia (v súčasnosti čoraz populárnejšie) štruktúrované editory, ktoré by mali WYSIWYG prístupom umožňovať autorovi naeditovať štrukturovaný dokument bez akýchkoľvek autorových znalostí o všeobecnom značkovaní. Veľmi špeciálnym prípadom takýchto editorov sú rozšírené HTML editory, ktoré ovšem nie vždy prehľadným spôsobom pracujú so štruktúrou HTML dokumentu. Navyše tieto editory majú HTML štruktúru obsiahnutú v svojom jadre (užívateľ nemá možnosť ju modifikovať) a tak pri akejkoľvek oficiálne prijatej zmene v HTML jazyku sú nutné zmeny v programe.

Dôležité je poznamenať, že nie nutne každý štrukturovaný editor musí pracovať priamo so SGML. Existujú aj iné jazyky na popis štruktúry dokumentu. Príkladom môže byť S-jazyk z projektu Opera, o ktorom je možno nájsť viac informácií na internetovskej adrese `http://opera.inrialpes.fr/OPERA/Thot.en.html`. SGML[3] je ale prijatou normou a preto všetky štrukturované editory, ktoré chcú byť otvorené pre iné aplikácie, musia s ňou spolupracovať. Z tohto dôvodu používame SGML terminológiu pri popise štruktúry dokumentu.

[2] Výnimku tvoria tzv. 'processing instructions', pomocou ktorých si môže textový systém vsúvať do dokumentu prezentačné informácie – napríklad miesta stránkových zlomov. Tieto sú štandardným spôsobom označené a inými aplikáciami ignorované.

[3] V poslednej dobe sa viac používa spojenie SGML/XML. XML je podmnožinou SGML a jeho cieľom je štandardizovaným spôsobom rozšíriť možnosti publikovania rôznych typov dokumentov na WWW, napr. matematických článkov.

Pozrime sa teraz bližšie na jeden konkrétny štrukturovaný SGML editor a výhody, ktoré vyplývajú z štrukturovaného prístupu k dokumentu. Editor GRIF bol navrhnutý Vincentom Quintom a ďalej vyvíjaný francúzskou firmou GRIF. Pre jeho flexibilitu a možnosť pridávať rôzne aplikácie nad dokument (resp. jeho logické prvky) tvorí tento editor jadro Euromath systému, o ktorom bližšie pojednáva nasledujúca kapitola.

GRIF podporuje niektoré základné DTD (odpovedajúce LaTeXovým štýlom ako 'article', 'letter', 'slide') a umožňuje užívateľovi pridávať vlastné DTD. Základné editovacie funkcie má GRIF rovnaké ako bežné WYSIWYG textové systémy (kontrola pravopisu, vkladanie obázkov, ...) Značky pre jednotlivé logické prvky sú pred užívateľom schované a tak pri jednoduchom editovaní dokumentu užívateľ ani nepostrehne, že pracuje so štrukturovaným editorom. Znalosť štruktúry dokumentu však GRIFu pridáva nový rozmer:

- *Štruktúra a prezentácia dokumentu sú dané, autor sa stará len o obsah dokumentu.* Logické prvky dokumentu spolu s ich vizuálnymi vlastnosťami sú definované a autor iba dopĺňa obsah jednotlivých prvkov. Autor nemusí mať žiadne znalosti o DTD, editor riadi pridávanie nových prvkov a nedovolí vykonať zmenu v logických prvkoch, ktorá by porušila štruktúru dokumentu (samozrejme vždy podľa odpovedajúceho DTD).
- *Pre každé DTD môže existovať niekoľko odlišných prezentácií.* Ako príklad uveďme dokument typu 'letter', ktorý môže byť zmenený prezentáciou na fax, súkromný list alebo memorandum. Podstatné je, že máme fyzicky len jeden dokument, v ktorom prevádzame všetky obsahové zmeny. GRIF poskytuje aj vlastný jazyk (tzv. P-jazyk) pre pridávanie vlastných prezentácií.
- *Jednotlivé logické prvky môžu byť zobrazené vo viacerých oknách.* V prípade editovania dokumentu napríklad typu 'article' je výhodná možnosť paralelného editovania literatúry vo vedľajšom okne.
- *'Cross-referencie' na jednotlivé logické prvky dokumentu* rozširujú možnosti LaTeXovských 'cross-referencií' v dvoch základných smeroch: sú umožnené 'cross-referencie' medzi dokumentami a referencie slúžia ako hypertextové uzly v dokumente, t.j. dvojklik na hypertextový uzol zobrazí referencovanú časť dokumentu.
- *Automatické vytváranie niektorých častí dokumentu,* napríklad obsahu dokumentu.
- *Inteligentný pohyb po dokumente.* Okrem pohybu po dokumente bežného z neštrukturovaných editorov, štruktúra dokumentu umožňuje realizovať navyše pohyb po logických prvkoch, vyhľadávanie logických prvkov, resp. referencovaných prvkov. Pohodlné je aj využitie 'cross-referencií' ako hypertextových uzlov.
- *Manipulácie s logickými prvkami v rámci DTD* zahŕňajú jednak zmeny atribútov, pridávanie nadštruktúry (napríklad štruktúru integrál nad štruk-

túru zlomok) logických prvkov, resp. zmeny jedného logického prvku na druhý. Sú kontextovo-závislé a editor ponúka autorovi uskutočniť len zmenu, ktorá je možná v rámci vybraného DTD.
- *Export do ostatných formátov* je dôležitý pre využitie dokumentu v iných systémoch. Príkladom môže byť export dokumentov do TEXu pre využitie typografických predností TEXu. Užívateľ má možnosť zadefinovať si export dokumentu (v tzv. T-jazyku) do ľubovoľného formátu vhodného pre príslušný typ aplikácie.

Zobecnenie predchádzajúcich výhod dáva základnú predstavu a možnostiach štrukturovaných editorov.

Euromath systém

Ako už bolo spomenuté, GRIF editor tvorí jadro Euromath systému. Vznik a vývoj Euromath systému bol koordinovaný cez European Mathematical Trust. Do jeho vývoja bolo zapojených niekoľko inštitúcií a firiem, spomeňme aspoň Fachinformationszentrum v Karlsruhe, 'Institut National de Researche en Informatique et en Automatique' (INRIA) sídliacu v Grenoble a Euromath Centre v Kodani. V súčasnosti sa jeho ďalším vývojom, distribúciou a užívateľskou podporou zaoberá Euromath Support Center so sídlom v Bratislave. Posledná distribuovaná verzia Euromath 2.0 bola dostupná pod Unixom na SUNovské platformy.[4]

Cieľom Euromath systému je vytvoriť uniformné prostredie pre základné počítačové aplikácie, ktoré môže potrebovať matematik pri svojej práci a zabezpečiť komunikáciu medzi aplikáciami založenú na jednotnom dátovom modeli. To zahŕňa pohodlné vytváranie dokumentu, komunikáciu s databázami, využívanie existujúceho matematického softwaru a sieťových vecí ako napríklad elektronickej pošty, resp. zobrazovanie vzdialených SGML dokumentov. Komunikácia medzi aplikáciami musí obsahovať informácie o štruktúre prenášaných dát, preto je založená na dátovom modeli využívajúcom SGML.

Editovanie dokumentu prebieha WYSIWYG spôsobom v štrukturovanom editore GRIF, o ktorého základných vlastnostiach sme sa už zmienili. Väčšina matematikov je však zvyknutá používať na tvorbu dokumentov TEX. TEX bez nejakého štandardného balíka makier (napr. LATEX) neposkytuje dostatočné prostriedky pre popis štruktúry dokumentu. Výnimku tvoria matematické výrazy, štruktúra ktorých sa dá vo väčšine prípadov rozanalyzovať. Vďaka tomuto faktu mohla byť realizovaná aplikácia umožňujúca preklad matematických výrazov z jednej štuktúry (TEX) do druhej (SGML) a to v obidvoch smeroch. Dôsledkom tohto je možnosť vkladania matematických výrazov aj ako LATEXového re-

[4] Podrobnejšie informácie o Euromath systéme je možné nájsť na adrese `http://www.dcs.fmph.uniba.sk/~emt`.

ťazca. Navyše kedykoľvek je možné prepínať sa medzi LaTeXovskou a WYSIWYG (SGML) formou matematického výrazu a editovať výraz vo zvolenom formáte.

LaTeXovské štýly zodpovedajú možnostiam všeobecného značkovania a tak bolo možné realizovať preklad štruktúry LaTeXovského štýlu do odpovedajúceho DTD (napr. 'article' do 'article'). Pre užívateľa to znamená možnosť importu LaTeXovských dokumentov do GRIFu, vizualizáciu matematických formuliek a možnosť pokračovať v editovaní dokumentu WYSIWYG formou. Opačný smer prekladu formátu bol zmieňovaný pri exporte do TeXového formátu.

Databázovou externou aplikáciou je pripojenie PFS (Personal File System) k editoru, ktorý umožňuje matematikovi hľadať informácie o článkoch v databáze Zentralblattu (online pripojenie do Karlsruhe alebo z CDromov) a následne zobraziť nájdené informácie ako dokument v GRIF editore. Výhodou takéhoto prepojenia je okamžité WYSIWYG zobrazenie formuliek vyskytujúcich sa v nájdených abstraktoch, resp. automatické vytváranie bibliografickej časti dokumentu.

Elektronická pošta je príkladom ďalšej aplikácie nad elektronickým dokumentom. Umožňuje exportovať dokument do zvoleného formátu (najčastejšie ASCII a TeX) a odoslať. (Prijímateľ zatiaľ nebol implementovaný.)

Významné sú aj ostatné sieťové aplikácie, ktoré realizujú URL linky, t.j. umožňujú sprístupňovanie vzdialených dokumentov. Uvedenú vlastnosť majú aj WWW-prehliadače ale s tým podstatným rozdielom, že WWW-prehliadače pridávajú prezentáciu len HTML dokumentom. Euromath systém môže vizualizovať ľubovoľný dokument, ktorého DTD pozná. Špeciálne teda aj dokument plný matematických výrazov alebo tabuliek.

Komunikácia s matematickými programami bola zatiaľ implementovaná len na experimentálnej úrovni. Snahou je umožniť výpočet užívateľom označenej formuly zavolaním nejakého 'computer algebra' systému (napr. MuPAD alebo Maple) a automatické dosadenie výsledku do dokumentu.

Zhrnutie

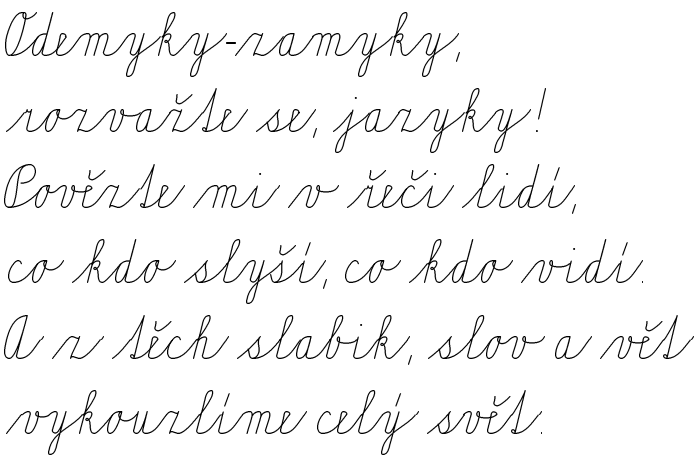
Euromath systém je príkladom štrukturovaného SGML editora, ktorý realizuje rôzne typy aplikácií postavených nad elektronickým dokumentom. Štruktúra dokumentu sa ukázala ako kľúčový pojem pre elektronické dokumenty. Štrukturované dokumenty sú nesporne výhodnejšie aj pre dlhodobejšiu použiteľnosť dokumentu (i dnes ešte neznámou aplikáciou) a ponúkanú otvorenosť. Preto sa dá očakávať nástup štrukturovaných textových editorov, medzi ktorými sa snáď objaví hviezdička s toľkými superlatívami, akou je TeX medzi textovými systémami (len tri za všetky: najlacnejší, najspoľahlivejší, najinteligentnejší).

Janka Chlebíková
Katedra vyučovania informatiky, MFF UK Bratislava
chlebikj@dcs.fmph.uniba.sk

Psané písmo ze slabikáře

Petr Olšák

Můj syn Mirek začal chodit do první třídy a učí se tedy psát. Rozhodl jsem se, že naučím psát počítač stejným písmem, s jakým se nyní setkává Mirek ve svých učebnicích. Znamenalo to vytvořit font v METAFONTu s psaným písmem podle slabikáře a zavést ligatury tak, aby písmena pěkně k sobě navazovala.



Přitom stačí do dokumentu napsat:

```
\font\pis=slabikar at 1cm
\def\\{\hfil\break}
\baselineskip=1cm  \pis \noindent
Odemyky-zamyky,\\ rozvažte se, jazyky!\\
Povězte mi v~řeči lidí,\\ co kdo slyší, co kdo vidí.\\
A~z~těch slabik, slov a~vět\\ vykouzlíme celý svět.
\end
```

K vytvoření tohoto písma jsem s výhodou využil tzv. „zobecněných ligatur“ a hraničních znaků. Aniž by se o tom příliš vědělo, i takové vymoženosti nám TEX poskytuje. V tomto článku se pokusím zmíněné mechanismy podrobně popsat a ukázat jejich použití na příkladu písma ze slabikáře.

Použité vlastnosti TEXu

Donald Knuth zavedl algoritmy pro zobecněné ligatury a pro hraniční znaky až od verze TEXu 3.0. Když doplňoval TEXbook [1] pro tuto novou verzi, pokusil se neměnit stránkování původní verze této knihy a přitom tam zanést informace o nových algoritmech. Protože novinek ve verzi 3.0 bylo více, znamenalo to vyvinout nadlidské úsilí, aby ty stránky zůstaly v původním stavu. Pokud člověk přesně zná rozdíly mezi verzí 2.x a 3.0, pak mu při četbě TEXbooku neujde, že se autor snažil některé formulace k novým algoritmům až nemístně stlačit do malého prostoru. Konečně, na některé věci nevybylo místo vůbec. V rejstříku například najdeme heslo `\noboundary`, ale k němu pouze stručnou zmínku v závorce o tom, že: »Ligatures and kerns may be influenced by invisible "boundary" characters at the left and right, unless `\noboundary` appears«. Co to ty "boundary" (hraniční) znaky jsou, se už nedovíme vůbec. K tomu je potřeba se podívat do zdrojového textu TEXu [2] nebo do [4], kde na straně 306 píšu:

»TEX může pracovat se dvěma hraničními znaky: pravý a levý. Pravý klade (jen teoreticky) na konec každého slova (souvislé posloupnosti znaků ze stejného fontu bez mezer). Levý hraniční znak se klade analogicky na začátek každého slova. Tyto znaky fakticky nejsou sázeny, ale mohou ovlivnit sazbu při vyhodnocování ligatur nebo implicitních kernů s okrajovým skutečně viditelným znakem ve slově. Viditelný znak α se může na začátku slova proměnit v ligaturu nebo před ním může být implicitní kern, pokud levý hraniční znak má ve svém odkazu do ligačního a kerningového programu test na znak α. Rovněž se znak α na konci slova může proměnit v ligaturu nebo za ním může být implicitní kern, pokud tento znak má odkaz do ligačního a kerningového programu, kde je test na pravý hraniční znak.«

Levý hraniční znak je v metrice fontu charakterizován ukazatelem do ligačního a kerningového programu, zatímco pravému hraničnímu znaku musí být přiřazen nějaký kód. Použití tohoto kódu v ligačním a kerningovém programu pak znamená, že se odkazujeme na pravý hraniční znak.

V METAFONTu je pro levý hraniční znak zaveden speciální symbol „`||:`“, který se použije jako návěští v záznamu `ligtable`. Není-li toto návěští nikde použito, TEX s levým hraničním znakem nepracuje. Pro případ pravého hraničního znaku je nutno v METAFONTu naplnit primitivní proměnnou `boundarychar` zvoleným kódem. Tento kód pak můžeme používat v záznamu `ligtable`. Není-li tato proměnná naplněna nebo má hodnotu mimo rozsah 0–255, pak TEX s pravým hraničním znakem nepracuje.

Kromě hraničních znaků je TEX vybaven schopností pracovat s tzv. „zobecněnými ligaturami“. Informace o nich můžeme rovněž hledat v [2] nebo [4]. Nechť dvojice znaků $\alpha\beta$ tvoří ligaturu γ. Pak TEX nejprve teoreticky vytvoří trojici $\alpha\gamma\beta$ a dále podle dodatečných údajů v metrice rozhodne o tom, zda α či β či oba tyto znaky ze sazby vyhodí.

V METAFONTbooku [3] pak najdeme možnosti, jak tyto zobecněné ligatury zapisovat do záznamu `ligtable` v METAFONTovém textu. Všechny případy jsou zahrnuty do následující tabulky:

`ligtable`	výsledná sekvence	případné další změny
α `:` β `|=:|` γ	$\alpha\gamma\beta$	$\alpha\gamma, \gamma\beta$
α `:` β `|=:|>` γ	$\alpha\gamma\beta$	$\gamma\beta$
α `:` β `|=:|>>` γ	$\alpha\gamma\beta$	
α `:` β `=:|` γ	$\gamma\beta$	$\gamma\beta$
α `:` β `=:|>` γ	$\gamma\beta$	
α `:` β `|=:` γ	$\alpha\gamma$	$\alpha\gamma$
α `:` β `|=:>` γ	$\alpha\gamma$	
α `:` β `=:` γ	γ	

Všimneme si, že „obyčejná“ ligatura, jakou jistě čtenáři znají z notorického příkladu „fi“, je realizována zápisem „`=:`“ podle posledního řádku tabulky. V tomto případě se dvojice „f“ a „i“ promění v ligaturu „fi“ a původní písmena „f“ a „i“ jsou ze sazby odstraněna. Ostatní řádky nabízejí další, rozšířené možnosti.

V posledním sloupci tabulky jsou uvedeny dvojice, které dále podléhají testu na zobecněnou ligaturu nebo implicitní kern. Je-li například v `ligtable` použit operátor „`|:=|>`“, pak kromě zařazení zobecněné ligatury γ mezi znaky α a β se bude dále testovat, zda dvojice $\gamma\beta$ netvoří nějakou ligaturu nebo implicitní kern. S touto vlastností je potřeba pracovat opatrně, protože může dojít k rekurzivnímu tvoření zobecněných ligatur až po zaplnění veškeré paměti TEXu.

Realizace psaného písma

V uvedeném písmu podle slabikáře jsem využil jednak vlastnosti hraničních znaků a jednak zobecněných ligatur. Písmena malé abecedy mají metriky i jednotlivé tahy voleny tak, aby znaky na sebe navazovaly v šesti sedminách střední výšky písma a aby v tomto bodě měly smluvený, pro všechny dvojice písmen stejný, sklon. Příklad:

Každé slovo začínající malým písmenem (s výjimkou písmen a, c, d, g, o, q) začíná náběhovou čárkou: ⁄ nebo: /. Jednotlivá písmena už tuto čárku nemají. Například písmeno „e“ z tohoto důvodu vypadá dost nezvykle – spíše jako „c“.

Jednotlivá písmena na sebe navazují bez výjimek přesně podle svých metrik (metriky znaků jsou v ukázce vyznačeny obdélníčkem). Nakonec každé slovo ukončíme maličkou dotahovou čárečkou: ´. Bez ní by totiž tah končil v šesti sedminách střední výšky písma. Ačkoli by to možná z estetického hlediska bylo lepší, prvňák si to nemůže dovolit a musí dokončovat tahy až na střední dotažnici. Naším úkolem bylo vytvořit písmo shodné s psaným písmem ve slabikáři bez kompromisů.

Náběhová i dotahová čárka bude ke každému slovu připojena automaticky prostřednictvím mechanismu ligatur a hraničních znaků. Jak již bylo řečeno, pro levý hraniční znak použijeme v METAFONTu v záznamu `ligtable` návěští „`||:`“. Například:

```
ligtable  ||: "-":
      ...,
      "r" |=:|> 3,  rcaron |=:|> 3,
      ...;
```

znamená, že levý hraniční znak nebo spojovník tvoří s následujícím „r“ nebo „ř“ ligaturu s kódem 3 (levá náběhová čárka). Přitom jednak neviditelný hraniční znak nebo spojovník a jednak znak „r“ nebo „ř“ v sazbě zůstávají. Samozřejmě, že ve fontu je v místě teček naší ukázky postupně vyjmenovaná celá abeceda malých písmen. U výjimečných písmen, která nepotřebují náběhovou čárku, můžeme například číst:

```
ligtable  ||: "-":
      "a" kern kk#,  aacute kern kk#,
      ...;
```

Vidíme, že tyto znaky budou pouze posunuty tak, aby byla kompenzována skutečnost, že vyčnívají vlevo ze svých hranic daných metrikou.

Pro pravý hraniční znak je nutno volit v METAFONTu nějaký kód pomocí proměnné `boundarychar`. V našem fontu byl za pravý hraniční znak zvolen kód 1. Tento kód má také pravá dotahová čárka, ale sama od sebe se nevykreslí. Musí se použít mechanismus ligatur. Část `ligtable`, která řeší tento problém, vypadá takto:

```
boundarychar:=1;
ligtable "a": aacute: "A": Aacute:  "b":
   ...:
   "z": zcaron: "Z": Zcaron:
      1 |=:> 1, "-" |=:|>> 1, "!" |=:|>> 1,
      "?" |=:|>> 1,  "," |=:|>> 1,  "." |=:|>> 1;
```

Místo teček najdete ve skutečném `mf` souboru všechna písmena, která potřebují dotáhnout na střední dotažnici. Kód `1 |=:> 1` zajistí, aby se na konci

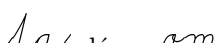
slov objevila dotahová čárka. Ostatní kódy zajistí totéž, pokud na konec slova navazuje spojovník, vykřičník, otazník, čárka a tečka.

Pokud dojde k automatickému dělení slov, TEX po rozdělení znova přehodnotí situaci, jak sestavit ligatury. Spojovník na konci řádku proto způsobí doplnění dotahu na střední dotažnici pro předchozí písmeno. Na začátku dalšího řádku TEX znova vloží levý hraniční znak a váže na něj ligatury. Proto se tam objeví správně náběhová čárka.

Velká písmena psané abecedy jsou dvojího druhu. Například „K“ se nijak neliší od malého písmene. Věc je dokonce jednodušší, protože písmeno nevyžaduje levou náběhovou čárku. Jeho dotah končí v šesti sedminách střední výšky písma a může na něj okamžitě navázat další malé písmeno. Druhým typem je například písmeno „B“, které v základním tvaru vypadá tak, jako bychom jej psali samostatně. Pokud se vyskytne těsně za písmenem „B“ jakékoli malé písmeno, reaguje tato dvojice vloženou zobecněnou ligaturou, která obsahuje spojovací čárku: ⁀. Samozřejmě, původní znaky, které tuto ligaturu vyvolaly, v takové situaci ze sazby nelikvidujeme.

Italické korekce byly vloženy jen k některým velkým písmenům. Metriky těchto písmen jsou totiž navrženy s ohledem na to, aby přesně navázalo následující malé písmeno (prostřednictvím spojovací čárky). Hranice kresby těchto písmen proto nemusí odpovídat metrikám. To začne vadit, až budeme klást velká písmena těsně vedle sebe (například ve zkratkách). V takovém případě doporučuji vyvolat italickou korekci. Písmo totiž není primárně navrženo pro psaní pouze velkými písmeny.

Za zmínku stojí ještě možnost automaticky vyrovnávat některé dvojice malých písmen.

 lépe: 

Vlevo vidíme nevyrovnané „o“ a „m“ (stojí opticky daleko od sebe). Vpravo je již tato dvojice vyrovnána. S tabulkou kerningových párů si tady nepomůžeme. Proto jsem pro tyto případy vytvořil alternativní zúžené „o“ (a také b, v a w). Tyto alternativy mají kratší dotah. Do tabulky ligatur jsem zavedl pro dvojice (b,o,v,w)–(m,n,v,w,y) zobecněnou ligaturu, která likviduje jen levý znak a nahrazuje jej zúženou alternativou.

V abecedě malých písmen je jeden zajímavý znak. Je jím písmeno „s“. Toto písmeno má ve fontu několik alternativ.

základní tvar: , např. ,

hlubší tvar:  např.  ,

plný tvar: např. .

Základní tvar je použitelný za všemi písmeny, která mají dotah stejného tvaru, jako například písmeno „m“. Kromě toho existuje „hlubší“ varianta písmene „s“, která se připojuje za písmena končící spodní kličkou: G, g, J, j, Q, q, Y, y. Nakonec ještě existuje samostatná plná varianta písmene „s“. Tato varianta se použije zcela na začátku slova. Experimenty totiž ukázaly, že pokud sestavíme toto písmeno pomocí běžné náběhové čárky, pak výsledek působí na začátku slova poněkud přiťáple. Proměnu písmene „s“ na začátku slova v plnou variantu zařídíme pomocí ligatury s levým hraničním znakem. Proměnu za písmenem s kličkou v hlubší variantu zařídíme pomocí zobecněné ligatury. Všechny varianty písmene „s“ automaticky připojují pomocí zobecněné ligatury spojovací čárku na další písmeno. Používá se stejný mechanismus, jako při vložení spojovací čárky třeba za velkým písmenem „B“. Přitom podle tvaru následujícího písmene jsou ve fontu připraveny různě dlouhé spojovací čárky. Naprosto stejné alternativy jako „s“ má samozřejmě písmeno „š“.

Několik postřehů

Písmo pro první třídu navrhli naši předkové tak, aby bylo jednoduché. Je tedy postaveno pouze na několika základních principech. Mezi tyto principy patří naprosto stejné napojování pro zcela všechna písmena. Jako obvykle platí: v jednoduchosti je síla.

Toto písmo tvoří základ, ze kterého se pak vyvine rukopis každého z nás. Právě pro svou jednoduchost je to velmi dobrý základ. Je ovšem pravda, že někteří jedinci nemají v podstatě žádný rukopis. S přibývajícím množstvím klávesnic bude těchto jedinců pravděpodobně přibývat. A to je škoda. Všiml jsem si, že tyto bytosti bez rukopisu se poměrně hojně vyskytují v řadách studentů ČVUT. Většinou jsou to dost velcí nešťastníci, protože si nejsou schopni zaznamenat ani přednášku. Občas se nějaký student na cvičení z matematiky brání na tabuli napsat souvislé slovo, protože mylně předpokládá, že v matematice se to nesluší a že stačí psát speciální značky. V takovém případě jej většinou požádám, aby napsal aspoň zadání příkladu slovy. Objeví-li se na tabuli hůlkové písmo, pak si pomyslím, co se ten student vlastně učil v první třídě. Podle grafických projevů z písemných prací studentů se přitom mohu přesvědčit, že takových lidí je více. Bohužel.

Jako vzor při návrhu písma jsem čerpal ze slabikáře [5]. I tyto vzory podléhají vývoji, takže můžeme nalézt v různých slabikářích nepatrné odlišnosti. Například písmeno „z“ se může zdát v této variantě poněkud nezvyklé. Zřejmě také záleží na kaligrafovi, který dělal pro příslušný slabikář vzory psaného písma.

Hotové písmo zveřejňuji v METAFONTovém zdroji na Internetu k volnému použití. Najdete je na `ftp://math.feld.cvut.cz/pub/olsak/slabikar`. Písmo je implementováno do jediného souboru `slabikar.mf`. V souboru

`ukazka.tex` najdete jednak ukázku písma a také doporučení, jak nastavit pro písmo některé parametry TEXu. Toto písmo je potřeba brát spíš jako příklad, co všechno TEX dovede. Nepředpokládám velké nasazení tohoto písma pro sazbu příštích slabikářů. Dokonce takovou věc ani nedoporučuji.

Všechny ukázky ve slabikáři a v písankách, které jsem měl možnost vidět, jsou psány lidskou rukou a ne strojem. Samozřejmě smekám před kaligrafem, který ty ukázky vytvořil. Člověk má na první pohled dojem, že to je „jak když tiskne“. Fušoval jsem také do kaligrafického řemesla, a proto dobře vím, že pokud písmo neobsahuje žádné ozdobné prvky, musí to napsat skutečně profesionál. Každá chybička, která by se třeba skryla za ozdobným prvkem, je totiž vidět.

Důležité ale je, že písmo v dnešním slabikáři bylo skutečně napsáno jen *jak* když tiskne a nikoli tištěno doopravdy. Písmu tak neschází lidský rozměr, který ten prvňák podvědomě z toho písma asi cítí. Kdyby se pro sazbu ukázek použil stroj (třebaže TEX), písmo by tento rozměr ztratilo. Takové písmo by bylo chladné, studené, stále stejné, bez výrazu, tedy vlastně mrtvé. Nepřeji žádným prvňákům, aby se někdy v budoucnu s takovým chladným písmem setkali. Pokud by snad někoho napadlo mým METAFONTovým písmem tisknout písanky nebo slabikáře, budu první, kdo bude proti. Raději se nabídnu jako kaligraf, který požadované ukázky napíše rukou.

Literatura

[1] Donald Knuth. *The TEXbook*, volume A of *Computers & Typesetting*. Addison-Wesley, Reading, MA, USA, 1989.

[2] Donald Knuth. *TEX: The Program*, volume B of *Computers & Typesetting*. Addison-Wesley, Reading, MA, USA, 1989.

[3] Donald Knuth. *The METAFONTbook*, volume C of *Computers & Typesetting*. Addison-Wesley, Reading, MA, USA, 1991.

[4] Petr Olšák. *TEXbook naruby*, Konvoj, 1997.

[5] Jiří Žáček, Helena Zmatlíková. *Slabikář*, Alter, 1996.

O makru PSTricks

Luboš Doležal

1. Úvod

PSTricks je kolekce maker TEXu založených na PostScriptu. Jejich autorem je Timothy Van Zandt. Makra jsou vhodná pro vytváření jednoduchých obrázků do technických dokumentů. Všechna makra PSTricks jsou kompatibilní s běžnými formáty TEXu, a to plainTEX, LATEX, $\mathcal{AMS}$-TEX – jak je udáno[2]. Sám jsem ovšem PSTricks vyzkoušel pouze v plainTEXu, takže uvedenou kompatibilitu k jinému formátu nemohu potvrdit, ale ani vyloučit. Tento krátký popis maker balíku PSTricks si nečiní nároky na kompletnost, mnoho možností zde nebude uvedeno – tyto možnosti lze nalézt v [1]. Odtud jsou též se souhlasem autora převzaty ukázky použité v článku.

Hlavní soubor, který na začátku načteme, je `pstricks.tex`. Lze také použít `pstricks.sty`. Tyto dva soubory jsou ekvivalentní, ale druhý uvedený je přizpůsoben pro použití v LATEXu.

2. Argumenty, parametry grafiky

Na začátku před vlastním vytvářením čehokoliv pomocí PSTricks musíme zajistit nastavení základních jednotek např. pro délku. To se provede příkazem `\psset` takto:

```
\psset{xunit=1pt, yunit=1pt}
\psset{xunit=1cm, yunit=2pt}
```

Jak je vidět, rozměrové jednotky osy x a y, tedy `xunit` a `yunit`, mohou mít buď stejné nebo i rozdílné velikosti. Implicitně jsou obě uvedené jednotky stejné a mají velikost 1 cm.

Kromě rozměrových jednotek se v PSTricks zadávají též úhly a směry. Tyto údaje zadáváme ve stupních, což je implicitní nastavení.

Uvedenou kontrolní sekvencí `\psset` lze nastavit více parametrů než jen rozměrové jednotky. Například i parametry čar – jejich tloušťku (`linewidth=`⟨*dim*⟩) a barvu (implicitně černá). Pro výplně ploch lze nastavit `fillstyle=`⟨*style*⟩,

[1] Tento článek vznikl jako semestrální práce z předmětu „Typografický systém TEX“, který přednáší Petr Olšák na Elektrotechnické fakultě ČVUT.

[2] Balík PSTricks předpokládá zpracování DVI-souboru PostScriptovou cestou.

`fillcolor=`⟨*color*⟩. Když je nastaveno `fillstyle=none`, není plocha vyplněna, ale když `fillstyle=solid`, je plocha vyplněna barvou `fillcolor`.

3. Základní grafické objekty

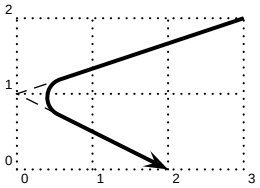
V této části budou uvedeny způsoby vytváření základní grafických objektů, tj. čar, uzavřených objektů vytvořených z čar, kružnic, elips a křivek.

Kontrolní sekvence pro vykreslení čáry má toto základní schema:

```
\psline[⟨par⟩]{⟨arrows⟩}(x0,y0)(x1,y1)...(xn,yn)
```

Tento příkaz vykreslí čáru přes dané souřadnice. V hranatých závorkách ⟨*par*⟩ jsou parametry dané čáry, např. barva, tloušťka – jak bylo uvedeno výše. Parametr ⟨*arrows*⟩ představuje tvar konců čáry. Například:

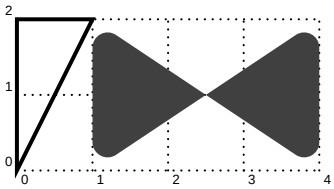
```
\psline[linewidth=1.5pt, linearc=0.25]{->}(3,2)(0,1)(2,0)
```



vykreslí čáru tloušťky 1,5 pt zakončenou šipkou. Počátek čáry je bodě `(3,2)` a její konec v bodě `(2,0)`. Parametr `linearc` představuje poloměr stočení čáry v „rohu“. Připojená čárkovaná čára (`linestyle=dashed`) na uvedeném obrázku má `linearc=0`, což představuje, jak je vidět, pouhé spojení tří uvedených bodů.

Zvláštním případem `\psline` je příkaz `\pspolygon`, který vytváří uzavřenou cestu. Jeho zápis je obdobný jako pro `\psline`, tedy:

```
\pspolygon[⟨par⟩](x0,y0)(x1,y1)...(xn,yn)
```



Příklad užití makra `\pspolygon` byl pořízen následujícími řádky:

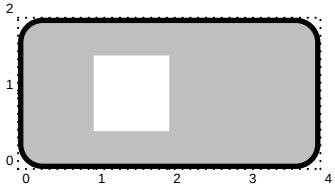
```
\pspolygon[linewidth=1.5pt](0,2)(1,2)
\pspolygon*[linearc=0.2,
linecolor=darkgray](1,0)(1,2)(4,0)(4,2)
```

Trojúhelník na obrázku vytvořil první řádek, tloušťka čáry je zřejmě 1,5 pt. Příkaz `\pspolygon*` je pouze zkráceným vyjádřením pro následující zápis:

```
\pspolygon[linecolor=⟨color⟩, fillstyle=solid, fillcolor=⟨color⟩]
```

Dalším možným příkazem této části může být `\psframe`. Tento příkaz se zadává ve formě:

`\psframe[`⟨*par*⟩`](x0,y0)(x1,y1)`



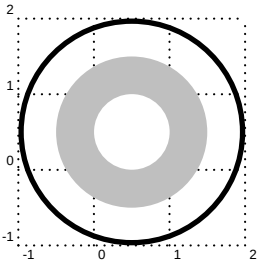
a způsobí vykreslení obdélníku, u něhož `(x0,y0)` je levý dolní a `(x1,y1)` pravý horní roh.

```
\psframe[linewidth=2pt, framearc=0.3,
fillstyle=solid,fillcolor=lightgray](4,2)
\psframe*[linecolor=white](1,0.5)(2,1.5)
```

Poslední řádek v předchozím zápisu uvozený `\psframe*` vyjadřuje, stejně jako v případě `\pspolygon`, určitý zkrácený zápis. V tomto případě je možné `\psframe*` chápat jako:

`\psframe[linecolor=`⟨*color*⟩`, fillstyle=solid, fillcolor=`⟨*color*⟩`]`

což pro `linecolor=white` a také `fillcolor=white` zřejmě dá „vepsaný“ obdélník z předchozího příkladu.



Kružnici lze vykreslit příkazem `\pscircle`. Zadání pozice a rozměrů je provedeno zadáním středu `(x0,y0)` a poloměru `{`⟨*radius*⟩`}`. Celkové schema je potom následující:

`\pscircle[`⟨*par*⟩`](x0,y0){`⟨*radius*⟩`}`

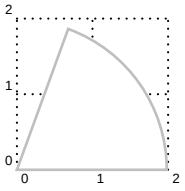
V ukázce je uvedeno několik dalších možností. Kružnice s největším poloměrem má tloušťku čáry 2 pt a její počátek náleží bodu `(.5,.5)`. Jak je patrno, její poloměr je 1,5 cm. Viz první řádek výpisu:

```
\pscircle[linewidth=2pt](0.5,0.5){1.5}
\pscircle*[linecolor=lightgray](0.5,0.5){1}
\pscircle*[linecolor=white](0.5,0.5){0.5}
```

Na druhém uvedeném řádku je zápis pro vykreslení kruhu, který bude vyplněn podle zadání `linecolor`. Rozdíl mezi příkazem `\pscircle` a `\pscircle*` je stejný jako např. u `\psframe`. Poslední řádek je obdobný druhému, pouze je nastavena jiná barva – `white` a menší poloměr – `{0.5}`.

Dalším možným grafickým prvkem definovaným v makru PSTricks je kruhová výseč. To znamená, že z kružnice daného poloměru zobrazíme pouze daný úsek

ohraničený mezi dvěma zadanými úhly. Obecné schema zápisu je následující:

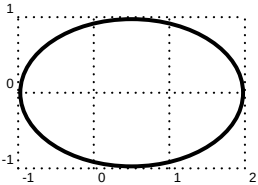


`\pswedge[`⟨*par*⟩`](x0,y0){`⟨*poloměr*⟩`}{`⟨*úhel1*⟩`}{`⟨*úhel2*⟩`}`

kde ⟨*par*⟩ jsou obvyklé parametry (barva, tloušťka čáry). Význam dalších potřebných parametrů – ⟨*poloměr*⟩, ⟨*úhel1*⟩, ⟨*úhel2*⟩ – byl již popsán výše. Ukázka tohoto příkazu byla vytvořena následující notací:

```
\pswedge[linecolor=lightgray,linewidth=1pt,
fillstyle=solid]{2}{0}{70}
```

U tohoto příkazu je také možná varianta `\pswedge*`, která má stejnou funkci jako v předešlých popsaných příkladech.



Rozšířením pro vykreslení kružnice je příkaz `\psellipse` resp. `\psellipse*`. Jak název napovídá jedná se o vykreslení elipsy. Ta je zadávána pomocí tří parametrů – středu, horizontálního poloměru a vertikálního poloměru, tedy:

`\psellipse[`⟨*par*⟩`](x0,y0)(r1,r2)`

kde ⟨*par*⟩ jsou znovu již popsané parametry, `(x0,y0)` je bod středu elipsy a `r1` resp. `r2` je velikost horizontálního resp. vertikálního poloměru. Jednoduchý příklad uvedený na obrázku má následující zápis:

```
\psellipse[linewidth=1.5pt](.5,0)(1.5,1)
```

Je zřejmé, že pomocí tohoto zápisu by šlo vytvořit také kružnici, samozřejmě zadáním `r1` = `r2`.

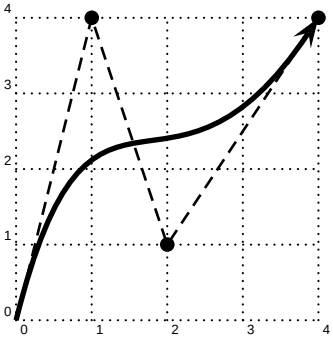
4. Křivky

V makru PSTricks jsou kromě grafických objektů uvedených výše definovány příkazy pro práci s křivkami. Snad nejzákladnějším příkazem této skupiny je `\psbezier`, jehož zápis je následující:

`\psbezier[`⟨*par*⟩`]{`⟨*arrows*⟩`}(x0,y0)(x1,y1)(x2,y2)(x3,y3)`

Uvedený příkaz vykreslí mezi `(x0,y0)` a `(x4,y4)` hladkou křivku. Ostatní dva zadané body jsou tzv. kontrolní body křivky. Tím je vytvořena Bézierova

křivka třetího řádu, která je v oblasti počítačové grafiky velice často používána, protože pro její konstrukci existují výkonné výpočetní mechanismy.

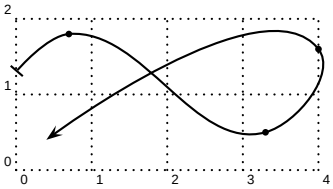


Dalšími parametry, které je možno zadat, jsou již popsané parametry ⟨*par*⟩. Také lze zadat tvar ukončení – parametr ⟨*arrows*⟩.

Příklad byl vytvořen následujícím zápisem:

```
\psbezier[linewidth=2pt,showpoints=true]%
{->}(0,0)(1,4)(2,1)(4,4)
```

Z předchozího popisu je zřejmé použití jednotlivých bodů, proto se blíže věnujme položce ⟨*par*⟩, ve které je uveden zajímavý a důležitý parametr `showpoints=true`. Ten nám (pokud je nastaven na `true`) vykreslí uvedené čtyři body a navzájem je propojí čárkovanou čarou – to je prospěšné, pokud potřebujeme danou křivku dále upravovat. V konečném případě lze samozřejmě nastavit `showpoints=false`, což nám dané body skryje.

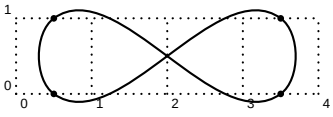


Předchozí příklad pro vykreslení křivky je sice názorný k předvedení Bézierovy kubiky, ale konstrukce pomocí něho je poněkud těžkopádná. Pokud máme několik bodů, které chceme proložit hladkou křivkou, užijeme spíše příkaz `\pscurve`. Jeho zápis je podobný předchozímu, tedy:

```
\pscurve[⟨par⟩]{⟨arrows⟩}(x1,y1)...(xn,yn)
```

Tento příkaz proloží přes body `(x1,y1)` až `(xn,yn)` otevřenou hladkou křivku. Jak již bylo uvedeno, tvar zakončení se řídí pomocí zadání ⟨*arrows*⟩. Parametry ⟨*par*⟩ mohou znovu obsahovat např. zadání tloušťky čáry, ale také příkaz `showpoint`, který nastaven na `true` označí body cesty, ale již je nepropojí čárkovanou čarou jako v případě `\psbezier`. Příklad byl vytvořen následujícím zápisem:

```
\pscurve[showpoints=true]{|->}(0,1.3)(0.7,1.8)(3.3,0.5)(4,1.6)
(0.4,0.4)
```



Rozšířením příkazu `\pscurve` je možnost vykreslit i uzavřenou křivku. Pro tyto případy existuje zápis `\psccurve`, kde „přebytečné c“ znamená *closed*. Zápis celého příkazu je obdobný, tedy:

`\psccurve[`⟨*par*⟩`]{`⟨*arrows*⟩`}(x1,y1)...(xn,yn)`

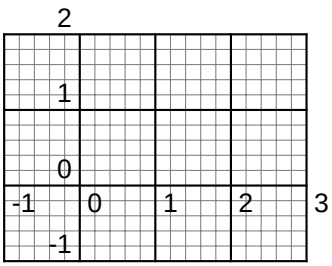
Obrázek k demonstraci tohoto příkazu byl pořízen zápisem:

```
\psccurve[showpoints=true](0,1.3)(0.5,0)%
(3.5,1)(3.5,0)(0.5,1)
```

5. Mřížky

Jak již bylo možno vidět, pro všechny zařazené ukázky makra PSTricks bylo užito jednoduché mřížky, do které byly příklady příkazů zakresleny. Tato mřížka je pouze speciálním případem pro příkaz `\psgrid`, jehož zápis je následující:

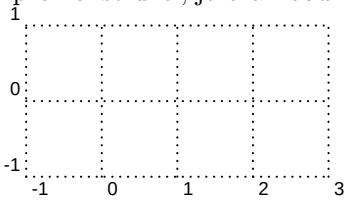
`\psgrid[`⟨*par*⟩`](x0,y0)(x1,y1)(x2,y2)`



Význam parametrů ⟨*par*⟩ bude uveden dále. Tento příkaz vykreslí mřížku s protilehlými rohy o souřadnicích `(x1,y1)` a `(x2,y2)`. Interval hodnot je číslován. Toto číslování začíná bodem `(x0,y0)`, jak ukazuje uvedený příklad:

```
\psgrid(0,0)(-1,-1)(3,2)
```

Koordinační systém je chápán jako kartézský souřadný systém. Dále je možné poznamenat, že pokud se neuvede pozice bodu `(x0,y0)`, tzn. provedeme zápis pouze dvou bodů pro konstrukci, je brán bod `(x1,y1)` jako počátek.



Na místo ⟨*par*⟩ lze zapsat mnoho parametrů, ze kterých asi nejdůležitější budou uvedeny v následujícím příkladě. Mřížka, která bude dále uvedena, byla v různých rozměrových obměnách použita v dříve uvedených ukázkách. Zápis pro vygenerování příkladu je následující:

```
\psgrid[subgriddiv=1,griddots=10,gridlabels=7pt](-1,-1)(3,1)
```

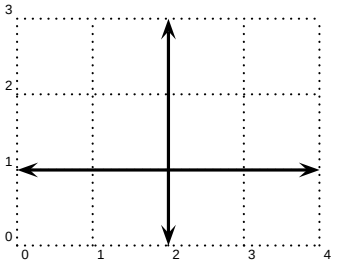
Parametr `subgriddiv` nastavuje jemnější dělení zakreslené mřížky. Implicitní hodnota je `subgriddiv=5`, což také koresponduje s první ukázkou této sekce. Dalším parametrem je `griddots=`⟨*num*⟩. Pokud je číselná hodnota ⟨*num*⟩ positivní, jsou čáry mřížky nakresleny tečkovaně, kde hodnota ⟨*num*⟩ je po-

čet bodů na dělení mřížky. Poslední parametr je `gridlabels`, který nastavuje stupeň písma pro popisek mřížky. Pro popisek je užíván implicitně bezserifový PostScriptový font Helvetica ve velikosti 10 pt, pokud není nastaveno jinak.

6. Osy

Velice podobné vlastnosti jako výše popsané makro `\psgrid` má také makro pro vytváření os (např. pro kreslení matematických funkcí nebo průběhů zjištěných měřením). Jedná se o makro `\psaxes`. Před jeho používáním je ale nutné načíst soubor `pst_plot.tex`, ve kterém je popsáno makro této sekce. Zápis je následující:

`\psaxes[`⟨*par*⟩`]{`⟨*arrows*⟩`}(x0,y0)(x1,y1)(x2,y2)`



Stejně jako v případě `\psgrid` jsou jednotlivé body `(x0,y0)` až `(x2,y2)` použity následujícím způsobem: počátek je zadán bodem `(x0,y0)`, levý dolní roh bodem `(x1,y1)` a protilehlý pravý horní `(x2,y2)`, jak je patrno srovnáním mezi uvedeným obrázkem a předpisem:

```
\psaxes[linewidth=1.2pt,labels=none,
ticks=none]{<->}(2,1)(0,0)(4,3)
```

Parametr ⟨*arrows*⟩ znovu dovoluje nastavit požadované ukončení jednotlivých os, v našem případě se šipkou. To je pravděpodobně nejpoužívanější možnost.

Nyní si blíže všimneme parametrů umístěných v ⟨*par*⟩, protože tento příkaz je má, oproti `\psgrid`, poněkud jiné. Některé nejdůležitější je možno vidět již u předchozího zápisu. Parametr `labels=none/x/y/all` dovoluje čtyři možnosti v umístění číslování pozic. Buď se toto číslování nepovolí nebo pouze na jednu nebo druhou osu nebo je možné číslování povolit na obě osy současně. Podobně se chová parametr `ticks`, který má za úkol zakreslit krátké (jejich délka se dá nastavit pomocí `ticksize=`⟨*num*⟩) kolmé čárky vůči osám na jednotlivých pozicích. Dále lze tyto čárky zakreslit pouze nad nebo pod osou, ale také oboustranně – parametr `tickstyle=full/top/bottom`.

7. Kreslení grafů funkcí

Téma této sekce může být jistým pokračováním sekcí předchozích. Do zakreslené mřížky nebo souřadných os lze dále zakreslit graf matematické funkce. Jediný

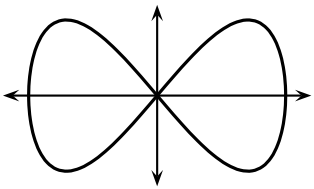
problém může způsobit asi zápis funkce – používá se PostScriptová notace zápisu. Pro vykreslení dané funkce se používá příkaz `\psplot` zápisem:

`\psplot[`⟨*par*⟩`]{`⟨*xmin*⟩`}{`⟨*xmax*⟩`}{`⟨*funkce*⟩`}`

Parametry ⟨*par*⟩ jsou již dříve uvedené parametry, např. pro nastavení tloušťky čáry. Numerické hodnoty ⟨*xmin*⟩ a ⟨*xmax*⟩ vymezují rozsah nezávisle proměnné x. Funkce je zadána zápisem ve ⟨*funkce*⟩. Jinak řečeno, ⟨*xmin*⟩ a ⟨*xmax*⟩ udávají interval, ve kterém bude daná funkce vykreslena. Užití jednotlivých parametrů je jistě patrné z následujícího velice jednoduchého příkladu:

```
\psplot[plotpoints=200,plotstyle=curve]{0}{720}{x sin}
```

který vytvoří graf funkce $\sin x$ pro x od 0° do 720°. V tomto intervalu bude vypočteno 200 hodnot (zadáno pomocí `plotpoints`), které budou proloženy hladkou křivkou. To je zadáno pomocí `plotstyle`. Lze také zadat `plotstyle=dots` – namísto hladké čáry bude graf tvořen tečkami.



Pro kreslení funkcí zadaných parametricky existuje příkaz `\parametricplot`, jehož zápis je následující:

`\parametricplot[`⟨*par*⟩`]%`
`{tmin}{tmax}{`⟨*funkce*⟩`}`

Snad jedinou změnou oproti příkazu `\psplot` je to, že namísto nezávisle proměnné x je zadáván interval s parametrem t. Způsob zápisu funkce (je to ale spíše vektor $(x(t), y(t))$ proměnné t) je patrný z následujícího příkladu užití:

```
\psset{xunit=1.7cm}
\parametricplot[linewidth=1.2pt,plotstyle=ccurve]%
{0}{360}{t sin t 2 mul sin}
\psline{<->}(0,-1.2)(0,1.2)
\psline{<->}(-1.2,0)(1.2,0)
```

Na obrázku byla vykreslena parametrická funkce $(\sin t, \sin 2t)$, pro $t_{\min} = 0°$ a $t_{\max} = 360°$. Jednoduché osy byly vytvořeny příkazem `\psline`. Samozřejmě, že by také šlo získat souřadný systém pomocí `\psaxes`.

8. Popisy

Často je třeba zařadit do nakresleného obrázku text, např. názvy os u souřadného systému při kreslení grafů nebo název vykreslené funkce. Pro tyto účely je v makru PSTricks definován příkaz `\uput`, jehož zápis je následující:

`\uput{`⟨*labelsep*⟩`}[`⟨*refangle*⟩`]{`⟨*angle*⟩`}(x,y){`⟨*popis*⟩`}`

Příkaz provede následující: vzhledem k bodu zadanému v `(x,y)` vloží ⟨*popis*⟩ ve vzdálenosti ⟨*labelsep*⟩ (implicitně je v makru nastaveno `labelsep=5pt`) od tohoto bodu. Dále se pomocí parametru ⟨*refangle*⟩ udává natočení celého textu v ⟨*popis*⟩ vzhledem k referenčnímu bodu `(x,y)`. Parametr ⟨*angle*⟩ udává potom vlastní natočení textu v ⟨*popis*⟩. K popisu se používá standardní TEXovský font. Implicitně není pro tento účel nastaveno používání PostScriptových písem.

Jednoduchý příklad – popis bodu – byl vytvořen zápisem:

```
\qdisk(1,1){2pt}
\uput[45](1,1){(1,1)}
```

Uvedený příkaz `\qdisk` je speciálním případem pro `\pscircle`. Zadává se bod umístění, v našem případě je to bod `(1,1)`, a průměr tohoto bodu pomocí čísla s jednotkou, např. 2 pt.

9. Umísťování grafiky do dokumentu

Grafické objekty, ale např. také příkaz `\uput`, nedokáží samostatně změnit chování TEXu. To znamená, že vytvářejí bezrozměrný box. Abychom mohli zařadit vytvořený obrázek do dokumentu, musíme příkazy tvořící tento obrázek umístit do prostředí `\pspicture`, jehož schema je následující:

`\pspicture[`⟨*baseline*⟩`](x0,y0)(x1,y1)`
⟨*popis objektu...*⟩
`\endpspicture`

Obrázek bude potom vložen do boxu. Levý dolní roh je v bodě `(x0,y0)` (implicitně nastaveno na `(0,0)`) a pravý horní okraj bude v bodě `(x1,y1)`.

Velikost parametru ⟨*baseline*⟩ je poměrná část z výšky daného boxu – může nabývat hodnoty od 0 do 1. Pokud je nastavena 0, potom se jedná o spodní okraj. Při hodnotě `baseline` 1 je nastaven horní okraj. Hodnoty „mezi“ pochopitelně

nastavují `baseline` mezi spodní a horní okraj. Implicitně je nastaveno `baseline` na spodní část vytvořeného boxu.

Pro uživatele LaTeXu je zápis pro toto prostředí poněkud jiný – odpovídá konvencím tohoto TeXovského makra, takže:

```
\begin{pspicture}[⟨baseline⟩](x0,y0)(x1,y1)
⟨popis objektu...⟩
\end{pspicture}
```

Dále lze v LaTeXu používat grafické objekty z PSTricks v LaTeXovském prostředí `picture` a naopak je možné používat objekty vytvořené v prostředí `picture` v prostředí `pspicture`. Samozřejmě se všemi omezeními, které jsou LaTeXovskému prostředí `picture` vlastní.

10. Závěr

Jak již bylo řečeno v úvodu tohoto dokumentu, není jeho cílem podat vyčerpávající popis všech možností a „triků“ s balíkem maker PSTricks. Není zde například podán úplný přehled parametrů, které se zapisují na místo ⟨*par*⟩ nebo do příkazu `\psset`. Jsou zde uvedeny pouze nejdůležitější parametry. Čtenář, který by se o toto makro více zajímal, si jistě vyhledá originální dokumentaci, samozřejmě v angličtině. Tam je celé makro velice podrobně popsáno včetně jeho možností. Tato originální dokumentace by měla být součástí každé distribuce PSTricks.

Literatura

[1] Timothy Van Zandt: *PSTricks: PostScript macros for Generic TeX. User's Guide.* Dostupné na CTAN, `graphics/pstricks/`.

Luboš Doležal
`dolezal1@feld.cvut.cz`

Obtekanie obrázkov v LaTeXu

Ján Buša

V Zpravodaji 1/96 bol publikovaný príspevok Libora Sýkoru o vkladaní obrázkov do LaTeXu. Rozhodol som sa doplniť niektoré informácie a uvádzam voľný a skrátený preklad príručky k makru `PicIns` a taktiež príkladov použitia makra `PicInPar`.

`PicIns` – Príručka Verzia 3.0

Joachim Bleser Edmund Lang

TH Darmstadt, Hochschulrechnenzentrum

September 1992

Úvod

Väčšina súčasných systémov na spracovanie textu a DTP – Desktop Publishing – systémov poskytuje možnosť integrácie obrázkov do textových dokumentov. Obrázky môžu byť ľubovoľne umiestnené a rôznymi spôsobmi orámované. Ako obzvlášť elegantné sa javí *obtekanie* obrázkov riadkami textu, čím sa dosiahne tesné prepojenie obrázkov s im odpovedajúcimi časťami textu.

TeX a LaTeX štandardne neposkytujú žiadne funkcie na tieto spôsoby. Pomocou nižšie popísaného balíka makier `PicIns`[1] je možná komfortná integrácia obrázkov do LaTeXovských dokumentov.

Pritom pod *obrázkom* rozumieme plošku, ohraničenú obdĺžnikom, pozostávajúcu z (možno neexistujúceho) rámčeka a z (možno prázdneho) obsahu.

Obsah obrázku môže byť určený pomocou funkcií LaTeXu (napr. normálnym textom, matematickými vzorcami, tabuľkami alebo konštrukciami, vyrobenými

[1] Predkladaná publikácia má chránené autorské práva. Zmeny obsahu si vyžadujú písomný súhlas autorov.

pomocou prostredia `picture`) alebo zaradením externe vyrobenej grafiky. Problematike integrácie externej grafiky sa budeme venovať v oddiele 5.

LaTeXovský štýlový súbor `picins.sty` neobsahuje *žiadne* funkcie na výrobu obsahu obrázkov. Poskytuje len nasledujúce LaTeXovské príkazy, potrebné na umiestnenie obrázkov, uvoľnenie priestoru obsadeného obrázkami, orámovanie alebo obtekanie obrázka textom:

`\parpic` — umiestňuje obrázok na začiatku odstavca

`\hpic` — umiestňuje obrázok do vlastného odstavca vedľa seba

`\picskip` — určuje počet riadkov nasledujúceho odstavca na obtekanie obrázku

`\pichskip` — určuje horizontálnu medzeru medzi obrázkom a textom

`\shadowthickness` — určuje hrúbku tieňa pri orámovaní obrázkov

`\dashlength` — určuje dĺžku čiaročiek pri prerušovanom rámčeku

`\boxlength` — určuje hĺbku škatuľky pri orámovaní 3D-škatuľkou

`\piccaption` — umožňuje zadanie nadpisu

`\newcaption` — vytvára nadpis trocha odlišne od LaTeXovského príkazu `\caption`

`\picchangemode` riadi umiestňovanie obrázkov v prípade obojstranných dokumentov

Popritom budú dané k dispozícii štyri nové prostredia. Budú otvárané, resp. zatvárané obvyklými `\begin{`*prostredie*`}` ... `\end{`*prostredie*`}`.

`frameenv` je prostredie, ktorého obsah bude orámovaný

`dashenv` je prostredie, ktorého obsah bude orámovaný prerušovanými čiarami

`ovalenv` je prostredie, ktorého obsah bude orámovaný rámčekom so zaoblenými rohmi

`frameenv` je prostredie, ktorého obsah bude orámovaný rámčekom s tieňom

Obrázky na začiatku odstavca: `\parpic`

Príkaz `\parpic` umiestňuje obrázok podľa voľby naľavo alebo napravo na *začiatku* odstavca. Text, ktorý môže pozostávať z viacerých odstavcov, obteká obrázok.

Syntax: `\parpic`(*šírka,výška*)(*x-vnútorný posun,y-posun*)[*voľby*][*pozícia*]{*Obsah obrázku*}

Všetky parametre, vrátane *obsahu obrázku*, sú voliteľné (nepovinné).

Popis parametrov:

šírka, výška: šírka určuje šírku obrázku. Riadky textu, obtekajúce obrázok, budú skrátené o hodnotu *šírky* zväčšenú o medzeru, definovanú na oddelenie

obrázku a textu. Tak sa zachová celková šírka odstavca. Na základe *výšky* obrázka sa vypočíta počet riadkov, potrebných na obtečenie obrázka.

Ak chýba toto zadanie, budú pre *šírku* a *výšku* použité rozmery najmenšieho obdĺžnika, úplne pokrývajúceho obsah obrázka (*bounding box*). Toto predpokladá, že obsah obrázka má skutočnú výšku a šírku, čo nie je prípad všetkých metód zaraďovania externe vyrobených obrázkov (pozri oddiel 5.6).

x-vnútorný posun,y-posun: Zadaním tejto dvojice hodnôt je možné posúvať obsah obrázka vo vnútri rámca v ľubovoľnom smere. Vzťažný bod (*počiatok, referenčný bod*) sa nachádza v ľavom hornom rohu. Kladný *x-posun* posúva obrázok napravo, kladný *y-posun* ho posúva smerom dole. Záporné hodnoty sú možné (pozri oddiel 5.6).

Ak chýba toto zadanie, bude *obsah obrázka* umiestnený odpovedajúco parametru *pozícia*. Parametre *posunu* sú zvlášť dôležité pri integrovaní externe vyrobeného obrázka (pozri oddiel 5.6).

voľby: Pomocou *volieb* môže byť určená poloha obrázka vo vzťahu k odstavcu a spôsob jeho „orámovania“:

`l` (*left*) — obrázok na ľavú stranu textu
`r` (*right*) — obrázok na pravú stranu textu
`f` (*frame*) — aj s rámčekom
`d` (*dash*) — čiarkovaný rámček
`o` (*oval*) — oválny rámček
`s` (*shadow*) — tieň
`x` (*box*) — 3D-škatulka

Kombinácie ľubovoľnej polohy a spôsobu orámovania sú možné (napr. `lf, dr, ro, ...`). Kombinácia viacerých parametrov pozície (napr. `lr, rl`) alebo parametrov rámčeka (napr. `os, do, ...`) spôsobuje chyby.

Ak nie sú zadané žiadne *voľby*, bude obrázok bez rámčeka umiestnený na ľavom okraji odstavca.

pozícia: Pomocou parametra *pozícia* môže byť určená pozícia vo vnútri rámčeka. Možné sú:

`l` (*left*) — obrázok na ľavý okraj rámčeka
`r` (*right*) — obrázok na pravý okraj rámčeka
`t` (*top*) — obrázok na vrchný okraj rámčeka
`b` (*bottom*) — obrázok na spodný okraj rámčeka

Ak nie je zadané horizontálne umiestňovanie (`l, r`), bude obrázok horizontálne centrovaný. Analogicky bude obsah obrázka centrovaný vertikálne, ak nebude zadané vertikálne nastavenie. Navzájom neodporujúce si pozície môžu byť spojené (`lt, lb, rt, rb`). Pri súčasnom zadaní parametrov *posun* a *pozícia* stráca parameter *pozícia* platnosť. Ak nie sú zadané ani *posun* ani *pozícia*, bude obrázok vo vnútri rámčeka vertikálne a horizontálne centrovaný. Ak bude zadaný

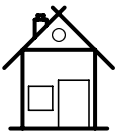
parameter *pozícia*, je potrebný aj parameter *voľby* (prípadne v tvare `[]`, teda napr. `\parpic(3cm,2cm)[][tr]{`*Obrázok*`}`).

Umiestnenie externe vytvorenej grafiky závisí na zvolenej metóde zaradenia tohoto obrázku (pozri oddiel 5).

obsah obrázka: Na záver nasleduje vlastný obsah obrázku, to, čo má byť umiestnené na voľné miesto. Obsah môže tvoriť ľubovoľná TEXovská alebo LATEXovská konštrukcia, napr. náčrtok vytvorený pomocou prostredia `picture`, alebo obrázok vytvorený externe. Príklady nasledujú nižšie.

Voľby príkazu `\parpic`

Väčšina nasledujúcich príkladov je orámovaná; to nie je bezpodmienečne nutné, ale ilustruje to možnosti príkazu `\parpic`. Príklady používajú polovičnú šírku riadku, aby bolo možné na pravej strane uviesť LATEXovské príkazy na výrobu príkladov. Príkaz `\Karty` je definovaný ako `$\clubsuit\diamondsuit\heartsuit\spadesuit$`. Box `\malydom` obsahuje domček, vyrobený pomocou prostredia `picture`[2]. Príkaz `\copy\malydom` skopíruje box `\malydom` na odpovedajúce miesto *bez* zmazania jeho obsahu (`\box\malydom` kopíruje box a maže jeho obsah). `\malydomsirka` a `\malydomvyska` sú rozmerové registre obsahujúce šírku a výšku boxu `\malydom`. Inicializácia zadania rozmerov je možná napr. pomocou `\malydomsirka=\wd\malydom`, ktorým sa zadá šírka boxu.



Najjednoduchší tvar príkazu `\parpic`. Všetky parametre, s výnimkou obsahu obrázka sú vynechané. To znamená, že šírka a výška obrázku sa vypočítajú automaticky.

```
\parpic{\copy\malydom}
Najjednoduchší tvar
príkazu ...
```

ešte trochu komplikovanejší tvar príkazu `\parpic`. Je možné zadať veľkosť obrázku. Ak chýbajú ďalšie parametre, obsah bude centrovaný.

```
\parpic(12mm,16mm)
{{\okrasny A}}
ešte trochu ...
```

Šírka tohoto okna je 3 cm a výška 1 cm. Keďže nie sú zadané žiadne posuny ani pozícia, je obsah automaticky centrovaný. `f` znamená rámček.

```
\parpic(3cm,1cm)[f]%
{\Karty}
Šírka tohoto okna
je 3 cm
a výška 1 cm ...
```

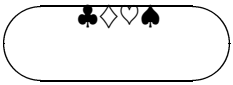
[2]Predtým je nutné zaviesť nový box príkazom `\newbox\malydom` a zadefinovať jeho obsah príkazom `\setbox\malydom=\hbox{obsah}` (pozn. prekl.).

V tomto príklade bude obsah obrázku posunutý vzhľadom na ľavý horný roh rámčeku (referenčný bod) pomocou vnútorných posunov. Zadávanie posunov má zmysel vtedy, keď automatické umiestnenie nevedie k uspokojujúcemu výsledku.

```
\parpic(3cm,1cm)%
(5mm,5mm)[f]{\Karty}
V~tomto
príklade ...
```

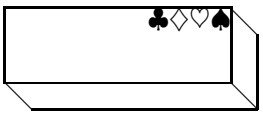
Obrázok[3] môže byť umiestnený aj na pravej strane odstavca. Na to je použitá voľba `[r]`. Dodatočná voľba `[s]` tento obrázok tieňuje. Obidve voľby sú dané spoločne.

```
\parpic(3cm,1cm)[sr]
{\Karty}
Obrázok môže
byť umiestnený ...
```



Voľba `[o]` rámuje obrázok so zaokrúhlenými rohmi. Pozícia `[t]` posunie obsah k hornému okraju rámčeka.

```
\parpic(3cm,1cm)[o]%
[t]{\Karty} Voľba
{\tt [o]}
rámuje obrázok so ...
```



Voľba `[x]` rámuje obrázok 3D-škatulkou. Pozícia `[tr]` posúva obsah k pravému hornému okraju rámčeka.

```
\parpic(3cm,1cm)[x]%
[tr]{\Karty} Voľba
{\tt [x]}
rámuje obrázok
3D-škatulkou. ...
```

Voľba `[d]` rámuje obrázok čiarkovane. Pozícia `[lb]` posúva obsah do ľavého dolného rohu rámčeka.

```
\parpic(3cm,1cm)[d]%
[lb]{\Karty} Voľba
{\tt [d]}
rámuje obrázok
čiarkovane. ...
```

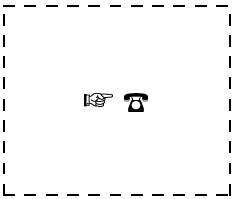
Umiestnenie obsahu obrázku vo vnútri rámčeka býva v normálnom prípade veľmi presné. Ak sa napriek tomu vyskytnú problémy, sú najčastejšie založené na tom, že konštrukcia, definujúca obrázok, nevykazuje skutočnú výšku a šírku, čo býva najčastejšie pri zaraďovaní externe vyrobeného obrázku. Problémy sa vyskytujú rovnako v prípade, ak je pri definícii obrázka vynechané na okrajoch voľné miesto. To sa stáva často v prostredí `picture` a vedie to tiež k tomu, že viditeľná časť obrázku je nesprávne umiestnená. V takých prípadoch musí byť obrázok na želateľnú pozíciu umiestnený pomocou parametra *posun*.

Text vedľa obrázka

Počet riadkov, ktoré majú byť vložené vedľa obrázka, sa dá ovplyvniť príkazom `\picskip`. Normálne bude skrátený taký počet riadkov, aby bol obrázok úplne obtečený. Ak chceme tento stav zmeniť, ako napr. ak chceme vytlačiť vedľa

[3] Medzery má na svedomí príkaz minipage.

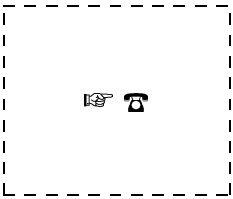
obrázka len jeden odstavec a ďalší odstavec chceme umiestniť znova pod obrázok, použijeme príkaz `\picskip{`n`}`. Príkaz `\picskip` ukončí aktuálny odstavec.



Nasledujúci odstavec sa už nemá obtekať.

```
\parpic(3cm,2.5cm)%
[d]{\rightruka\
    \telefon}
Nasledujúci odstavec
sa už nemá obtekať.
\picskip{0} $n=0$ ...
```

$n = 0$ (`\picskip{0}`) znamená, že už žiadne ďalšie riadky nasledujúceho odstavca nebudú obtekať text. Nasledujúci odstavec začína pod obrázkom.

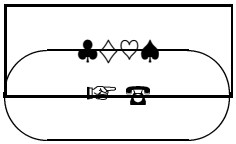


Ak bude mať `\picskip{`n`}` hodnotu $n > 0$, tak ešte n riadkov *nasledujúceho odstavca* bude obtekať obrázok. To môže byť potrebné v jednotlivých prípadoch, pri ktorých nebol správne vypočítaný počet kratších riadkov (pozri tiež oddiel „*Problémy*“). V takom prípade môže časť nasledujúcich riadkov textu prepísať spodnú časť obrázka, prípadne bude skrátených viac riadkov, ako bolo požadované.

```
\parpic(3cm,2.5cm)%
[d]{\rightruka\
    \telefon}
Ak bude mať
$\backslash$%
{\tt picskip}$\{n\}$
hodnotu $n>0$, tak
ešte $n$ riadkov
\emph{nasledujúceho
odstavca} bude
obtekať obrázok.
\picskip{3}
To môže byť ...
```

Za príkazom `\picskip` s hodnotou $n > 0$ nesmie nasledovať príkaz `\par` (ukončenie odstavca), ani prázdny riadok[4]. Príkaz `\par` je po príkaze `\picskip` s hodnotou $n = 0$ nadbytočný.

Príkaz `\picskip{0}` je potrebný zvlášť vtedy, ak za sebou nasledujú dva príkazy `\parpic` s malým množstvom textu. Potom môže nastať nasledujúci neželateľný stav:



Odstavec 1: Tento odstavec neobteká obrázok úplne.

Odstavec 2: Preto sa obrázky prekrývajú.

Tejto chybe sa dá vyhnúť, ak sa po *prvom odstavci* použije vyššie popísaný príkaz `\picskip{0}`.

[4]Skrátenie riadkov sa dosahuje TEXovskými primitívmi `\hangindent` a `\hangafter` Príkaz `\par` nastaví `\hangindent` naspäť na `0pt`, takže žiadne ďalšie riadky nebudú viac skrátené.

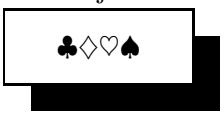
Hrúbka čiar, tieňa, dĺžka strán a čiaročiek

LaTeXovský príkaz `\linethickness` umožňuje meniť hrúbku rámčeka v príkaze `\parpic` s voľbami `f, d` alebo `s`. Zmena platí dovtedy, kým nebude znova nastavená pôvodná hodnota. Nastavená hodnota hrúbky rámčeka je `0.4pt`. Dĺžka čiaročiek pri čiarkovanom rámčeku sa môže meniť pomocou príkazu `\dashlength` definovaného v `picins.sty` (napr. `\dashlength{2pt}`). Predefinovaná hodnota je `4pt`.

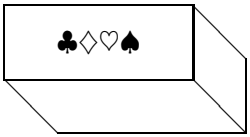
Tri nasledujúce príklady budú vyrobené vo vnútri prostredia `enumerate` a preto budú očíslované. Príkazy `\parpic` a `\hpic` sa môžu používať vo vnútri (aj hlbšie vnorené) zoznamov (list-ov). Chceli by sme len varovať pred použitím `description`-listu, v ktorom môže `itemtext` prepísať obrázok alebo vlastný text.

1. 

Kombinácia príkazov `dashlength` a `linethickness` umožňuje ďalšie možnosti utvárania rámčekov. V takom prípade je však potrebné dávať pozor na to, aby výška a šírka obrázka boli celočíselnými násobkami hodnoty `dashlength`. V tomto príklade boli zvolené `\dashlength{1mm}` a `\linethickness{1mm}`. Toto sú `default` hodnoty. Dĺžka čiaročiek sa samozrejme môže meniť nezávisle na hodnote hrúbky čiary.

2. 

V prípade tieňovaných rámčekov je možné voľne meniť *hrúbku tieňa*. Príkaz `\shadowthickness`, definovaný v `picins.sty` funguje analogicky ako LaTeXovský príkaz `\linethickness`. V tomto príklade bolo zvolené `\shadowthickness{10pt}`. Preddefinovaná je hodnota hrúbky tieňa `4pt`.

3. 

Dĺžka strany škatulky, t. zn. hĺbka škatulky, sa dá meniť príkazom `\boxlength`. Pritom treba dávať pozor na to, že naklonené čiary v LaTeXu nemôžu mať ľubovoľnú dĺžku. Preto pri určitých zadaniach hĺbky nebudú šikmé strany zobrazené. V tomto príklade sme zvolili `\boxlength{20pt}`. Nastavená je hĺbka `10pt`.

Medzera medzi obrázkom a textom

Horizontálna medzera medzi obrázkom a textom sa dá meniť príkazom `\pichskip{`*dim*`}` (*h* v príkaze `\pichskip` znamená *horizontálna*). Predvoľba medzery je `1em`. V uvedenom príklade používame `3em`.

```
\pichskip{3em}%
\parpic(3cm,1cm)%
[d]{\Karty}
Horizontálna ...
```

Nadpisy

Ako sme vyššie spomínali, *obsah obrázku* nemusí byť náčrtok. Ako príklad môže slúžiť matematický výraz:

$$V = \begin{vmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \end{vmatrix}$$

Obrázek 1: Objem $V = (abc)$ kvádra, natiahnutého na vektory a, b, c sa počíta podľa vzorca, uvedeného vľavo.

```
\piccaptionside
\piccaption{Objem
$V=(abc)$ kvádra,
natiahnutého na
vektory $a$, $b$,
$c$ sa počíta
podľa vzorca,
uvedeného vľavo.}
\parpic{$ V= \left|
\begin{array}{lcl}
a_x & a_y & a_z \\
b_x & b_y & b_z \\
c_x & c_y & c_z
\end{array}\right|$}
```

Ako sa dá z tohoto príkladu vidieť, môžu byť obrázky označené nadpismi. Poloha nadpisu sa dá riadiť 4-mi príkazmi:

1. `piccaptionoutside` umiestni nadpis *pod* obrázkom, *mimo* rámčeka (tento spôsob je preddefinovaný).
2. `piccaptioninside` umiestni nadpis *pod* obrázkom, ale *vo vnútri* rámčeka.
3. `picccaptionside` umiestni nadpis *vedľa* obrázka, vertikálne centrovaný vzhľadom na výšku obrázka.
4. `piccaptiontopside` umiestni nadpis *vedľa* obrázka, ale zarovnaný na horný kraj obrázka.

> **Pozor!** Nadpis musí byť definovaný *pred* príkazom `\parpic`. Napriek tomu sa objaví pod, resp. vedľa obrázka.

Pri použití príkazu `\piccaption` sa registre pre obrázky a tabuľky zvyšujú rovnakým spôsobom ako pri originálnom LaTeXovskom príkaze `\caption` vo vnútri prostredí `figure` alebo `table`.

Obrázek 2: Kartové značky

Najčastejšie sa používa prvý prípad, teda nadpis pod obrázkom mimo rámčeka. V prípade, ak nebudú zadané žiadne voľby pre rámček, prvé dva prípady sa ničím neodlišujú.

```
\piccaptionoutside
\piccaption{Kartové
značky}
\parpic(2.5cm,1cm)%
[f]{\Karty}
Najčastejšie sa ...
```

Dvojstranné publikácie
Voľbami príkazu `\parpic` – `[l]` alebo `[r]` – sa umiestňujú obrázky na ľavý, resp. pravý okraj odstavca, nezávisle na tom, či je aktuálna strana ľavá alebo pravá. Režimom `\picchangemode`, ktorý bol vymyslený v prvom rade kvôli `twoside` dokumentom, ale je možné ho používať pri všetkých ostatných `sty`-voľbách, je možné umiestňovanie závislé na tom, či sa jedná o ľavú alebo pravú stranu. Príkaz `\picchangemode` otvára a príkaz `\nopicchangemode` uzatvára tento špeciálny režim. Účinok `\picchangemode` je nasledujúci:
Na *nepárnych* stranách (to sú obyčajne *pravé* strany) bude obrázok umiestnený *v súlade* so zadanou voľbou (teda `[l]` umiestni obrázok na *ľavú* stranu odstavca). Na *párnych* stranách (to sú obyčajne *ľavé* strany) bude obrázok umiestnený *na opačnej strane* ako je daná voľba (teda `[l]` umiestni obrázok na *pravú* stranu odstavca).

Obrázky medzi odstavcami: `\hpic`

Príkaz `\parpic` sa používa na spojenie obrázkov a textu. Pre obrázky, ktoré majú byť bez textu umiestnené medzi odstavcami, ponúka `PicIns` príkaz `\hpic`. Na rozdiel od príkazu `\parpic` je pomocou `\hpic` možné umiestnenie viacerých obrázkov vedľa seba.

Syntax: `\hpic`(*šírka,výška*)(*x-posun,y-posun*)[*voľby*][*pozícia*]{*Obsah obrázku*}

Všetky parametre vrátane *obsahu obrázku* sú voliteľné.

Popis parametrov:

šírka,výška, x-posun, y-posun, pozícia a *obsah obrázku* majú rovnaký význam ako v príkaze `\parpic` opísanom vyššie.

Pri voľbách sú k dispozícii:

`l` (*left*) — za sebou nasledujúce obrázky sa zarovnávajú na horný okraj
`r` (*right*) — zarovnávanie na spodný okraj
`f` (*frame*) — obrázok s rámčekom
`d` (*dash*) — čiarkovaný rámček okolo obrázku
`o` (*oval*) — rohy rámčeku budú zaoblené
`s` (*shadow*) — obrázok bude zarámovaný a s tieňom
`x` (*box*) — obrázok bude zarámovaný ako 3D-škatulka

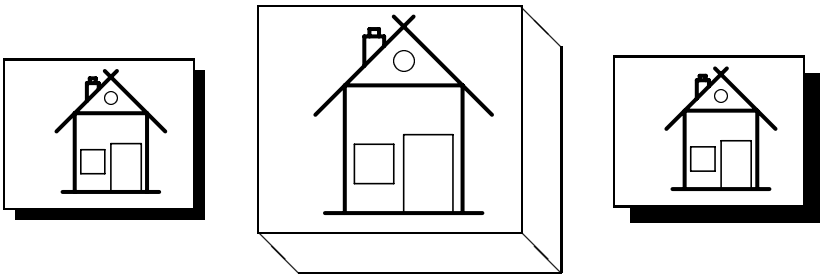
Bez parametra `t` alebo `b` budú viaceré za sebou nasledujúce obrázky vertikálne centrované, bez parametrov pre rámček budú bez rámčeka.

Nasledujúcim spôsobom:

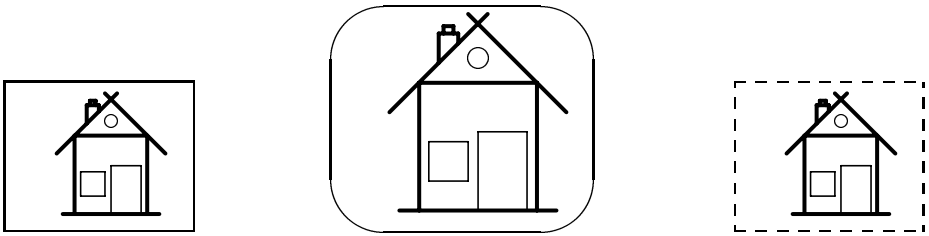
```
\centerline{%
```

```
\hpic(2.5cm,2cm)[s]{\copy\malydom}\hfil%
\boxlength{15pt}\hpic(3.5cm,3cm)[x]{\copy\dom}\hfil%
\shadowthickness{6pt}\hpic(2.5cm,2cm)[s]{\copy\malydom}%
}
```

bol vytvorený *riadok obrázkov*



Alebo ďalší riadok:



Obrázek 3: Účinok volieb `[fb]`, `[ob]` a `[db]` a zároveň ukážka príkazu `\newcaption`, ktorý text nadpisu formátuje so *zarovnávaním vľavo.*

Tento *riadok obrázkov* vznikol podobne ako vrchný, ale boli použité voľby `[fb]`, `[ob]` a `[db]`. Tak sa obrázky stali zarámovaným, zarámovaným oválom a zarámovaným čiarkovane. Nastavenie na ľavý a pravý okraj strany bolo dosiahnuté použitím príkazov `\hfill` medzi jednotlivými obrázkami, namiesto vyššie použitých `\hfil`.

Vovnútri jedného *riadku obrázkov* nesmú byť pomiešané voľby `t` a `b`, v opačnom prípade je možné očakávať neočakávané výsledky.

K *obrázkovému riadku* sa pod *najspodnejší* obrázok pripojí text.

„*Podpisy*“ pod *obrázkové riadky* môžu byť vyrobené LaTeXovským príkazom `\caption` alebo `PicIns` príkazom `\newcaption`, ale nie príkazom `\piccaption`. Príkaz `\newcaption` sa líši od príkazu `\caption` zarovnaním nadpisu, ktorý je možný viacriadkový. V príkaze `\newcaption` budú riadky nadpisu zarovnané *pod sebou* (pozri nadpis predchádzajúceho obrázku).

Orámované prostredia

Na zvýraznenie častí textu, je možné tieto umiestniť do jedného alebo do kombinácie viacerých prostredí. Ako voľba sa v prípade prostredí definuje *šírka* (bez tejto voľby bude použitá aktuálna šírka riadku, prípadne pri dvojstĺpcovom texte aktuálna šírka stĺpca). Je možné vnáranie s ostatnými prostrediami.

Syntax: `\begin{`*Názov prostredia*`}[`*Šírka*`]` ... `\end{`*Názov prostredia*`}`

Ako *názov prostredia* je možné použiť `frameenv`, `dashenv`, `ovalenv` alebo `shadowenv`.

Nasledujúci príklad ukazuje užšie, čiarkovane zarámované prostredie vovnútri prostredia s oválovým rámčekom.

1. `\begin{ovalenv}`
2. `\begin{center}`
3. `\begin{dashenv}[6cm]`
4. `\begin{enumerate}`
5. nasledujú mnohé body (`\item`y)
6. `\end{enumerate}`
7. `\end{dashenv}`
8. `\end{center}`
9. `\end{ovalenv}`

Orámovať sa dajú aj rovnice[5], napr. rovnica

$$\lim_{x\to\infty} \frac{\sin x}{x} = 0, \tag{1}$$

bola vytvorená príkazmi:

```
\begin{equation}\mbox{\raise7mm\hbox{\begin{minipage}{40mm}
\parpic(30mm,10mm)[s]{$\displaystyle\lim_{x\to\infty}
\frac{\sinx}{x}=0,$}\end{minipage}}}\end{equation}
```

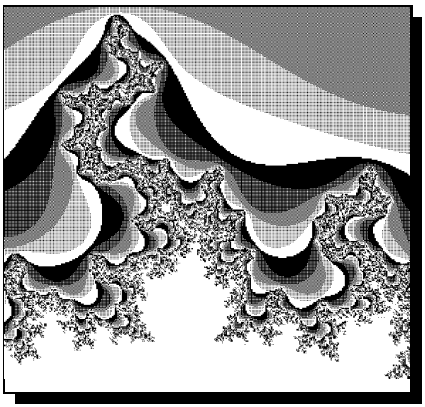
alebo

$$\lim_{x\to\infty} \frac{\sin x}{x} = 0, \tag{2}$$

príkazmi

```
\begin{shadowenv}\begin{equation}\lim_{x\to\infty}
\frac{\sin x}{x}=0,\end{equation}\end{shadowenv}
```

Zvlášť zaujímavá môže byť kombinácia makra `PicIns` s prostredím `figure`. Rámček môže zvlášť dobre zvýrazniť spolupatričnosť obrázku a textu. Príklad:



Tento obrázok je v kodovaní MSP, rovnako sa dajú zaradiť aj obrázky BMP, resp. PCX. Bol zaradený príkazom `\parpic(5.4cm,5.4cm)(0pt,0pt)[s]{\special{em:graph mandel.msp}}`
Tento príklad je prostredie `figure`, v ktorom sa nachádza prostredie `shadowenv`. Vo vnútri je po prvom odstavci použitý príkaz `\parpic`. Vzorec je obsah odstavca. Namiesto príkazu `\caption` je použitý príkaz `\newcaption`, v ktorom sa text podpisu zarovnáva.

Obrázek 4: Mandelbrotova množina. Je takmer neuveriteľné, že taká zložitá množina súvisí s jednoduchým zobrazením: $z \leftarrow z^2 + c$ v komplexnej rovine.

Rovnako je možná aj kombinácia s ostatnými prostrediami. V tomto príklade sme zmenili hrúbku rámčeku príkazom `\linethickness{4pt}`.

Tu končí voľný preklad cca. polovice príručky k makru `PicIns`. Ďalej nasledujú príklady použitia `picinpar.sty`.

Upozornenie! Pri spájaní dvoch častí textu mi načítanie súboru `picinpar.sty` spôsobilo pokazenie príkladov, používajúcich príkaz `\hpic`. Bolo to spôsobené tým, že v makre `picinpar.sty` je `\newdimen\hpic`. Tento som v celom súbore `picinpar.sty` nahradil novým rozmerom `\hpics` a všetko zafungovalo. Nevylučujem však aj nejaké iné prekrývania.

[5] Tieto dve ukážky zaradil prekladateľ na vyplnenie stránky.

PicInPar – Príklady

FRIEDHELM SOWA,

Heinrich-Heine-Universiät Düsseldorf,

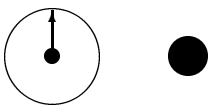
Email: sowa@convex.rz.uni-duesseldorf.de

```
\begin{window}[0,l,{\Huge P},{}]
\noindent
red niekoľkými rokmi publikoval ...
... odsek nasleduje automaticky.
\end{window}
```

Pred niekoľkými rokmi publikoval v časopise TUGboat Donald E. Knuth jeden malý problém s prosbou o pomoc pri hľadaní riešenia. Išlo o vytvorenie takého okna v odstavci, do ktorého by sa dal vložiť ľubovoľný text alebo obrázok. Čoskoro na to vyšli v nasledujúcich číslach návrhy na riešenie: jeden súkromne od DEK, ďalší od Alana Hoeniga. Posledne menovaný prišiel na elegantné riešenie, ktoré nepotrebovalo žiadne ďalšie manuálne korektúry. Jeho makro žiada len v parametroch informácie o výške a šírke miesta, vynechaného v odstavci. Vloženia a sadzba častí odstavca sa uskutočňujú automaticky.

```
\begin{figwindow}[2,r,{
\unitlength1cm
\begin{picture}(3,1.4)
\put(0.7,0.7){\circle*{0.2}}     \put(0.7,0.7){\circle{1.2}}
\put(0.7,0.7){\vector(0,1){0.6}} \put(2.5,0.7){\circle*{1.2}}
\end{picture}
},{Krúžky a~šípka}]
\noindent
Čo robia tieto makrá ...
... tu vidno ako to pokračuje:
\end{figwindow}
```

Čo robia tieto makrá? Určitú predstavu sme už získali na začiatku tohto článku: prvé písmeno odstavca bolo napísané v inej veľkosti a vložené do odstavca. Nemusí to však byť len text, ktorý sa takto vkladá. K odstavcu sa takto môže pripájať aj prostredie `picture`. Napravo uvádzaný príklad, ktorý vám môže byť známy z knižky „Úvod do LaTeXu“, bol zasadený do prostredia `minipage`. Zadanie v texte pritom vyzerá rovnako ako vyššie.



Obrázek 5: Krúžky a šípka

```
\begin{tabwindow}[1,r,{
\begin{tabular}[t]{|r|l|r@{:}l|}
\hline
1&HSV&12&0\\
...
\end{tabular}
},{Tabuľka}]
Ani tabuľku ...
... bude trocha presahovať.

Potom čo ...
... číslovanie
\end{tabwindow}
```

Ani tabuľku vložiť do odstavca nie je žiaden veľký problém. Vezmeme si (alebo predstavíme) nejakú tabuľku. Potom musíme dávať pozor na to, aby spodný riadok tabuľky bol zarovnaný so spodným riadkom textu. V opačnom prípade bude trocha presahovať..

1	HSV	12:0
2	MSV	11:1
3	VfB	10:2
4	SVW	9:3
5	1.FCK	8:4

Tabulka 1: Tabuľka

Potom, čo trochu potrápime klávesnicu a mozog, vyjde z toho niečo také ako je napravo. Dokonca aj označenie tabuľky je zvýraznené. Ale naozaj zaujímavé to bude, keď aj pri ďalšej tabuľke bude sedieť číslovanie.

```
\begin{tabwindow}[2,l,{
\begin{tabular}[t]{|r|l|r@{:}l|}
\hline
1&HSV&12&0\\
...
\end{tabular}
},{Tabuľka}]
Čo pôsobí ...
... povedal?
\end{tabwindow}
```

Čo pôsobí šialene uvoľňujúco, je neustála zmena polohy obrázku na jednej strane. Oko sa tak neunaví pri prezeraní typografických záhybov, obsiahnutých vo vytlačenom diele. Ale mali by sme sa sústrediť na podstatné informácie, spájajúce sa s prácami tohoto druhu.

1	HSV	12:0
2	MSV	11:1
3	VfB	10:2
4	SVW	9:3
5	1.FCK	8:4

Tabulka 2: Tabuľka

Ale čo sú teraz dôležité informácie? Áno, malé `1` (L) a `2` poskytujú úplne iný obraz reality, ako by mohli byť videné očami futbalového fanúšika šesťdesiatych rokov. Skôr ako nato zabudneme: neboli sme si istí, či budú tabuľky dobre číslované. Takže, kto by to bol o nich povedal?

```
\begin{tabwindow}[4,c,{
\begin{tabular}[t]{|r|l|r@{:}l|}
\hline
1&HSV&12&0\\
...
\end{tabular}
},{Tabuľka}]
\sloppy
Ale teraz ...
... dosť bolo príkladov.
\end{tabwindow}
```

Ale teraz bude všetko hnané na vrchol. Alebo mnohokrát dané do stredu. Áno, predsa ani tabuľka vo vnútri odstavca nie je žiaden veľký problém. Jeden problém predsa len existuje. Ako sa má čítať text. Najprv ľavý stĺpec a potom pravý, alebo jednoducho zľava doprava? Človek to nesmie často robiť nesprávne. To je predsa neobyčajne pomalé upozornenie v jednej odstavca je dobre mať nastavcami na `0pt` alebo pouvod? Rozostupy môzu byť kov a tak by bola opticky ľavo a napravo od okna tronič nenechalo na náhodu, vybaví to hneď `picinpar`.

1	HSV	12:0
2	MSV	11:1
3	VfB	10:2
4	SVW	9:3
5	1.FCK	8:4

Tabulka 3: Tabuľka

tešujúce! Teraz ešte jedno veci: pri tabuľke v strede stavenú medzeru medzi odžiť len jeden odstavec. Dônezávislé na rozostupe riadkorektná sadzba stĺpcov nachu namáhavá `:-)`. Aby sa

Teraz ešte prípad, pri ktorom má byť obrázok vložený do stredu textu, ale naľavo a napravo ostáva tak málo miesta ($<= 72pt$), že tam budú priveľké problémy s delením slov. Ako príklad bude v okne uvedený TEXovský vstup pre tento prípad. Aby sme zabránili prípadným problémom, budeme najskôr postupovať nasledovne:

```
\newbox\pppbox
\setbox\pppbox=\vbox{\hsize=9cm
\begin{verbatim}
\begin{figwindow}[4,c,{\wframepic{ppp}
},
{Vstup pre túto časť textu!}]
Tento postup
...
nestojí v~ceste.
\end{figwindow}
\end{verbatim}
}
```

Tento postup je potrebný pre použitie `figwindow` so vstupom opísaným po-

mocou `verbatim`. Ale je to aj tak jedno. Jednoducho je dôležité, či zistíme, či je naľavo a napravo od centrovaného obrázku ešte dosť voľného miesta, aby sme tam mohli sádzať text bez veľkých problémov. Voľba $72pt$ bola viacmenej ľu-

```
\begin{figwindow}[4,c,{\wframepic{ppp}
},
{Vstup pre túto časť textu!}]
Tento postup
...
nestojí v~ceste.
\end{figwindow}
```

Obrázek 6: Vstup pre túto časť textu!

bovoľne zvolená. Táto hranica nakoniec závisí aj na používanej reči. Keď teda na stranách nie je dosť miesta k dispozícii, tak sa s textom bude jednoducho pokračovať po okne. Ešte raz rýchly pohľad na číslovanie obrázkov.... Áno, sedí to presne. Teraz už nasadeniu (makra) nič nestojí v ceste.

Ale počkať! Tu sa predsa jedná o TEX. A to je dosť veľký dôvod na to, aby sme sa ešte krátko zastavili na sadzbe matematických výrazov. V okne rovnako ako v texte vedľa okna sa môže objaviť matematický výraz, ktorého rozmery pri sadzbe budú mať ďalekosiahly vplyv. Tu je predsa napravo `array` a tu nasleduje výraz:

$$F(b) - F(a) = \int_a^b \sum_{j=0}^{n} f(x_j) \prod_{\substack{k=0 \\ k\neq j}}^{n} \frac{x - x_k}{x_j - x_k}\, dx$$

Centrovaná sadzba vzorcov vedľa textu nie je doteraz s `picinpar.sty` možná, snáď ale jedného dňa príde. Vo všeobecnosti sa práve sadzba matematiky ukazuje ako veľký problém rozvoja, pretože sa tam používa veľa gleja (`glue`) v `box`e. Ale predsa to napoly ide aj tak.

Ján Buša
busaj@ccsun.tuke.sk

Balíček maker `cards` pro sazbu kartiček

Vít Zýka

Tento text seznamuje čtenáře s balíčkem[1] TEXových maker `cards` původně navržených pro sazbu navštívenek. Makra zajišťují rozmístění různě velikých kartiček (navštívenek) po stránce zadané velikosti, volitelný tisk rámečků a ořezových značek, dále podporu sazby na osvitce. Proměnný text na kartičkách stejného typu je separován do datového souboru. Makra umožňují navrhnout takřka libovolné množství typů (stylů) kartiček zároveň a volit mezi nimi parametrem v datovém souboru. Balíček tak má obecné využití pro sazbu kartiček stejných velikostí, jejichž vzhled si uživatel sám definuje včetně míst s proměnným textem. Makra jsou určena pro formát plain TEX, i když jsou použitelná i ve formátu LATEX[2].

1. Motivace vzniku a původní požadavky

Tato makra byla vytvořena s cílem umožnit rychlou sazbu navštívenek jednoho výzkumného týmu. Tyto navštívenky se vyznačují podobným designem, ale různými konkrétními údaji.

O tisku navštívenek pojednával článek Josefa Baráka s názvem „Vizitky v LATEXu“ otištěný ve Zpravodaji 4/96. Z textu vyplývá, že makra `A4CARDS` určitým způsobem definují vzhled navštívenek. Je to výhoda pro ty, kteří si chtějí rychle vizitky vyrobit a spokojí se se standardní grafickou podobou. Další omezení plyne z tisku jen na formát A4. Pokud požadujeme kvalitní tisk v tiskárně, je mnohem finančně výhodnější sázet na co největší formát, nejčastěji na A2. Je samozřejmě možné seřadit určitý počet hotových stránek formátu A4 na větší arch pomocí některých programů (např. psutils, emTEXové DVI ovladače), ale tento způsob má minimálně jednu nevýhodu: připravíme se o spoustu místa, protože každá stránka má svůj okraj, ořezové značky a vlastní rozmístění navštívenek.

Naše původní požadavky se dají shrnout do následujících bodů:

- Snadná změna údajů (např. změna telefonu).
- Výběr z několika variant pro přizpůsobení nestandardním požadavkům (jazykové mutace).

[1] Toto je popis v. 1.0, ze dne 10. 4. 1997. Jakákoliv podobnost s jinými „balíčky“ je čistě náhodná.

[2] Tuto možnost ilustruje i příklad v tomto článku. Musíme však oželet vnější vertikální ořezové značky. Proto pro skutečnou sazbu je výhodnější plainTEX, jak je popsáno v tomto článku.

- Tisk ve vysoké kvalitě v tiskárně i nouzový dotisk v běžných podmínkách.
- Snadný ořez řezačkou i nůžkami.
- Tisk na různě velký formát různě velkých navštívenek.
- Snadnost sazby (na úrovni znalosti obsluhy TEXu).
- Možnost tvorby libovolného vzhledu navštívenek (na úrovni programování v plain TEXu).

Výsledkem jsou makra, která jsou nezávislá na obsahu navštívenek, takže je můžeme zobecnit pro sazbu jakýchkoliv kartiček. Podle jejich velikosti makra sama zajistí optimální pokrytí stránky daného formátu.

2. Použití z pohledu uživatele

Pro sazbu kartiček potřebujeme čtyři soubory (v závorce jsou uvedeny jejich příklady obsažené v distribuci):

(1) řídící soubor (`cardmain.tex`, `akela.tex`),
(2) datový soubor (`cmp4-97.dat`, `akela.dat`),
(3) stylový soubor s definicemi vzhledů kartiček (`cmp4-97.sty`, `akela.sty`) a
(4) soubor se základními makry (`cards.tex`).

V tomto okamžiku budeme předpokládat, že již máme vytvořen stylový soubor (3). Protože soubor základních maker (4) většinou netřeba editovat, zbývá vytvořit řídící a datový soubor (1) a (2). Popíšeme si jejich strukturu v následujících odstavcích.

2.1. Řídící soubor a jeho parametry

Řídící soubor je ten, který necháme přeložit plain TEXem. Ačkoliv jeho struktura je velmi jednoduchá, nemusíme jej tvořit, stačí zkopírovat soubor `cardmain.tex` z distribuce a pozměnit jej dle našich představ. Jde vlastně o jakýsi formulář, do kterého na začátku vložíme soubor (4), změníme v komentář přepínače které nepotřebujeme, vložíme (3) a nakonec v makru (makrech) \`cards`{*soubor*} zadáme jméno souboru (2). Poslední tři kroky můžeme libovolně opakovat.

2.2. Datový soubor

Datový soubor obsahuje informace, které se mění na kartičkách stejného typu. Je členěn po řádcích; každý řádek obsahuje data o jedné konkrétní kartičce. Žádný, tedy ani poslední, řádek nesmí být prázdný ani komentovaný.

Jednotlivé údaje jsou odděleny znakem ‚|‘. První dvě položky jsou pevné a znamenají po řadě pořadové číslo definice vzhledu kartičky, která se má pro tyto údaje použít a počet těchto kartiček, který se bude tisknout. Další údaje

i jejich počet je již závislý na zvoleném stylovém souboru, přesněji na tom, jak je definováno makro \card v něm. Ale o tom až v další kapitole.

3. Použití z pohledu návrháře kartiček

Návrh vzhledu kartiček vyžaduje určité znalosti makrojazyka plain TEXu. Spočívá ve vytvoření stylového souboru (3) s definicí makra \card s následující strukturou:

```
\card#1|#2|...|#n\enddata{%
  \ifcase\the\style%
     % box s definicí 0. stylu
  \or%
     % box s definicí 1. stylu, atd. n krát
     % ....
  \fi}
```

Důležité je, aby výška a délka boxu každého stylu byla právě rovna konstantám \cardheight a \cardwidth. Hloubka boxu musí být nulová.

OBLAST	PŘEPÍNAČ	POPIS (*≡default)
Velikost kartičky	\cardwidth=	délka (*=9cm)
	\cardheight=	výška (*=5cm)
Velikost strany	\aIV *	tisk na formát A4
	\bIII	tisk na B3, zarovnáno na A3
	\bII	tisk na B2, zarovnáno na A2
	\LaTeXpage	tisk uvnitř LATEXu
Rámeček	\frame *	zviditelní rámeček
	\noframe	zneviditelní rámeček
Ořezové značky	\cutline	zviditelní ořezové značky
	\nocutline *	zneviditelní ořezové značky
Rozprostření kartiček po stránce	\fillstyle *	mezery mezi kartičkami
	\cutlinestyle	ořezové značky mezi kartičkami
	\nospacestyle	bez mezer
Zrcadlení	\mirror	zrcadlově obrácený tisk pro osvit
Barva (závislé na styl. souboru)	\black	separace černé barvy pro osvit
	\red	separace červené barvy pro osvit

Tabulka parametrů řídícího souboru podle funkčních skupin.

4. Jednoduchý příklad použití

Použití tohoto balíčku si ukážeme na následujícím příkladu. Představme si, že si chceme vytvořit kartičky ke hře Akela. Tuto hru vede jeden hráč, který postupně říká nějaké obecné pojmy jako např. *historická postava, řeka, ulice, rostlina, ap.* a úkolem ostatních je co nejrychleji říci konkrétní pojem z dané oblasti začínající na předem dohodnuté písmeno. Kdo jej vyřkne jako první, má bod. Účelem je mít co nejvíce bodů. Aby hra byla objektivní a měla spád, má vedoucí hry obecné pojmy napsané na kartičkách, které na začátku hry zamíchal. Hráči si pak nemusejí své body pamatovat, protože dostanou kartičku s pojmem, který řekli jako první. Pomocí maker `cards` si tedy připravíme kartičky k této hře.

Definice řídícího souboru (akela.tex). Nejdříve do něj vložíme soubor základních maker:

```
\input cards.tex
```

Zvolme velikost kartiček na 2,8×5 cm:

```
\cardheight=2.2cm \cardwidth=5cm
```

Rámeček okolo kartiček se tiskne standardně, chceme také tisknou ořezové značky a protože budeme kartičky stříhat doma nůžkami, budeme chtít ořezové značky i mezi kartičkami:

```
\cutline \cutlinestyle
```

Pak již zbývá vložit stylový soubor, nazvěme jej `akela.sty`, spustit sazbu makrem `\cards` s parametrem datového souboru `akela.dat` a vše standardně ukončit.

```
\input akela.sty \cards{akela.dat} \bye
```

Definice datového souboru (akela.dat). Pro každou kartičku použijeme první styl ve stylovém souboru (číslováno od 0) a budeme ji tisknout jen jednou. Definice prvních čtyř kartiček vypadá následovně:

```
0|1|Historická postava
0|1|Řeka
0|1|Ulice
0|1|Rostlina
```

Definice stylového souboru (akela.sty). Budeme požadovat, aby každá kartička měla třímilimetrový vertikální a dvoumilimetrový horizontální okraj kam nebude zasahovat text a v pravém horním rohu byl nápis „Akela“. Samotný název oblasti bude horizontálně vystředěn. Pro text použijeme českou variantu písma Computer Modern Dunhill Roman patřičně zvětšenou:

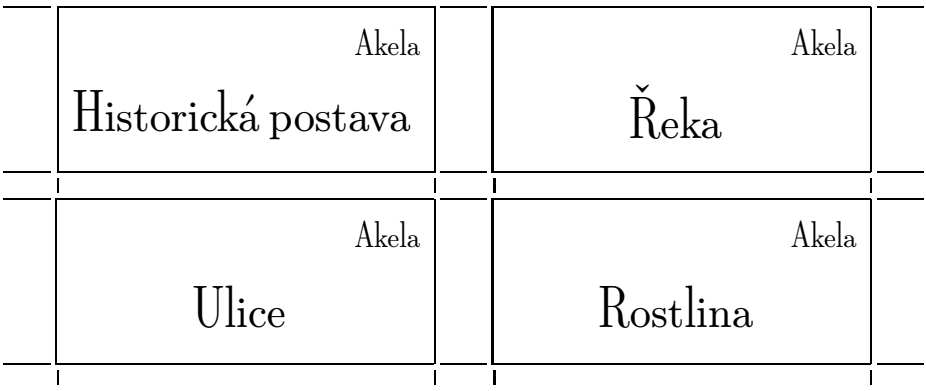
```
\newdimen\vborder \vborder=3mm
\newdimen\hborder \hborder=2mm
\font\spherefn=csdunh12 at 16pt
\font\otherfn=csdunh10
```

Nakonec definujeme makro `\card`, ve kterém jsou jednotlivé styly odděleny příkazem `\ifcase`. Makro bude mít jen jeden parametr a tím bude proměnný údaj názvu oblasti (*řeka*, ...) načtený z datového souboru.

```
\def\card#1\enddata{%
  \ifcase\the\style% ---------------------- Design 0
    \vbox to\cardheight{
      \vskip\vborder
      \hbox to\cardwidth{\hskip\hborder\hfil%
                  {\otherfn Akela}\hskip\hborder}
      \vfil\vskip2ex
      \hbox to\cardwidth{\hskip\hborder\hfil%
                  {\spherefn#1}\hfil\hskip\hborder}
      \vfil
      \vskip\vborder}%
    \fi}
```

Velmi důležitý je znak % na konci předposledního řádku. Pokud bychom ho nenapsali, zvětšila by se délka kartičky o mezeru a nesouhlasily by ořezové značky.

Nyní si můžeme prohlédnout výsledek prvních čtyř kartiček. Pokud si budeme chtít hru Akela skutečně vyzkoušet, nesmíme zapomenout vytvořit dalších alespoň padesát podobných kartiček. Dá nám to však poměrně málo práce, protože je budeme jen dopisovat do datového souboru. Pokud některou kartičku tisknout nechceme, nastavíme její počet na nulu. Komentář v datovém souboru užít nemůžeme.



5. Potisk samolepících štítků

Balíček obsahuje podporu pro potisk samolepících štítků dostupných na našem trhu. Jde o štítky pro laserové tiskárny rozprostřené na listu formátu A4 v několika řadách těsně vedle sebe. Velikosti štítků a jejich počet na jednom listu uvádí následující tabulka:

Štítek [*mm*]	Počet na A4	Makro
148.5×105	2×2=4	`\labIIxII`
74.25×105	4×2=8	`\labIVxII`
42.43×105	7×2=14	`\labVIIxII`
42.43×70	7×3=21	`\labVIIxIII`
37.13×70	8×3=24	`\labVIIIxIII`
21×64	13×3=39	`\labXIIIxIII`
21.21×52.5	14×4=56	`\labXIVxIV`

Pro sazbu je důležité znát:

- horizontální a vertikální rozměr jednoho štítku,
- šířku všech štítků na řádce a celkovou výšku všech štítků na stránce nebo počet sloupců a řad štítků na stránce,
- levé a horní odsazení štítků od okraje stránky.

Těmto údajům přizpůsobíme obsah řídícího souboru. Nejdříve načteme makra:

```
\input cards
```

Zadáme rozměr jednoho štítku jako rozměr kartičky (výška×délka, např. 21×64 milimetrů):

```
\cardheight=21mm \cardwidth=64mm
```

Definujeme rozměr všech štítků v řádce a sloupci. S výhodou můžeme využít násobení TEXu a zadat jej jako násobek velikosti jednoho štítku (3 sloupce a 13 řad).

```
\hsize=3\cardwidth \vsize=13\cardheight
```

Standardní levý horní roh tisku začíná 1 palec (2,54 cm) od levého i horního okraje stránky. Naše odsazení štítků bude pravděpodobně jiné, proto je třeba přičíst (odečíst) odpovídající vzdálenost (v našem příkladu je odsazení štítků 9 mm zleva a 12 mm zhora stránky):

```
\hoffset=-16.4mm \voffset=-13.4mm
```

Protože se i při nastavení `\nocutline` a `\nospacestyle` po obvodu stránky tisknou neviditelné ořezové značky, které by rozšířily stránku nad námi stanovené rozměry `\hsize` a `\vsize`, je nutné nastavit následující rozměry (odsazení ořezových značek od kartičky a délka ořezových značek) na nulové rozměry:

```
\cutlineskip=0pt \cutlinelength=0pt
```

Potlačíme tisk rámečků okolo štítků a vybereme rozprostření kartiček bez mezer:

```
\noframe
\nospacestyle
```

Soubor zakončíme standardním načtením stylového a datového souboru:

```
\input stitky.sty
\cards{stitky.dat} \bye
```

K balíčku je přibalen soubor `labels.tex` s předdefinovanými makry pro standardní archy se štítky, jejichž jména jsou uvedena v tabulce. S jeho použitím se náš řídící soubor zjednoduší na:

```
\input cards \input labels
\labXIIIxIII
\noframe \nospacestyle
\input stitky.sty
\cards{stitky.dat} \bye
```

6. Rámeček a ořezové značky

Vyvstává problém, kam vlastně umístit rámeček a kam ořezové značky. V podstatě jsou tři možnosti: (a) vnitřní hrana čáry se kryje s vnější hranou kartičky, (b) střed čáry leží na hraně kartičky nebo (c) se kryje vnější hrana s hranou kartičky. Jsou-li dostatečně slabé, je to asi ve většině případů nepodstatné. Někdy se však může stát, že na tom záleží. Současná verze implementuje variantu (c). Správnější by asi bylo sázet ořezové značky dle (b) a pozici rámečku umožnit zadat uživatelem. Je to námět na další vylepšení.

Balíček `cards` najdete na anonymním ftp serveru `cmp.felk.cvut.cz` v adresáři `/pub/cmp/users/zyka/cards. Bezproblémový tisk a hodně zábavy s Akelou přeje`

Vít Zýka, zyka@vision.felk.cvut.cz

4allTEX version 4

Erik Frambach

The fourth edition of the 4allTEX cdroms is now available. This article describes the new features of this edition.

Introduction

4allTEX is a ready-to-run system for MS-DOS, Windows and OS/2 users. Since 1994 this system is available on cdrom. In fact, back then it was the first TEX cdrom that offered true 'plug & play' for TEX. After this first revolutionary cdrom two more editions were produced, from which in total more than 5000 copies were sold worldwide, mostly of the third edition from 1995.

A new edition

Since 1995 a lot has changed in the MS-DOS and Windows world. Therefore we (Wietse Dol, Maarten van der Vlerk and me) thought it was time to produce a new edition which would completely up-to-date. An important aspect in this respect was the emergence of Windows 95 and the increasing popularity of Windows NT. These operating systems enabled us to implement several features that simply couldn't be done before under MS-DOS or Windows 3.x.

What's new?

Naturally we started off by updating all the relevant software that we supported on the old cdroms, and lots of new stuff that we found during the last two years. Of course the recently produced TEX Live cdrom was also a very helpful resource, for which we have Sebastian Rahtz to thank.

We copied many packages from that cdrom, updated things (it's amazing to see how fast updates of everything appear) and corrected some errors. Of course we added many other packages, fonts, programs and utilities that we thinks could be worthwhile for any TEXie to have at hand.

Here is a list of the most important differences between the third and the fourth edition of the 4allTEX cdroms:

- The installation script now recognizes the operating system on which 4TEX will run. This way you can have different configurations depending on

what OS you are running (MS-DOS, OS/2, WINDOWS 3.x, WINDOWS 95 or WINDOWS NT). The 4TEX batch files do not need any configuring, they automatically adapt to the OS, providing all features that are available on that OS.

- When running on WINDOWS 95/NT 4TEX can start other programs concurrently, so the menu remains available next to, e.g., a WINDOWS editor and a WINDOWS previewer. 4TEX is also able to automatically "kill" WINDOWS programs that you want to exit when 4TEX exits itself.
- 4TEX no longer supports computers equipped with a 80286 cpu or lower. If you have such a computer, you will have to manage with the standard EMTEX stuff.
- The spell-checker (still AMSPELL because ISPELL is still troublesome) now also supports South African, and of course the new Dutch spelling.
- 4TEX comes with and supports many more editors than before. E.g., the whole TSE family (formerly called QEDIT), PFE, WINEDT, NOTEPAD, EMACS, ZEUS, QUICKEDIT (not to be confused with QEDIT or QUICKEDITOR) and MULTI-EDIT.
- 4TEX runs on the newest version of 4DOS, 6.0, and uses lots of its new features.
- Chinese is now supported via the LATEX package CJK. Fonts and everything else you need is ready to use.
- Donald Knuth himself requested a possibility to make 4TEX run TEX interactively (the "TEX prompt"). This feature is now available by choosing `null` as the main TEX file.
- Several dvi drivers cannot handle virtual fonts. Therefore you need to "devirtualize" a dvi file before such a driver can use it. The program DVICOPY does that for you, automatically if you want. E.g., if you choose DVIWIN as your previewer 4TEX will call DVICOPY before it starts the previewer. This feature can also be useful when using DVISCR. If the dvi file contains lots of references to virtual fonts DVISCR can easily run out of memory. Devirtualizing can solve this problem.
- Documentation in HTML format can now be viewed even by MS-DOS users in graphic mode (i.e., with pictures) by choosing the browser ARACHNE. It looks so much better than DOSLYNX... WINDOWS users can choose their own browser (e.g., INTERNET EXPLORER or NETSCAPE) or OPERA, a quick and small yet very powerful competitor.
- The METAPOST menu is much more powerful now. GHOSTSCRIPT (MS-DOS or WINDOWS version) can be used to preview pictures.
- The list of supported output devices has grown. Completely new is the option to use GHOSTSCRIPT to produce PDF.
- 1200 dpi laserjets are now supported thanks to a 32-bit version of EMTEX's DVIHPLJ.

- The PostScript Type 1 versions of the Computer Modern and AMS fonts that were donated to the public domain recently are being used now instead of he older BaKoMa/Paradissa set.
- Conversion of *any*TEX to HTML through Gurari's TEX4HT is now fully supported.
- The ConTEXt format by Hans Hagen (Pragma) is available, including documentation.
- Gnuplot (both MS-DOS and Windows version) can be started from the 4TEX utilities menu.
- GSview (version 2.3), the Windows PostScript previewer based on Ghostscript (version 5.03) can now be selected as previewer.
- An up-to-date LATEX 2ε Windows help file can be consulted for information on LATEX commands.
- LATEXmac is available to make typing of e.g., formulas, symbols and complete LATEX environments a lot simpler. The WYSIWYG interface lets you choose what you need and it then simply pastes the code into your editor.
- The Windows program LATEXcad can be used to make drawings that can easily be incorporated into TEX or LATEX documents.
- 4TEX has been optimized for speed. Internally large pieces of code have been rewritten and/or simplified. Externally an enormous excellaration was achieved (up to a factor 2!) by giving the emTEX compiler faster access to files by reducing the number of subdirectories to a minimum (subdirectory searching costs lots of time, especially on slow media such as cdroms).
- Thanks to Ghostscript good quality public domain versions of all 35 standard PostScript fonts are now available.
- Both cdroms can be 'browsed' with any WWW browser thanks to a file `index.htm` in each directory, which contains links to all directories and files.
- In the root of both cdroms you will find complete listings of all there is on the cdrom (like `FILES.byname` on CTAN). In the emTEX root there is a file `ls-R` that can give fast access to all TEX sources to any Web2c user (be it an MS-DOS, Window, Unix user or whatever).
- The new cdroms are ISO-9660 compliant which makes them useable on almost any operating system. This is useful because most of the stuff is operating system independent. Windows users will find that they can see long file names (thanks to the 'Joliet' extensions). Especially on the second cdrom long files names are used, e.g. for WWW files with extension `html`.
- The complete 4TEX documentation is available on the cdrom in electronic versions (DVI, PostScript, PDF and HTML). Of course the documentation has been fully updated. The installation process is now more thoroughly explained.

- The installation script has been improved significantly. Now it's much easier to install 'all' on your hard disk, simply by choosing the modules that you need. Even *after* doing the installation it's easy to add modules without going through the whole installation process again.
- On the second cdrom you will find a wealth of programs and information which can be useful for any TEX user. Several distribution sets of other TEX user groups are available. Also other 'shells' such as emTEXgi and TEXtelMeXtel are available.
- The complete 'LATEX catalogue' is there in html, PostScript and pdf. The catalogue describes a huge set of LATEX packages.
- Information on user groups is as complete as we could get it. The new Greek user group has been added. Electronic magazines produced by several user groups are available.
- The TEX courses have been updated and extended. Now there is also a Spanish, a Polish and a Greek course.

Many thanks to...

Just like before many, many people have contributed to this new edition of 4allTEX, either by sending us files or programs, by giving hints for improvements and extensions, by sending bug reports, or by beta-testing. There are simply too many of them to list here, so we'll just say to all: many thanks!

Availability

Like before this new edition of 4allTEX will be sold by the Dutch TEX users group ntg. The price of the double cdrom including the paper manual will be the same as before: 60 Dutch guilders (currently about 15 US dollars). But you can also buy the cdroms without manual because the manual is also on the cdroms. In that case you pay only 30 Dutch guilders. Prices include shipping. Note that TEX user groups can get a significant discount if they buy larger quantities from the ntg and resell to their members.

More details about payment, distribution etc. will be made available through the usual channels, such as www, mailing lists and Usenet.

Erik Frambach
University of Groningen
The Netherlands
email: E.H.M.Frambach@eco.rug.nl

GUST Annual Meeting April 30–May 3, 1998, Bachotek (Brodnica Lake District, Poland)

The 6th annual meeting of the Polish TeX Users' Group (GUST) will be held in Bachotek, Brodnica Lake District, April 30–May 3, 1998. We invite TeX users, no matter from where, to join us. The theme of the meeting is

TEX – A STANDARD FOR SCIENTIFIC AND TECHNICAL PUBLISHING

As the theme suggests, we would like to concentrate this year on the practical aspects of the production of scientific-technical documents using TeX, but papers on related topics are not excluded. The list of possible topics includes:

- Tools: TEX-ware, editors, drivers etc. . . ;
- Specialized macros and formats;
- Multilingual publications: formats, macros, utilities;
- Fonts;
- Multimedia publications;
- PostScript, PDF, SGML, HTML, XML, MathML, tools and applications;
- Graphics, audio/video: tools and applications;
- New digital media;
- Paper publications vs electronic publishing;
- Textual databases;
- Typography issues;
- Standards;
- Other.

As always, the conference programme will be completed with tutorials covering issues concerning TEX, METAFONT, PostScript etc.

Paper submissions, proposals for tutorials and remarks relating to the programme of the meeting should be sent to the Programme Committee: Tomasz Przechlewski, Uniwersytet Gdański, ul. Armii Krajowej 119/121, 81-824 SOPOT, POLAND (`ekotp@univ.gda.pl`). Lectures in English are acceptable. The deadline for submissions is 31 March 1998.

For more information, please contact the GUST secretary, Jola Szelatyńska, Ogólnouczelniany Ośrodek Obliczeniowy UMK, Chopina 12/18, 87-100 Toruń, POLAND (`mjsz@cc.uni.torun.pl`).

LaTeXová kuchařka – doplněk minulé části

Zdeněk Wagner

V minulé části jsme se věnovali formátování obsahu. Tematicky k tomu patří i popis, jak byly vysázeny ukázky. Z prostorových důvodů se však tato drobnost do minulého čísla nevešla, uvádíme ji proto zde.

Pomocné soubory (s příponou `.toc`) jsme převzali již hotové z citovaných knih. Ze stylových souborů jsme ponechali jen části týkající se sazby obsahu. Každý příklad má tedy dva soubory, které jsme vypisovali s použitím makra `\verbatiminput`. Přitom bylo nutno rozdělit příliš dlouhé řádky. Pro vlastní ukázku bylo definováno prostředí `xmp` takto:

```
\newcommand\CSnosplit{\catcode`\-=12\relax} % csbul.sty
\newenvironment{xmp}{\makeatletter \CSnosplit
  \bigskip \hrule\nobreak\smallskip}{\par
     \nobreak\medskip\nobreak\hrule\bigskip}
```

Uvnitř prostředí tedy můžeme používat „@“ i minus v kategorii 12. Lze proto načíst odpovídající stylový soubor i pomocný soubor se záznamy pro obsah. Tím automaticky při sazbě zjistíme, že se na žádné makro nezapomnělo. Jak ovšem zjistíte, obsahují stylové soubory makra, která smíme použít pouze v preambuli. Pokud je přidáme do příkladů obsahů, bude se LaTeX zlobit. Ve stylu pro Mahájánské texty je to makro `\AtBeginDocument`. To se naštěstí stará o vložení záznamu do pomocného souboru, ale ten již máme hotov. Stačí tedy, když jej předefinujeme tak, že spolkne svůj argument. Na začátek prostředí `xmp` vložíme:

```
\let\AtBeginDocument\@gobble
```

Ve stylu pro knihy Tajemství Tibetu nám vadí dvě makra: `\DeclareOption` a `\ProcessOptions`. Situace je o něco složitější. Zde potřebujeme nasimulovat, že zpracováváme první svazek. Musíme tedy aktivovat příkazy, které se vyskytují za `\DeclareOption{Ruzenec}`. Proto změníme definice tak, aby `\DeclareOption` nadefinovalo nové makro (budou to `\Ruzenec` a `\Bardo`), a `\ProcessOptions` zavolá `\Ruzenec`. Definice vypadá takto:

```
\renewcommand*\DeclareOption{\@namedef}
\renewcommand*\ProcessOptions{\Ruzenec}
```

To je vše, co jsme pro sazbu příkladů potřebovali.

Zdeněk Wagner
wagner@mbox.cesnet.cz

Zpravodaj Československého sdružení uživatelů TEXu
ISSN 1211-6661

Vydalo:	Československé sdružení uživatelů TEXu vlastním nákladem jako interní publikaci
Obálka:	Bohumil Bednář
Počet výtisků:	710
Uzávěrka:	30. listopadu 1997
Odpovědný redaktor:	Zdeněk Wagner
Tisk a distribuce:	KONVOJ, spol. s r. o., Berkova 22, 612 00 Brno, tel. 05-740233
Adresa:	CSTUG, c/o FI MU, Botanická 68a, 602 00 Brno
fax:	05–412 125 68
e-mail:	cstug@cstug.cz

Zřízené poštovní aliasy sdružení CSTUG:

`bulletin@cstug.cz`, `zpravodaj@cstug.cz`
korespondence ohledně Zpravodaje sdružení

`board@cstug.cz`
korespondence členům výboru

`cstug@cstug.cz`, `president@cstug.cz`
korespondence předsedovi sdružení

`cstug-members@cstug.cz`
korespondence členům sdružení

`cstug-faq@cstug.cz`
řešené otázky s odpověďmi navrhované k zařazení do dokumentu CSFAQ

`secretary@cstug.cz`, `orders@cstug.cz`
korespondence administrativní síle sdružení, objednávky CD-ROM

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

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

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

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