Drsný úvod do LaTeXu

aneb

Moc se s tím nemazlete

Obsah

1 Úvod k TEXu

Program TEX je typografický systém pro sazbu pěkných knih, především technických publikací, což znamená, že je určen pro zpracování textů v tiskové podobě, nikoliv „obrazovkové." Byl vytvořen tuším v roce 1983 D. E. Knuthem a je volně šiřitelný (což nemusí platit o všech jeho instalacích). Jeho hlavní silnou stránkou je sazba matematických výrazů, v čemž mnoho jiných komerčních systémů poněkud pokulhává (např. MS Word). Slabou stránkou je grafika[1], neboť TEX byl konstruován jako systém s výstupem **nezávislým** na výstupním zařízení (tiskárně, plotru, ...), zatímco grafika (především bitmapová – rastrová) je stále na zařízení závislá. Nicméně program umožňuje externí zařazení obrázků, především PostScriptových, nebo vytvořených METAFONTem. V případě potřeby začlenit do textu obrázky se nejčastěji výstupní soubor převádí do formátu PostScript nebo Pdf.

Další výhodou TEXu je jeho nezávislost na operačním systému, kterou je nutno chápat tak, že zdrojový text pořízený v jednom operačním systému bude (až na malé výjimky – především obrázky) přeložitelný a zpracovatelný i v jiném systému. Existují totiž instalace pro mnoho systémů, např. Dos, Unix, Apple, Windows, ... Ony výjimky jsou dány např. tím, že ve zdrojovém textu použijeme cestu k jinému souboru a její zápis se může lišit pro různé operační systémy (tato poznámka platí především pro parametry příkazu `\special`), popř. jde o českou libůstku – volbu kódování, což lze ovšem snadno překonat.

V některých případech se může za výhodu považovat také možnost oddělit zdrojový text (tedy ten, ve kterém je možno provádět úpravy vzhledu) od tisknutelného textu (ten se může šířit elektronickou cestou, aniž by někdo mohl do jeho vzhledu jednoduše zasáhnout). Rovněž tak se jeví výhodnou nízká spotřeba paměti na zdrojový soubor a stabilita systému (TEX je stále stejný, jen se opravuje těch pár chybiček, co v něm bylo – tedy na instalaci systému z roku 1990 si klidně můžete přeložit text napsaný pro instalaci z roku 2000; nejsou tedy vyvíjeny žádné nové verze se záměrnou nekompatibilitou souborů směrem nahoru).

Protože však pokrok v počítačích jejich možnosti od roku 1983 značně rozšířil, je vhodné uvažovat o následovníku TEXu. Na něm se již pracuje, dochází k přepisování původního algoritmu do jazyku Java, takže se nemusíte obávat, že se budete učit něco, co bude za pár let definitivně staré. Nový systém je totiž tvořen se značným důrazem na kompatibilitu s TEXem, který jde až tak daleko, že se binárně porovnávají přeložené (Dvi) soubory obou systémů a musí být zcela shodné.

Poznámka: Pro označení svého systému použil autor prvních písmen řeckého $\tau\varepsilon\chi\nu\eta$, znamenajícího umění. Proto je i název řecký (vybraná tři písmenka vypadají stejně v latince i řeckém písmu) a čte se [tech], nikoliv [teks]. Autor si rovněž vymyslel jeho „logo" ve tvaru „TEX". V prostředcích, kde není toto logo dostupné, se doporučuje psát `TeX`, aby bylo jasné, o co jde (existuje prý ještě editor s názvem Tex).

1.1 TEX, LATEX, teTEX, ...

Na program TEX je nutno nahlížet jako na programovací jazyk s příkazy umožňujícími efektivní sazbu textu. Tyto příkazy se nazývají **TEXprimitiva** a jsou pro přímé použití poněkud těžkopádné. Proto sám autor vytvořil **plainTEX**, což je soubor maker (makro – blok příkazů vystupujících a použitelných jako jeden celek), který umožňuje lehčí formátování textu. PlainTEX je základní a mnohdy používaný, vyžaduje však hlubší znalosti; tento text se jím proto nezabývá. Pro označení takto rozsáhlých základních souborů maker (čítajích tisíce řádků) se používá slovo **formát**.

Druhým užívaným formátem je **LATEX**, jehož základem jsou bloky, tzv. prostředí – v textu se vyskytuje spousta párů příkazů `\begin{...}` a `\end{...}`. Tento systém je vhodnější pro začátečníky, protože uživatel se stará pouze o to, **co** chce vysázet, **jak** se to vysází už je záležitostí LATEXu (má zabudována některá typografická pravidla, např. o kolik má být větší nadpis kapitoly či jiné sekce než běžný text, aby to vypadalo dobře). Tento formát také umožňuje vkládání jednoduchých obrázků čárové grafiky přímo v zdrojovém souboru. Malou nevýhodou je, že formát se vyvíjí a existují jeho různé verze, přičemž starší LATEXy nerozumí novým příkazům. To se týká především používání balíčků a rozšířené možnosti volby řezu písma (NFSS). Druhou nevýhodou je, že většina dokumentů psaných v LATEXu vypadá „stejně", nepoužijeme-li speciálních balíčků či vlastních úprav.

Existují i další formáty. Jmenujme například $\mathcal{AMS}$-**TEX**, který vytvořila Americká matematická společnost a umožňuje velmi kvalitní sazbu matematiky včetně všech podivných písmen v matematice se vyskytujících nebo $\mathcal{AMS}$-**LATEX**, který je kombinací předchozího a LATEXu (opět používá prostředí).

[1] Není vhodný například pro sazbu reklamních letáků a podobných tiskovin, práce by v těchto případech byla příliš těžkopádná.

Jednotlivé formáty byly napsány jako zdrojové texty pomocí primitivů TEXu. Protože tyto soubory jsou dlouhé, musí se nejprve provést tzv. inicializace formátu, při které se tyto soubory v textovém tvaru načtou, přeloží a stav paměti (nastavení proměnných, příkazů, ...) se uloží v binárním tvaru do souboru `*.fmt`, z něhož se data načtou při zahájení překladu vámi napsaného dokumentu. V systému pak existuje množství spustitelných souborů TEXu (např. `csplain`, `cslatex`), které „ovládají“ právě jeden formát.

Pokud prahnete po nějakém přirovnání ve vztahu TEXu a LATEXu, a umíte pracovat s příkazovým řádkem, lze uvést toto srovnání: V příkazovém řádku používáte spoustu velmi jednoduchých příkazů občas s komplikovanou syntaxí (způsobem zápisu parametrů), i k provedení jednoduchých akcí je potřeba spousty příkazů. Tuto úroveň můžeme položit za rovnu úrovni TEXu. Chcete-li nějaké úkony často provozovat, máte ve většině operačních systémů možnost si vytvořit **dávkové soubory** (skripty). Tyto soubory samozřejmě pracují s původními příkazy, ale běžný uživatel je nevidí, ten pracuje jen s názvy těchto dávek, jimž předává nějaké parametry a které považuje za **příkazy jiného systému** – s trochou nadsázky teda lze říci, že jsme nad původním systémem vytvořili nějaký nový „formát“, pomocí něhož systém ovládáme. A tuto úroveň můžeme považovat za úroveň LATEXu. Je rovněž zřejmé, že můžeme vytvořit velké množství souborů tak, že budou seskupeny do navzájem nezávislých oddělení, i když budou vykonávat občas stejnou funkci (seskupení rozuměj tak, že v rámci jedné skupiny mohu při vytváření dávkového souboru použít jméno již dříve vytvořené dávky, zatímco v druhé skupině je tato dávka nepřístupná); rovněž tak vzniklo množství formátů TEXu, které byly zmíněny.

Pokud s TEXem chcete pracovat, budete potřebovat nejen spustitelný program, ale mnohé další soubory, které obsahují příslušné formáty, doplňující balíčky, fonty apod. Všechny soubory dohromady se pak označují jako **implementace** či distribuce TEXu. Jako příklad lze uvést distribuci **teTEX**, která je určena především pro Unixové systémy, distribuci **Web2c**[2], distribuci **emTEX** pod systémem OS/2 a MS DOS, nebo komerční implementaci **VTEX**. V současné době je hodně aktivní **TEXLive**, což je souhrnná distribuce pro mnoho operačních systémů, která je založená na teTEXu a obsahuje i soubory pro spouštění TEXu přímo z CD, bez instalace na disk.

1.2 Uspořádání souborů

Celý systém TEX je tvořen velkým množstvím různých souborů, které jsou nutné ke smysluplnému chodu sytému. Z nich můžeme jmenovat tyto spustitelné (v závorce je uvedena běžně používaná přípona vstupního a výstupního souboru)

- vlastní program TEX, který překládá zdrojový soubor (`tex`; `dvi`, `log`, `toc`, `aux`, ...);
- program METAFONT, který vytváří grafický (a rastrový) tvar používaných fontů (`mf`; `*gf`, `tfm`);
- zobrazovače, které za použití fontů zobrazí text tak, jak ho poskládal TEX (`dvi`);
- editor, ve kterém je vytvářen zdrojový soubor a který jej ukládá v ASCII[3] tvaru na disk, tento editor není vlastní částí systému TEX a může být jakýkoliv, jen nesmí vkládat své vlastní řídící znaky (vyhovuje dosovský QEdit, EdLin, windowsovský NotePad, i většina vyšších editorů umožňuje export v ASCII). Zpravidla se používají speciální editory, které umí pomocí zkratkových kláves vkládat příkazy TEXu (například Q, CsEd, Vim, Emacs);
- pomocné programy pro METAFONT, které zajišťují např. komprimaci fontů apod.;
- pomocné programy pro TEX, které zajišťují jeho konfiguraci;
- program METAPOST, který slouží k tvorbě vektorových obrázků (`mp`; čísla);
- další programy, které se používají pro danou instalaci ke zvětšení komfortu (např. změna kódování češtiny, kreslící programy, abecední třídiče pro rejstříky, ...);
- dávkové soubory, které si vytvořil sám uživatel, aby to měl snazší.

Další hromada souborů, které systému TEX náležejí, jsou buďto nespustitelné vstupní soubory (zpravidla textové), které jsou využívány už dříve uvedenými spustitelnými soubory, nebo výstupní soubory, které některý ze spustitelných programů produkuje a které využívá další program nebo uživatel. Z typických nespustitelných souborů lze uvést tyto:

- zdrojový soubor – zpravidla s příponou `tex`, který obsahuje obyčejný text s požadovaným textem a formátovacími příkazy;

[2] Tato distribuce obsahuje především binární soubory a slouží jako základ pro jiné distribuce.

[3] Označení ASCII znamená jen 7-bitové kódování dle příslušného standardu. Zde jím budeme poněkud nepřesně rozumět jakékoliv 8-bitové kódování, které je v intervalu 0–127 shodné s ASCII.

- zdrojový soubor METAFONTu s příponou `mf`, který obsahuje v textovém tvaru pokyny, jak mají vypadat jednotlivé znaky a jaké mají rozměry (pro různé velikosti písma mohou existovat různé soubory `mf`);
- metrický soubor `tfm`, který obsahuje rozměry obdélníku (boxu) souvisejícího s šířkou, výškou a hloubkou znaku[4] (ne jeho vzhled), s tímto souborem pracuje TEX, aby mohl určovat délky řádků apod., soubor je stejný pro jeden font všech velikostí;
- soubor s rastry znaků `*gf`, `*pk`, které obsahují přímo vzhled znaků v rozlišení `*`. Soubory `gf` jsou zpravidla rozsáhlé, proto se komprimují do souborů `pk`. Tyto soubory jsou nezbytné pro funkci zobrazovače, většina zobrazovačů umí pracovat s oběma typy;
- soubory `fmt`, které už byly zmíněny a obsahují binární zápis jednotlivých formátů;
- stylové soubory `sty` v textovém tvaru, které se používají (nejen) v LATEXu a které obsahují definice nových příkazů, které buď nechcete umísťovat přímo do zdrojového textu, nebo které mění různé parametry. Jako příklad mohou sloužit styly `czech`, který zavádí používání češtiny v textech, předefinuje některé příkazy (např. příkaz `\today` bude nyní psát české datum, nikoliv anglické), nebo styl `a4`, který upraví rozměry textové oblasti tak, aby odpovídaly evropskému formátu papíru A4;
- třídové soubory `cls`, které v LATEXu definují základní vzhled dokumentu;
- přeložené soubory `dvi`, které obsahují binární informaci pro zobrazovač, kam má který znak v sazbě umístit, tento soubor je cílem celého snažení;
- protokol o překladu s příponou `log`, který obsahuje chybová a pomocná hlášení, která byla produkována během zpracovávání některého souboru (např. které soubory byly vloženy, kde nastaly chyby, které fonty se nepovedlo otevřít, kde se nepodařilo dodržet všechny požadavky uživatele);
- další běžně používané soubory vytvářené překladem jsou uvedeny v tabulce

Přípona	**Význam a obsah souboru**
aux	křížové reference, citace
toc	položky automaticky produkovaného obsahu
lof	seznam obrázků
lot	seznam tabulek

- mnoho dalších souborů (`clo`, `fd`, ...).

1.3 Schéma překladu

Zde bude uvedeno jednoduché schéma, jak postupovat při překládání souboru. Toto schéma obsahuje jen typické rysy, ne všechny podrobnosti. V mnoha prostředích pro práci s TEXem jsou některé kroky realizovány současně stiskem jednoho tlačítka.

1. **Napsání zdrojového souboru** v některém ASCII editoru. Tento soubor obsahuje veškerý požadovaný text, příkazy upravující způsob jeho formátování a odkazy na vkládané soubory.
2. Spuštění METAFONTu. Že je nutno zařadit i tento bod poznáte zpravidla až tehdy, kdy se pokusíte o bod následující, nicméně logicky sem patří. Tento krok je nutný **jen tehdy**, jestliže ve svém programu použijete font (dodaný či Vámi vytvořený), k němuž není vytvořen soubor `tfm`[5] – ve většině implementací TEXu však dojde k vytvoření metriky a fontu automaticky a nemusíte se o to starat, ve starších je zpravidla potřeba ruční práce – když spustíte překlad zdrojového programu a nemáte k dispozici `tfm` soubor pro daný font, překladač TEXu vyhlásí chybu.

   Ke spuštění METAFONTu musíte nastavit následující parametry:

   - jméno vstupního souboru s fontem (přípona `mf` se nemusí nezadávat),
   - mód překladu, což je typ výstupního zařízení, např. `laserjet` laserová tiskárna s rozlišením 300 DPI a `ljfour` laserová tiskárna s rozlišením 600 DPI,
   - zvětšení `mag`, které udává stupeň zvětšení fontu, bývá zpravidla 1, někdy je vhodnější udat jej pomocí `magstep(`n`)`, kde n je číslo značící stupeň zvětšení (pokud nenastavíte některý ze dvou předchozích parametrů, vygeneruje se font pomocí implicitního nastavení, které může být závislé na implementaci).

   Vlastní spuštění probíhá tak, že spustíte METAFONT s danými parametry, čímž dojde k vytvoření souborů `tfm` a `*gf`, kde hvězdička zastupuje velikost rozlišení. Zpravidla se pak provádí ještě konverze fontů z formátu `gf` do formátu `pk`, který má menší velikost.

[4] Kresba znaku může z boxu libovolně vyčnívat.

[5] Tento soubor je nutný, aby systém mohl poznat rozměry znaků a mohl vůbec text sázet (potřebuje určit, jakou *skutečnou* délku daný text zabírá, protože šířka jednotlivých písmen se samozřejmě liší).

Poznámka: V Unixových implementacích TEXu spusťte METAFONT v případě potřeby příkazem `mf '\mode:=laserjet; mag:=1; input soubor'`, kde příslušně upravíte mód, zvětšení a jméno souboru.

3. Nastavení formátu překladu. Zde si v případě integrovaného rozhraní (např. TEXSHELL, TEXNICCENTER, KILE) vyberete, v jakém formátu je zdrojový text napsán (LATEX, plainTEX, ...).
4. **Vlastní překlad**. Spustíte TEX (zvolením akce v integrovaném rozhraní nebo zadáním příkazu) a necháte se unášet pěknými hláškami, které systém produkuje. Pokud překlad proběhl bez chyb (**Pozor!** Ještě nemáte vyhráno, přeložený text vůbec nemusí vypadat tak, jak chcete, pouze syntaxe je správná.), na obrazovce se objeví hlášení o tom, kde byl výsledek překladu zapsán, kolik bylo stránek a kde je hlášení o průběhu překladu. Nyní můžete přistoupit k následujícímu bodu. Pokud překlad v pořádku není, objeví se nějaké chybové hlášení charakterizované otazníkem, nad nímž jsou uvedeny řádky, které oznamují, co se programu nelíbí (případně co navrhuje – např. `Missing $ inserted`), na další řádce bude číslo řádku ve zdrojovém souboru a jeho text až do místa, kdy se objevila chyba. Zbytek chybného řádku bude pokračovat v témže místě o řádek níž a na novém řádku se objeví otazník. V tomto případě můžete buď překlad natvrdo ukončit vypsáním a odesláním znaku `x`, nebo přímo vepsat upravený příkaz se správnou syntaxí. Další možností je odeslat prázdný řádek, překladač se pokusí překlad dokončit a použije jim navržené řešení. Zpravidla toto řešení nestačí, ale je vhodné k tomu, abyste podle přeloženého textu poznali, ve kterém místě jste udělali chybu (nejčastěji chybí označení konce matematického prostředí a vše od jeho začátku je pak psáno matematickou italikou bez mezer apod.).

   Další „chybovou“ hláškou může být znak `*`, který signalizuje, že překladač už dopřekládal a nějak nedošlo ke konci. Tato chyba nastane, zapomenete-li ukončit dokument příkazem `\end{document}` nebo tehdy, byl-li spuštěn TEX bez jména souboru, který má přeložit. Můžete zde dopsat jakýkoliv příkaz TEXu, zpravidla `\end{document}`. Pokud jste nezadali jméno zpracovávaného souboru, můžete napsat `\input jmeno_souboru.tex` (pozorně si prohlédněte, kam v tomto případě překladač uložil výstup). Můžete rovněž stisknout CTRL+C, čímž se běh přeruší a dostanete otazník, pak zadáte `x`.

   Posledním typem hlášek je nenalezení souboru, které se ohlásí chybovým textem a řádkem ukončeným dvojtečkou, na nějž máte napsat jméno požadovaného souboru. V případě, že udáváte i cestu, vyzkoušejte oba typy lomítek. Chcete-li tuto hlášku zrušit, stiskněte CTRL+Z (symbol konce souboru, pro Unix platí zpravidla CTRL+D).

   Poznámka: Pokud v programu používáte reference (odkazy na jiné části) nebo automaticky vytvářený obsah, musíte překlad vyvolat několikrát (třeba třikrát), jinak nebudou odkazy „sedět“. Tam, kde zatím nebyl odkaz dosazen, se objeví dva otazníky, nevytvořený obsah bude obsahovat jen nadpis „Obsah“ a nic víc. O tom, že je třeba nechat proběhnout překlad ještě jednou, jste informováni hláškou `Label(s) may have changed. Rerun to get cross-references right`.
5. **Prohlédnutí výsledku**. K prohlédnutí souboru slouží zobrazovač pro obrazovkový výstup. Pokud jsou k dispozici všechny fonty, přeložený text se okamžitě zobrazí a je možno si ho prohlížet (většina prohlížečů zobrazuje jen jednu stránku, k přechodu mezi stránkami slouží klávesy PAGE UP a PAGE DOWN, k posunu šipky a mezerník, ke zvětšení a zmenšení obrazu mohou sloužit klávesy + a − na číselné klávesnici, program se zpravidla ukončí stiskem Q, některé prohlížeče umožňují i vyhledávání v textu). Pokud fonty chybí, je zpravidla zavolán METAFONT (u starších systémů budete dotázání, zda se má vytváření fontů spustit, protože generování může zabrat hodně času).

   Poznámka: Pokud jste překlad po chybě natvrdo přerušili, bude náhled k dispozici pouze po poslední celou stránku před přerušením.
6. Oprava chyb. Nyní po prohlédnutí celého výstupu se zpravidla opakuje celý cyklus včetně editace zdrojového souboru, ve kterém opravíte veškeré chyby[6].
7. Vytištění hotového textu. Před vlastním tiskem nejprve musíte vybrat vhodný typ tiskárny a pak až lze kvalitně tisknout. Některé zobrazovače umožňují tisk přímo, u jiných systémů musíte nejprve soubor `dvi` převést do jiného formátu (např. `dvips` převádí do POSTSCRIPTu, `dvilj` převádí do jazyka PCL, používaného v laserových tiskárnách HP). Některé systémy umožňují ještě nastavit některé parametry, např. oboustranný tisk (zda tisknout liché, sudé nebo všechny stránky), volbu zmenšení či zvětšení, otočení apod. Spuštění tisku může vyvolat novou generaci fontů.

Poznámka: Pokud nepoužíváte či nemáte systém s nabídkami, ale musíte vypisovat přímo příkazy, jsou uvedeny zde. Ke spuštění překladu použijte příkaz tvořený jménem formátu s předponou `cs` (tedy `csplain` pro plainTEX, `cslatex` pro LATEX apod.), jehož prvním parametrem bude název souboru. K prohlížení výsledku můžete použít `xdvi`, `kdvi` (pro Unix) nebo třeba `tkdvi`, pro Windows pak třeba `windvi`.

[6] Nebo si to alespoň budete myslet.

2 Sazba běžného textu

V této kapitole budou uvedeny základy sazby textu a úvodní vzhled dokumentu, popsány první příkazy.

Protože je TEX systém pro úpravu textů, musí mít příkazy, které se v něm vyskytují v průběhu běžného textu, nějaký zvláštní oddělovač, aby je mohl překladač rozeznat. Tento oddělovač je obrácené lomítko „\“ (na české Windows klávesnici dosažitelné stiskem Ctrl+Alt+Q), které předchází **každému** příkazu. Kromě něj jsou ještě další speciální znaky, které mají jiný význam než textový. Nejvýznamnějšími takovými znaky jsou složené závorky užívané pro vyznačení skupin `{}`, znak `&` používaný v tabulkách apod.

Všechny příkazy mají tvar zpětného lomítka a dalšího řetězce textu. Příkazy jsou citlivé na velká a malá písmena. Příkaz musí být z pravé strany oddělen od textu buď mezerou, nebo jiným „nepísmenným“ znakem (např. čárkou, číslem) apod. Pokud mají příkazy nějaký parametr, považuje se za něj první znak po příkazu (mimo mezeru). V případě příkazů, které akceptují více parametrů, berou se za jednotlivé parametry postupně jdoucí znaky (vždy jeden znak je jeden parametr), tedy všechny parametry musí být uvedeny ve správném pořadí. Chceme-li, aby jako parametr vystupoval delší znakový řetězec, musíme ho uzavřít do složených závorek. Dalším typem příkazů jsou příkazy s **volitelnými** parametry, které není nutno uvádět a které specifikují další požadavky uživatele. Takovéto volitelné parametry je nutno uzavřít do hranatých závorek.

2.1 Minimální struktura textu

Protože LaTeX je strukturovaný, vyžaduje vyznačení začátku, konce a typu dokumentu (je trochu ukecaný). Minimální struktura proto vypadá například takto (řádky začínající procentem obsahovat nemusí):

```
%hlavička
\documentclass[12pt]{article}
\usepackage{czech,a4}
%zde je preambule
\begin{document}
%prázdný dokument
\end{document}.
```

Tato ukázka toho moc nevypíše, je prázdná. Můžeme si na ni však vysvětlit některé první příkazy.

- `\documentclass[`*volby*`]{`*třída*`}` – je úvodní příkaz LaTeXovských dokumentů. Parametr *třída* může nabývat především hodnot `report`, `article`, `book` a `letter`[7]. Tyto parametry určují, jak bude text formátován – nastavují velikost stránek, typ číslování, záhlaví, úrovně oddílů (zda budou kapitoly, dodatky apod.) a také definice příkazů. Ve většině případů se vystačí se stylem `article`, který dává nejrozumnější členění – nejvyšší oddíl je sekce, záhlaví není, formát stránky je Letter, číslo stránky je dole uprostřed, ... Parametr *volby* je volitelný a určuje doplňkové vlastnosti stylu (např. základní velikost písma – `10pt` (implicitně), `11pt`, `12pt` či velikost stránky – `letterpaper` (implicitně), `a4paper`, `a5paper`).
- `\usepackage[`*volby*`]{`*balíček*`}` – příkaz načte určitá rozšíření (tzv. *styly*), která buď rozšiřují možnosti LaTeXu (nové definice, nové fonty, podpora grafiky), nebo mění jeho základní nastavení (např. pro „sjednocení“ struktury textu více autorů při sazbě článků do časopisu). Zde jsou uvedeny styly pro sazbu českého textu (zavádí české nadpisy, popisy obrázků, uvozovky, rejstříky, české datum, ...)[8] a pro velikost stránky A4. Obecně může být uvedeno jakékoliv jméno souboru s příponou `sty`, problémem může být vznik nekompatibility, neboť některé nestandardní balíky (zvláště vlastní) se musí přenášet zároveň se zdrojovým souborem.
- `\begin{document}` označuje začátek úseku, v němž se bude vyskytovat vlastní text.
- `\end{document}` ukončuje úsek textu; tyto příkazy musí tvořit pár a být v dokumentu jen jednou, vše mimo ně je ignorováno (vyjma příkazů před `\begin{document}`).

[7] Další možnosti jsou uvedeny v dokumentaci `usrguide.dvi`.
[8] České fonty se zpravidla zavádějí už při vytváření formátu.

- `%` je jedním ze speciálních znaků, uvozuje poznámku a platí až do fyzického konce řádku (do stisknutí ENTER), vše za ním je ignorováno.

Sekce označená v příkladu jako hlavička rozhodně není povinná, ale vyplatí se ji používat k označování souborů. Každý řádek v hlavičce musí začínat jako poznámkový, znakem `%`, který se často používá i ke „grafické“ úpravě hlavičky. V hlavičce může být uveden název dokumentu, autor, použitý formát (LaTeX, PlainTeX, ...) a hlavně použité kódování češtiny.

2.2 Psaní textu, velikost a řezy písma

Psaní textu je záležitost velmi jednoduchá, stačí pouze psát (a vědět, co psát). Je ovšem nutno pamatovat na pár věcí:

1. v textu se nesmí vyskytovat některé znaky, například zpětné lomítko, procento či složené závorky (tam, kde je chceme vytisknout, musíme uvést `\backslash`, `\%`, `\{` nebo `\}`),
2. překladač respektuje pouze jednu mezeru, nepomůže tedy posouvání textu tabulátorem či vkládáním mezer, nelze ani posouvat směrem dolů,
3. ukončení řádku stiskem ENTER nezačne nový odstavec, má pouze „optický“ vliv na čitelnost zdrojového textu,
4. na nový řádek se přejde příkazem `\newline`, `\linebreak` nebo vložením příkazu `\\[`*vel*`]`, jehož volitelný parametr udává, kolik místa se má vynechat (může se zadat i záporná hodnota pro přiblížení) a to v jednotkách `mm`, `cm`, `in` (v palcích), `ex` (šířka písmene `x` v aktuálním písmu) nebo `pt` (v tiskařských bodech),
5. nový odstavec lze udělat vložením:
   (a) nejméně jednoho prázdného řádku,
   (b) příkazu `\par`,
6. uvozovky anglického tvaru se začínají jako dva obrácené apostrofy (`‘`) a končí jako dva apostrofy (`’`), uvozovky českého tvaru (tzn. dole začínají a nahoře končí) se vkládají pomocí příkazu `\uv{`*text*`}`, který je definován v balíku `czech.sty`,
7. pro označení výpustky (elipsy) nepoužíváme tři tečky, ale příkaz `\ldots` nebo `\dots`, protože zde musí každá tečka zaujímat šířku jednoho znaku.

Řezy písma Systém TEX používá své vlastní písma (fonty), které jsou odvozeny z originálních rytin z období klasicismu. Toto písmo bylo pojmenováno Computer Modern (CM). Písmo je dostupné v různých velikostech a v různých řezech, což jsou jeho tvarové modifikace, které slouží k odlišení částí textu.

V typografii se k běžné sazbě používá normální stojaté netučné písmo – označované jako antikva (roman), které používáme právě teď. Pro zvýraznění pasáží textů, pro citáty apod. se používá *italika*, neboli *kurzíva* (jak vypadá, to jste četli před chvílí). Italika bývá skloněná, užší než stojaté písmo a zpravidla bývá zdobnější, více připomíná rukopis. Některá písmena mají zcela jinou kresbu, většina však je jen více prokreslená. V tabulce jsou pro srovnání první písmena psána antikvou, druhá dvě italikou a poslední matematickou italikou[9]

$$\text{aa} \quad aa \quad aa \;\big|\; \text{ff} \quad ff \quad ff \;\big|\; \text{gg} \quad gg \quad gg$$

Používáme-li italiku, musíme pamatovat, že toto písmo má „šikmou osu“ a po jeho skončení přepnutím do neskloněného písma by mohlo dojít k „prolnutí“ písmen. Abychom tomu zabránili, musíme použít tzv. kurzívní korekci příkazem `\/`, která následující písmena mírně posune (např. *f*F $\rightarrow$*f*F).

Pro zdůraznění důležitých krátkých textů, označování nadpisů apod. se používá **tučný řez**, který má tučnější tahy (především svislé) a je také o něco širší. Pro vyznačování především jmen a názvů se používá KAPITÁLEK, což je písmo, u něhož jsou malá písmena stejného tvaru jako velká, jen jsou v měřítku zmenšena[10]. Dalším užívaným řezem v běžném textu je Sans Serif (grotesk), což je bezpatkový

[9] U mnoha písem se výrazně liší také malé písmeno „k“. Ve fontech Computer Modern se liší jen nepatrně. Velká písmena bývají také více prokreslená nebo – jako ve fontech Computer Modern – jsou pouze skloněnou variantou stojatého řezu.

[10] Používání KAPITÁLEK nebo přímo VELKÝCH PÍSMEN dodává písmu slavnostní ráz, ale na úkor čitelnosti. Proto by se mělo s tímto řezem šetřit.

řez (chybí mu pozvolná zakončení písmen, je ostře zakončeno). Grotesk se používá především u krátkých pasáží, protože písmo bez patek se hůře čte. Ale v moderní sazbě se užívá i v knižní beletrii. Posledním řezem je *skloněné* písmo, které se liší od kurzívy tím, že zůstane přesně zachován tvar stojatých písmen (např. *a*), jen mají písmena šikmou osu. U mnoha písem se skloněné písmo používá místo italiky (např. písma Helvetica, Arial).

V matematických textech je navíc pro označení proměnných používána tzv. *matematická italika*, která se od běžné italiky nepatrně liší (v mezerování a šířce písmen). Tento řez není v textu běžně dostupný. Pro označování příkazů, systémových hlášení, napodobení písma psacího stroje apod. je zaveden řez `strojového písma`, které má pevnou šířku znaku.

Další metody zvýrazňování, které se používaly například u psacích strojů, nejsou typograficky moc vhodné a jsou přijatelné jen v malém měřítku. Jedná se o prostrkávání (za každým písmenem je jedna mezera), psaní velkých písmen a podtrhávání (není vhodné, protože se jedná o formu zvýraznění, která je nejvíc dostupná čtenáři – ten jen těžko něco přepíše v tučném řezu). Budete-li však na nich trvat, je možno jich použít: prostrkávání a velká písmena musíte provést sami, pro podtržení lze použít příkaz `\underline{`*text*`}`.

Ve starší verzi LaTeXu byly k dispozici pouze základní řezy písma, které se vybíraly pomocí příkazů `\rm` (antikva), `\it` (italika), `\bf` (tučná antikva), `\sf` (grotesk), `\sl` (skloněná antikva), `\sc` (kapitálky) a `\tt` (strojopis). Tyto příkazy (které jsou kompatibilní s plainTeXem) se používaly tak, že se uvedly před zvýrazňovaným textem a po něm se vrátil původní řez, např. `zapnu \bf tučné \rm písmo`. Častěji se však využívalo vlastností bloků. **Blok** je část textu uzavřená buďto do svorek `\begin{...}` a `\end{...}` nebo jen do složených závorek `{...}`. Takto ohraničená část se chová jako uzavřená směrem ven, tedy veškeré změny provedené **před** blokem budou platit i v bloku, ale veškeré změny uvedené uprostřed bloku mimo blok již neplatí. Totéž platí i pro řez písma a proto se předchozí příklad dá psát jako `zapnu {\bf tučné} písmo`, což dá text „zapnu **tučné** písmo“.

Poznámka: Jako zvláštní blok vystupují prázdné závorky `{}`, které nic neuzavírají, ale mohou například ukončovat příkazy nebo pomáhat při třídění. Například příkaz pro sázení loga je `\TeX`, ke kterému je možno přidat jakýkoliv další text, tedy například příkazem `\TeX u` mohu psát o TeXu. Když však potřebuji vynechat mezeru mezi logem a dalšími znaky, musím příkaz ukončit: `\TeX{}`. U třídění jde například o problém se spřežkami (písmeno ch) – pokud mám třídicí program, který třídí česky, zařadí mi všechny kombinace písmen ch za písmeno h. Toto však není správné, pokud se jedná o cizí nebo složené slovo, kde jde o dvě nezávislá písmena a mají být řazena pod c. Abych tomu vyhověl, mohu zadat například `c{}h`, což se v textu neprojeví.

Rovněž lze prázdnou skupinu použít i u **slitků**, což jsou kombinace písmen, která se „slila“ do jednoho (jedná se například o ff, fl, fi, ffi). Takové slitky jsou povoleny uvnitř slov, ale nejsou správné, jde-li o slovo složené (např. jedno slovo končí na `f` a druhé na `f` začíná). Zde si lze pomoci právě prázdným blokem – `-f{}f-` dá -ff-.

Uvedené příkazy přepínají vždy na konkrétní řez písma a nelze s nimi dosáhnout kombinací (např. skloněný strojopis). Problémy také nastávají v případě, když chcete používat jiné písmo než Computer Modern. Z důvodu systematizace se proto zavedlo jiné schéma volby písma a řezu, označované zkratkou NFSS. V něm se místo příkazů zavádějících konkrétní řez používají ortogonální přepínače atributů písma a LaTeX sám pak vybere příslušné písmo. Nové příkazy, které začínají na `\text`, přepnou při sazbě svého argumentu právě jeden atribut. Existují příkazy, které nastavují rodinu písma (`\textrm` nastaví antikvu, `\textsf` nastaví grotesk a `\texttt` strojové písmo), příkazy, které nastavují váhu (`\textbf` tučné, `\textmd` normální) a příkazy, které nastavují tvar písma (`\textit` italiku, `\textsl` skloněné, `\textsc` kapitálky a `\textup` vzpřímené). Tyto příkazy pak lze libovolně[11] kombinovat, například `\textit{\texttt{italiku psacího stroje}}` použije ***italiku psacího stroje***. Dále existují ještě příkazy `\textnormal`, který nastaví normální písmo dokumentu (tj. zruší všechny další atributy) a `\emph`, který argument zvýrazní (nastaví nebo zruší naklonění písma, podle toho, jaký je text v jeho okolí). Lze například psát `\textit{okolní \textbf{text \textnormal{je} tučný} a skloněný}`, což dá „*okolní **text** je **tučný** a skloněný*“.

V některých případech není vhodné aplikovat změnu atributu jen na argument příkazu, proto jsou nadefinovány příkazy `\rmfamily`, `\sffamily`, `\ttfamily`, které nastavují rodinu, `\itshape`, `\slshape`, `\scshape`, `\upshape` nastavující tvar a `\bfseries`, `\mdseries` nastavující váhu. Ty mají stejnou působnost jako příkazy ze starší verze LaTeXu. Dále jsou ještě k dispozici příkazy `\normalfont` a `\em`, které opět nastaví normální font nebo zvýrazňující řez.

[11] Písma pro všechny kombinace však nemusí být v systému dostupná, LaTeX pak použije náhradní a vypíše varování.

Velikost písma Druhým parametrem písma je jeho velikost[12]. Běžně se používá pouze několika násobků základní velikosti. Základní velikost písma je 10 bodů. Tuto velikost je možno změnit zavedením stylů `11pt` nebo `12pt` v příkazu `\documentclass`. Velikost 12 bodů přibližně odpovídá velikosti písma psacího stroje. Pro označování velikostí písma se používají následující příkazy[13]:

Příkaz	Ukázka
`\tiny`	písmo
`\scriptsize`	písmo
`\footnotesize`	písmo
`\small`	písmo
`\normalsize`	písmo
`\large`	písmo
`\Large`	písmo
`\LARGE`	písmo
`\huge`	písmo
`\Huge`	písmo

Při přepínání velikosti písma starým způsobem je nutno mít na paměti, že se automaticky nastaví i normální řez, je tedy nutno psát vždy např. `\Large\bf`, nikdy `\bf\Large`. Při volbě vhodné velikosti písma je vhodné dodržet zásadu, že se nemá písmo najednou zvětšovat o více stupňů a nemá se střídat moc velikostí. Zpravidla se využívá jen jedna velikost pro běžný text a o stupeň či dva menší pro samostatné poznámkové odstavce vložené do textu.

Standardní nastavení velikostí je: `\normalsize` pro normální text, `\scriptsize` pro první úroveň indexů, `\footnotesize` pro poznámky pod čarou.

Obecná definice písma – NFSS Nový systém výběru písma umožňuje použití libovolného písma a nezávislou volbu jeho atributů. Jednotlivé atributy jsou

- kódování fontu, nejčastěji `OT1`, které udává způsob uspořádání znaků v sadě (může být třeba `IL2` pro fonty s ISO Latin2 kódováním);
- rodina písma, v podstatě libovolný název, např. `cmr` (Computer Modern Roman), `cmss` (Computer Modern Sans Serif);
- váha, udávající sílu tahů (duktus), nabývá hodnot `m` Medium (běžné), `b` Bold (tučné), `bx` Bold Extended (tučné širší), `sb` Semibold (polotučné), `c` Condensed (zúžené);
- tvar, popisující modifikaci základního typu, `n` Normal (vzpřímené), `it` Italic (kurzíva), `sl` Slanted (skloněné), `sc` Small caps (kapitálky);
- velikost písma.

Pokud chceme ovlivnit pouze samostatné atributy již zavedeného písma, lze použít příslušný příkaz `\fontencoding{`*kódování*`}`, `\fontfamily{`*rodina*`}`, `\fontseries{`*váha*`}`, `\fontshape{`*tvar*`}` a `\fontsize{`*velikost*`}{`*řádkování*`}`. Pro „zprovoznění“ je nutno po příkazech uvést ještě `\selectfont`. Je-li potřeba použít nového písma, musí být nejprve zavedena jeho rodina příkazem `\DeclareFontFamily{`*kódování*`}{`*jméno rodiny*`}{}` (třetí parametr je obvykle prázdný). Každá rodina může sestávat z mnoha členů, kteří se definují příkazem `\DeclareFontShape{`*kódování*`}{`*jméno rodiny*`}{`*váha*`}{`*tvar*`}{`*seznam velikostí a jméno souboru*`}{}`. Po deklaraci členů je již můžeme používat, jako ukázka budiž uvedeno použití písma Dunhill posloupností příkazů `\DeclareFontFamily{IL2}{dunhill}{}`, `\DeclareFontShape{IL2}{dunhill}{m}{n}{<8><10>csdunh10}{}` a `\fontencoding{IL2}\fontfamily{dunhill}\fontseries{m}\fontshape{n}\fontsize{8pt}{10pt}\selectfont`. Ukázka je zde. Obdobně lze užívat i jiná písma dostupná v distribuci, např. písmo Pandora či Fibonacci.

Poslední možností, jak použít jiný font – zpravidla PostScriptový, je použití balíčku, který změní standardní písmo Computer Modern na jiné v celém dokumentu, tedy `\usepackage{`*jméno*`}`, kde *jméno* může být `palatino`, `bookman` apod.

[12] Udávaná velikost písma však nemusí souviset se skutečnou výškou znaků, tj. dva fonty o velikosti 10 pt mohou mít třeba jinou výšku.

[13] Tyto příkazy nenastavují absolutní hodnotu velikosti, ale vždy relativní vzhledem k velikosti základní – 10, 11 a 12 bodů.

2.3 Tok textu

Implicitně je text zarovnáván na obě strany (do bloku) s dělením slov a není možno vkládat mezery navíc či posunovat odstavce apod. Chceme-li však změnit toto uspořádání, můžeme použít některý z následujících postupů:

1. Prostředí `center` umožňuje sazbu textu na střed řádků, středování se provádí v části textu mezi příkazy `\begin{center}` a `\end{center}`.
2. Příkaz `\centerline{`*text*`}` vysází centrovaný jednořádkový *text*, obdobně příkaz `\rightline{`*text*`}` vysází jeden řádek zarovnaný napravo a `\leftline{`*text*`}` nalevo.
3. Příkaz `\␣` vysází právě jednu mezislovní mezeru, další příkazy pro sazbu vodorovných mezer jsou `\,` (menší než mezislovní), `\quad` a `\qquad`, pro vysázení mezery o šířce jedné číslice použijeme `\enspace`.
4. Příkaz `\hspace{`*vel*`}` vysází horizontální mezeru o velikosti *vel*, pokud se příkaz nevyskytuje na začátku řádku, příkaz `\hspace*{`*vel*`}` ji vysází vždy.
5. Příkaz `\vspace{`*vel*`}` vysází vertikální mezeru o velikosti *vel*, pokud se příkaz nevyskytuje na začátku nebo konci stránky, příkaz `\vspace*{`*vel*`}` ji vysází vždy. Další příkazy `\smallskip`, `\medskip` a `\bigskip` vytisknou svislé mezery o velikosti čtvrtiny, poloviny nebo celé výšky řádku.
6. Příkaz `\hfill` vysází tak dlouhou horizontální mezeru, jak jen je to možné – tzv. pružnou mezeru, obdobně příkaz `\vfill` sází vertikální mezeru. Těmito příkazy je možno vysázet text například vpravo dole.
7. Příkazy `\hrulefill` a `\dotfill` pracují podobně jako `\hfill`, ale danou mezeru zaplní buď čárou nebo tečkami. Podobně příkazy `\leftarrowfill` a `\rightarrowfill` ji vyplní vodorovnou šipkou dané orientace, `\upbracefill` a `\downbracefill` ji vyplní vodorovnými svorkami.
8. Příkaz `\indent` způsobí odsazení začátku řádku, i když se nejedná o začátek odstavce (není-li uveden na začátku řádku, vloží jen stejně velkou mezeru); naopak příkaz `\noindent` zamezí odsazení nového odstavce.
9. Příkaz `\newpage` či `\clearpage`[14] způsobí přechod na novou stránku, příkaz `\pagebreak` se bude navíc snažit zaplnit stránku roztažením pružných mezer.

2.4 Členění textů, obsahy

Každý text bývá členěn do kapitol, sekcí, paragrafů apod. K jejich zavádění má LaTeX samozřejmě nadefinované své příkazy. Protože se jedná o systém pro sazbu technických publikací, jsou implicitně všechny názvy oddílů číslované, ale není problém číslování vypnout (přidáním hvězdičky `*` těsně za příkaz). V sytému LaTeX se zadává pouze údaj o tom, jaká úroveň má být vytištěna a co má být vytištěno, ne jak. Všechny číslované oddíly jsou automaticky zanášeny do obsahu, nečíslované je nutno „ručně“ vložit.

Nejvyšším oddílem textu pro třídy `book` a `report` jsou kapitoly. Každá kapitola je při tisku označena (při stylu `czech`) slovem „Kapitola“ a pořadovým číslem kapitoly, pod tím je vytištěn vlastní název, nová kapitola je vysázena na novou stránku (případně na lichou ve třídě `book`). Příkazem pro vytvoření číslované kapitoly je `\chapter[`*textob*`]{`*text*`}`, kde *text* je vlastní nadpis kapitoly, volitelný *textob* je text, který má být zanesen do obsahu. Příkaz `\chapter*{`*text*`}` způsobí vysázení nečíslované kapitoly, její obsah nebude v seznamu.

Vytvoření nového oddílu automaticky způsobí vynulování čísel všech ostatních nižších sekcí, tj. budou dále nabíhat s číslováním opět od jedné. Číslování jednotlivých sekcí je **hierarchické**, tedy může vypadat např. takto: 2.1.4 Je třeba si všimnout, že za posledním číslem není tečka (odpovídá anglické sazbě) a hned tak tam taky nebude.

Oddíly nižší úrovně se tvoří obdobně, jenom se mění klíčová slova a způsob sazby. O tom, do jaké hloubky se budou úrovně číslovat a zanášet do obsahu, rozhoduje proměnná `secnumdepth`. Klíčová slova pro oddíly nižší úrovně postupně jsou `\section`, `\subsection`, `\subsubsection`, `\paragraph` a `\subparagraph`. Oddíl vyšší úrovně než kapitola existuje jen jeden – část knihy, uváděná příkazem `\part`.

[14] Tento příkaz má ještě další vlastnosti

Po vysázení oddílu textu se první odstavec neodsazuje (anglická sazba), a to ani když zadáte příkaz `\indent`. Trváte-li na odsazení (česká sazba), použijte kombinaci `\hspace\parindent` nebo balíček `indentfirst`.

Vysázení automaticky vytvářeného obsahu dosáhnete pomocí uvedení příkazu `\tableofcontents` v místě, v němž chcete obsah umístit. Obsah se automaticky neumísťuje na novou stránku a ani ji neukončuje.

Chcete-li do obsahu přidat nečíslovaný oddíl, použijete **po** jeho uvedení příkaz `\addcontentsline{`*soub*`}{`*sekce*`}{`*text*`}`, kde *soub* je přípona označující do kterého souboru se má *text* zapsat (pro obsah je přípona `toc`), položka `sekce` je označení do jaké úrovně má být položka zanesena (je tvořena klíčovým slovem bez zpětného lomítka, např. `subsection`). K této položce je v obsahu automaticky vygenerováno číslo stránky.

Chcete-li přidat do obsahu obyčejný text (např. autora kapitoly), použijte příkaz `\addtocontents{`*soub*`}{`*text*`}`, kde *soub* je opět přípona souboru a *text* je zanášený text; zde se číslo stránky do obsahu nezanáší.

Poznámky pod čarou K vysázení poznámky pod čarou se v místě, kde chceme uvést odkaz na poznámku (číslo jako horní index), použije příkazu `\footnote{`*text*`}`, poznámka *text* se vysází na spodní část stránky pod třetinovou čáru (tu lze rovněž vytvořit příkazem `\footnoterule`). Poznámky se číslují automaticky, pokud chceme, aby poznámka měla přesně určené číslo, použijeme volitelný parametr s daným číslem. V některých prostředích nelze poznámku uvést (např. `tabular`), zde použijeme `\footnotemark` pro vytvoření odkazu a po skončení prostředí příkaz `\footnotetext` vysází text s číslem odpovídajícím naposled použitému příkazu `\footnotemark` (může mít volitelný parametr čísla poznámky).

Poznámky na okraji Marginální poznámky mohou být vkládány pomocí příkazu `\marginpar{`*text*`}`, v němž *text* je právě to, co bude vysázeno. Tyto poznámky jsou sázeny vždy na vnější okraj, u jednostranného tisku vpravo (pokud chcete oboustranný tisk, použijte volbu `twoside` v `\documentclass`). V oboustranném režimu má příkaz volitelný parametr, který může obsahovat odlišný text pro levý okraj, podle toho, zda se poznámka umístí na sudé nebo liché stránce.

2.5 Různá prostředí

Pro různé typy sazby má LaTeX zabudována různá prostředí, která umožňují jednoduché vytváření efektů. Největším prostředím je `document`, kterým musí každý text začínat. Definice prostředí je vždy stejná – `\begin{...}`, rovněž i ukončení `\end{...}`, kde uvedené výrazy v závorkách musí být naprosto shodné. Prostředí se mohou do sebe vnořovat, ale nesmí se překrývat. Za příkazem `\begin` mohou následovat volitelné či povinné parametry.

Krátký soupis **výčtových** prostředí:

1. `enumerate` je prostředí, které umožňuje zadávání číslovaných položek. Každá položka je zadána příkazem `\item`, v sazbě je uvedena na novém řádku, označena svým číslem a oddělovačem (tečkou, závorkou) a celý odstavec je mírně odsazen z levé strany. Je-li za `\item` uveden v hranatých závorkách volitelný parametr, bude místo čísla vysázen tento parametr. Prostředí se mohou do sebe vnořovat do 4 úrovní, každá úroveň použije své označení (čísla, malá písmena, velká písmena, římské číslování). Příkaz `\itemsep` *vel* nastavuje velikost svislé mezery mezi jednotlivými položkami na hodnotu *vel* (např. `0pt`).
2. `itemize` je prostředí podobné předchozímu, jen pro označení položky používá nějakého neliterálního symbolu (např. puntík), lze rovněž umístit volitelné označení.
3. `description` je prostředí pro sazbu popisných výčtů (např. encyklopedické výklady), položka se zadává jako `\item[`*návěští*`]`, ve výčtu bude *návěští* vytištěno tučně a zarovnáno napravo.
4. `trivlist` je nejjednodušší prostředí, které pracuje jako `itemize`, jen před položku nevkládá žádný doplňkový symbol, musí být zadán jako volitelný parametr.
5. `list` je nejobecnější výčtové prostředí, z něhož mohou být ostatní odvozena. Má dva parametry, kde první parametr udává způsob formování „znaku“ před položkou (zpravidla obsahuje příkaz LaTeXu a nějaký čítač) a druhý parametr určuje formátování. Ve druhém parametru se také nastavují parametry odsazení:

(a) `\rightmargin` — nastavuje odsazení od pravého okraje,

(b) `\leftmargin` — totéž vlevo,

(c) `\parsep` — vzdálenost mezi odstavci jedné položky,

(d) `\itemsep` — vzdálenost mezi položkami (přičítá se ještě `\parsep`),

(e) `\label` — vodorovná vzdálenost od návěští k textu položky,

(f) `\listparindent` — odsazení odstavce,

(g) `\labelwidth` — šířka návěští.

6. `thebibliography` je prostředí pro sazbu literatury a obsahuje povinný parametr, jehož počet znaků je roven maximálnímu počtu znaků v návěští. Jednotlivé položky se zadávají příkazem `\bibitem[`*tisk*`]{`*text*`}`, jehož parametr *text* udává název, pomocí něhož se na literaturu bude odkazovat příkazem `\cite[`*vol*`]{`*text*`}` a *tisk* je volitelný parametr, který bude případně vytištěn v seznamu místo čísla. Volba *vol* se přidá do citace za čárku (například odkaz na stránku). Vlastní odkaz je uzavřen do hranatých závorek a může obsahovat i více odkazů, oddělených čárkami. Uvedený příkaz ale neumožňuje výpis odkazů formou intervalu (např. [4–7]). Potřebujete-li takový způsob, použijte balík `cite`, který původní příkaz rozšiřuje o uvedenou funkci a navíc umí odkazy setřídit abecedně a podle velikosti. Nad prostředím se vypíše nadpis „Literatura“ nebo „Reference“ dle stylu dokumentu. Chcete-li změnit tento nadpis, použijte `\renewcommand{\refname}{`*text*`}`.

7. `theindex` je prostředí pro vytváření rejstříků, jednotlivé položky se zadávají pomocí položek `\item`, `\subitem` a `\subsubitem` (podle úrovně vnoření) a za nimi následují příslušná čísla stránek. Vlastní index se vytváří pomocí příkazů `\index{`*text*`}`, které způsobí (je-li v preambuli příkaz `\makeindex`) výpis položek do souboru s příponou `idx`. Tento soubor se prožene speciálním programem (CsIndex), který výstup abecedně přetřídí a vytvoří nový, který již bude obsahovat položky `\item`, `\subitem`, `\subsubitem`.

Krátký seznam **sloupcových** prostředí:

1. `tabbing` slouží k vytváření zarovnaných textů bez produkce čar obdobně jako na psacím stroji pomocí tabelačních zarážek, prostředí nemá žádný parametr. V průběhu textu se uvádějí symboly `\=` pro nastavení zarážky, příkaz `\>` přesune text na pozici další zarážky, příkazy `\+` a `\-` posouvají místo první zarážky na novém řádku (po jednom uvedení `\+` nebude text sázen hned zleva, ale až od první další zarážky, druhý příkaz ruší vliv posunutí), příkaz `\kill` je možno uvést na nastavovacím řádku (vypíšeme jeden řádek, v němž označíme pozice zarážek, ale tento řádek nechceme tisknout – místo ukončení `\\` použijeme `\kill`), příkazy `\pushtabs` a `\poptabs` uloží a znovu načtou pozice všech zarážek na řádku, tyto páry příkazů je možno vnořovat, ale nesmí se překrývat. Další příkazy `\'` a `\‘` posouvají text vpravo, první jen k předchozí zarážce (od níž bude vzdálen o hodnotu `\tabbingsep`), druhý až zcela doprava (po něm nesmí následovat žádný příkaz `\>`, `\=` nebo `\'`).

   V prostředí `tabbing` navíc neplatí některé příkazy pro sazbu akcentů apod. (většinou je nutno vsunout `a`, tedy místo příkazu pro sazbu čárky `\'` se uvede `\a'`). Prostředí se nejčastěji používá pro sazbu zdrojových textů programů napsaných v některém strukturovaném jazyce.

2. `tabular` je prostředí pro sazbu sloupcových tabulek včetně čar. Ve volitelném parametru je možno zadat umístění tabulky vzhledem k okolnímu textu (`t` – připojení horním okrajem, `b` – připojení dolním okrajem), v povinném parametru se udává počet sloupců a jejich oddělení tak, že vždy je uveden symbol charakterizující zarovnání a kolem něj jsou oddělovací symboly. Symboly pro zarovnání jsou

   - `c` — způsobí sazbu textu uprostřed sloupce,
   - `r` — sazba textu zarovnaného vpravo,
   - `l` — sazba textu zarovnaného vlevo, u všech tří symbolů se šířka sloupců nastavuje automaticky dle nejširšího obsahu sloupce a každá položka v tabulce může obsahovat pouze jeden řádek (text není v sloupci zalamován),
   - `p{`*vel*`}`— vytvoří sloupec o šířce *vel*, v němž bude text zarovnán dle pravidel zarovnávání běžného textu a bude zalamován,
   - `*{`*n*`}{`*def*`}`— používá se ke zkrácení definice, má-li několik sloupců stejný tvar, parametr *n* udává, kolikrát se má zopakovat definice sloupce *def*, která obsahuje některý z výše uvedených symbolů.

Před a za každým symbolem v parametru může být uveden symbol jeho oddělení:

- `|` způsobí vytištění jednoduché čáry podél sloupce (umístění čáry záleží na tom, na které straně zarovnávacího symbolu je uveden), uvedený symbol je na klávesnici ve Windows dosažitelný stiskem Ctrl+Alt+W, na některých klávesnicích může být zobrazen jako přerušovaná čára,
- `||` způsobí vysázení dvojité čáry podél sloupce,
- `@{`*def*`}` způsobí vysázení symbolu *def* místo čar, např. zajímavým efektem pro sazbu seznamu symbolů je `l@{\dotfill{ }}l`,
- bez zvláštního znaku — sloupec nebude oddělen od předchozího, jen se vloží mezera udávaná délkovým příkazem `\tabcolsep`.

Jako příklad definice lze uvést např. posloupnost `|c|ll||r`, která způsobí vysázení jednoho sloupce středovaného, dvou zarovnaných vlevo a jednoho vpravo, první sloupec bude z obou stran oddělen čárou, poslední od předposledního dokonce dvojitou. Prostředí má ještě jednu variantu s hvězdičkou, při níž se před definici sloupců ve složených závorkách vloží celková šířka tabulky. Pokud chcete vysázet jeden řádek tabulky zcela bez zarovnání, lze použít `\noalign{`*text*`}\\`.

Sázení jednotlivých sloupců vypadá tak, že vypisujete jejich obsahy a jako oddělovače sloupců používáte znak `&` (dostupný na české Windows klávesnici přes Ctrl+Alt+C). Tento znak nesmí být uveden za posledním sloupcem, zde musí být uveden symbol konce řádku `\\` (pokud znak `&` přebývá, vypíše TEX hlášku `Extra alignment tab has been changed to \cr.`). Je-li v tabulce na posledním místě sloupec se zarovnáním `p`, může obsahovat symbol `\\` pro řádkový zlom uvnitř sloupce a konec řádky tabulky se zadá příkazem `\tabularnewline[`*vel*`]` s nepovinným parametrem odsazení dalšího řádku o *vel*. Chceme-li text umístit přes více sloupců, musíme použít příkaz `\multicolumn{`*počet*`}{`*zar*`}{`*text*`}`, který vysází *text* přes *počet* sloupců se zarovnáním *zar*. Pro takto vepsaný text *neplatí* oddělování sloupců uvedené v hlavičce, pokud chcete např. vystředit text a ohraničit řádek na okrajích, musíte jako zarovnání uvést `|c|`. Nastavení prostředí (např. řezu písma), provedené v jednom sloupci, neplatí v dalších sloupcích ani řádcích.

Pro vysázení čar je možno použít tyto příkazy

- `\hline` vysází jednu vodorovnou čáru přes celou tabulku, příkaz může být uveden hned na začátku tabulky nebo po znaku konce řádky. Jsou-li uvedeny dva příkazy za sebou, vysází se dvojitá čára (tabulky se však od sebe opticky „oddělí“),
- `\cline{`*n*`-`*m*`}` vysází vodorovnou čáru jen mezi *n*-tým a *m*-tým sloupcem[15],
- `\vline` vysází svislou čáru v místě uvedení přes jeden řádek tabulky.

Používáte-li zarovnání sloupce `p` a máte text, který tvoří více řádků, může se stát, že budete chtít posunout svisle text v dalších jednořádkových sloupcích tak, aby byl uprostřed výšky řádku (automaticky se texty sází vždy od vrchu). V tom případě můžete užít v každém sloupci příkaz `\raisebox{`*posun*`}{`*text*`}`, který vysází *text* s vertikálním posunutím *posun*, v našem případě musíte udat hodnotu posunutí zápornou.

Prostředí `tabular` narozdíl od předchozího nesmí přesáhnout rozsah jedné stránky[16].

Některá **textová** prostředí:

1. `quote` se používá pro vysázení krátkého citátu, je odsazeno z obou stran a umístěno doprostřed, první řádky odstavců nejsou odsazeny, mezi odstavci je svislá mezera.
2. `quotation` je podobné jako předchozí, jen chybí svislá mezera a naopak je odsazení prvních řádek odstavců.
3. `flushleft` je prostředí pro zarovnání na levý prapor, obdobně pracují prostředí `flushright` a `center`.
4. `verse` je prostředí pro sazbu veršů, každá strofa začíná po prázdném řádku, není odsazení, jednotlivé verše se ukončují `\\`.

[15] Uvedený příkaz přestane fungovat, použijete-li volbu `split` u balíčku `czech`.
[16] Potřebujete-li tabulky delší, můžete použít balíček `longtable`. Rozšířené možnosti sazby tabulek nabízí soubor `tap`.

5. `verbatim` je prostředí pro doslovnou sazbu textu, především programů apod., text nepodléhá formátování. Standardně se používá písmo psacího stroje, veškeré příkazy jsou ignorovány vyjma `\end{verbatim}`. Při přejímání programů přímo z editorů daného jazyka je nutno pamatovat na to, že je ignorován tabulátor, teda všechna posunutí (zarovnání) musí být dosažena pomocí *mezer*.

   Pro krátký úsek textu je možno použít rovněž příkazu `\verb`, který na prvním místě obsahuje nějaký zvláštní znak (který se v dalším textu nevyskytuje, např. `#`), za ním je text, který má být vytištěn bez formátování a tento text je ukončen oním zvláštním znakem. Tedy např. `\verb#\multicolumn#` dává výsledek `\multicolumn`. Jako zvláštní znak nesmí být použita hvězdička, protože ta je vyhražena pro prostředí, v němž se mezery nahrazují znakem ␣ (tento znak je dosažitelný i pomocí příkazu `\textvisiblespace`). Prostředí pro doslovnou sazbu textu se nesmí vyskytnout jako parametr jiného příkazu (např. v poznámce pod čarou), respektive se v něm nesmí vyskytovat některé znaky (např. zpětné lomítko; pokud ho tam potřebujete, použijte příkaz `\char92` ve strojopisném řezu).

6. `minipage` je prostředí, které se chová jako malá stránka, má volitelný parametr určující jeho polohu k okolnímu textu (`t` znamená připojení vrchní částí, `b` spodní) a povinný parametr určující šířku prostředí. V prostředí dochází k normálnímu zalamování textů, může docházet k řádkovému zlomu apod. Dalšími volitelnými parametry jsou celková výška prostředí a zarovnání textu v prostředí (nabývá hodnot `tbcs`, kde `c` je vystředění a `s` je rozptýlení po celém prostoru). Podobného efektu lze dosáhnout příkazem `\parbox`, který má stejné parametry, jen jako poslední parametr je ve složených závorkách vlastní obsah boxu. V tomto boxu však nelze umístit např. poznámku (u prostředí `minipage` se poznámka umístí ihned pod prostředí).

   Na začátku prostředí se neprovádí odsazení odstavce (nelze ani vložením `\hspace\parindent`), jednoduše proto, že proměnná `\parindent` má nulovou délku. Chcete-li stejné odsazení jako v předchozím textu, musíte v něm zavést novou délkovou proměnnou a do ni uložit hodnotu `\parindent` a po začátku prostředí uvést příkaz `\parindent=\`*mojeproměnná* .

Pro umístění objektů, které nemají přesné umístění v textu (jako jsou tabulky a obrázky) se používají takzvaná **plovoucí prostředí**. Tato prostředí se mohou v textu volně pohybovat tak dlouho, dokud se nenajde vhodné místo pro jejich vysázení. Toto umístění můžete částečně ovlivňovat svým přáním, ale moc se to nedoporučuje, protože při přísných požadavcích na umístění se začnou plovoucí objekty hromadit a může pochybět pamět. Z téhož důvodu je vhodné umístit někde občas příkaz pro stránkový zlom `\clearpage`, který má stejnou funkci jako `\newpage`, ale přinutí TEX vysázet veškeré plovoucí objekty. Existují tato dvě plovoucí prostředí:

1. `figure`, které slouží ke vkládání obrázků. Jakýkoliv obsah vnořený do tohoto prostředí bude považován za plovoucí. Nejčastěji je zde vloženo prostředí `picture` nebo příkaz `\special` pro zobrazovač. Pokud chceme vysázet i popisek k obrázku, použijeme příkaz `\caption[`*obsah*`]{`*text*`}`, který vloží popisek *text* přímo nad nebo pod obrázek (záleží na vzájemném umístění příkazu a obrázku) a volitelný parametr *obsah* znamená text, který se přesune do seznamu obrázků (jinak se tam přesune přímo *text*). Vysázení seznamu obrázků se dosáhne příkazem `\listoffigures`, tento seznam je uložen v souboru s příponou `lof`.
   Upozornění: do popisku nevkládáme označení „Obrázek“, toto bude vytvořeno automaticky i s číslem obrázku.

2. `table` je prostředí pro plovoucí tabulky, pracuje podobně jako předchozí, seznam tabulek je dosažitelný příkazem `\listoftables`, přípona souboru je `lot`.

Pro umístění plovoucích objektů je možno zadat tyto parametry (i jejich smysluplné kombinace, v nichž záleží na pořadí):

- `h` je přednostní umístění právě zde,
- `t` je přednostní umístění v horní části stránky,
- `b` je umístění v dolní části stránky,
- `p` je umístění na stránce, na níž jsou **pouze** plovoucí objekty,
- `!` je specifikátor, který vyžaduje nejméně jeden z předchozích symbolů a umožňuje potlačit omezení maximálního počtu plovoucích objektů na stránce a minimálního obsahu textu na stránce.

Chceme-li pevně svázat dva obrázky či dvě tabulky, je nejvhodnější vložit je do jednoho prostředí a uvést dva páry příkazů `\caption` a `\label` pro případné odkazy.

2.6 Sazba akcentů a speciálních symbolů

Ze speciálních tisknutelných znaků můžeme uvést tyto:

- Tilda (~) představuje nezlomitelnou mezeru, vkládá se zpravidla (v češtině) mezi předložku a další slovo, protože čeština přenáší přízvuk na předložky a je tudíž nevhodné, aby se zde objevil konec řádku. Dále se udává mezi číselným údajem data a názvem měsíce, mezi hodnotou veličiny a její jednotkou (někdy se zde místo celé mezery používá jen malá mezera) apod. Chcete-li tildu vytisknout, musíte zadat `\~`.
- Pomlčka existuje ve třech verzích: jako spojovník `-` (například příklonka -li), rozsah `--` ve významu od – do a interpunkční znaménko — (pomlčka) `---`. Rozdílné je také matematické mínus −.
- speciální grafické znaky † `\dag`, ‡ `\ddag`, § `\S`, ¶ `\P`, © `\copyright` a £ `\pounds`. Vysázení znaku ‰ je možno příkazem `\char141`.
- speciální „jazykové“ znaky œ `\oe`, Œ `\OE`, æ `\ae`, Æ `\AE`, å `\aa`, Å `\AA`, ø `\o`, Ø `\O`, ł `\l`, Ł `\L`, ß `\ss`, ¿ `?‘`, ¡ `!‘`.
- příkazem `\textcircled` lze zakroužkovat jeden znak.

V textu jsou dostupné všechny běžné akcenty, které se vkládají pomocí příkazů. Pokud používáte styl `czech`, lze vkládat českou diakritiku přímo (např. ě), jinak musíte pomocí příkazů (tzv. **TEXsekvence**). Toto vkládání také musíte[17] provést, chcete-li text s diakritikou poslat někam do zahraničí.

Původní fonty Computer Modern (to jsou ty, které začínají na `cm`, např. `cmr8`) nepočítaly s češtinou, ale uměly akcenty, tzn. že písmena s diakritikou byla poskládána z písmene bez diakritiky a z akcentu. Tento způsob však dává poněkud horší výsledky. Proto Československé sdružení uživatelů TEXu ($\mathcal{CS}$TUG) navrhlo vlastní fonty, které znaky s diakritikou již přímo obsahují. Tyto fonty vznikly přepracováním a používají podobné označení, jen se symbolem `cs` (např. `csr8`). Přeložíte-li v české instalaci zdrojový soubor, bude se odkazovat na české fonty (otevřete si `dvi` soubor v binárním prohlížeči a uvidíte tam na ně odkazy). Pošlete-li takovýto přeložený soubor do zahraničí, bude jejich zobrazovač vyžadovat také fonty $\mathcal{CS}$ a protože je nenajde, nezobrazí nic nebo použije nesprávný font. Proto se musí před odesláním buď použít program typu DVI Cs–>Cm, který označení souborů (fontů) zamění, nebo zdrojový text přeložit „nepočeštěným“ systémem (použít formát LATEX místo $\mathcal{CS}$LATEXu, případná písmenka s diakritikou se musí nahradit podle tabulky níže).

Značení fontů je dáno schématem: označení `cm` nebo `cs`, označení řezu (např. `b` je tučný, `r` je antikva, ...) a číslo udávající vhodnou velikost písma v bodech (pt). Označení velikosti je pouze informativní, teoreticky je možno použít pro vygenerování všech velikostí jeden soubor, ale zvláště malé velikosti by měly horší kvalitu – proto byly fonty pro různé velikosti rozděleny.

Tabulka akcentů dostupných v běžném textu je uvedena zde, v ukázce se akcenty vkládají nad písmeno `o`. Pokud chcete vkládat akcent nad písmena s tečkou (`i`, `j`), musíte nejprve tečku odstranit příkazem `\i` nebo `\j` (dostanete ı, ȷ).

ò	`\‘o`	õ	`\~o`	ǒ	`\v o`	o̧	`\c o`
ó	`\’o`	ō	`\=o`	ő	`\H o`	ọ	`\d o`
ô	`\^o`	ȯ	`\.o`	o͡o	`\t{oo}`	o̲	`\b o`
ö	`\"o`	ŏ	`\u o`	ů	`\r o`		

[17] Vlastně nemusíte, protože můžete použít vhodný program, např. `cstocs`, který to provede za vás. S jednou nepříjemností – zakódují se i odkazy a budou se hlásit chyby; proto raději v odkazech v tomto případě nepoužívejte češtinu.

3 Sazba matematiky

Silnou stránkou TEXu je výborná sazba všech typů matematických výrazů. Při této sazbě jsme omezeni vpodstatě jen počtem dostupných znaků (proto je pro skutečně složité matematické texty vhodné použít např. $\mathcal{AMS}$-LaTeX, ve kterém je k dispozici velké množství příkazů, rozšiřujících možnosti matematické sazby a také celá řada nových symbolů). Matematická sazba se do textu vkládá v různých matematických prostředích pomocí speciálních příkazů.

Základní charakteristiky matematického prostředí jsou asi tyto:

- Pokud je matematika sázena na zvláštních řádcích, je implicitně centrovaná.
- Veškerý text je sázen matematickou italikou (není-li řečeno jinak) a je chápán jako součin proměnných.
- V matematickém textu platí jiné příkazy pro sazbu akcentů; především nelze přímo vkládat akcenty pomocí českých znaků přímo z klávesnice, např. vložení `slůvka` v matematickém textu dá *slvka*.
- Pro matematický text se nepoužívají běžné způsoby změny velikosti písma a řezu (i když je lze použít), příkazy pro změnu řezu zpravidla předchází slovo `math`, například tučný řez se zadá jako `\mathbf`. Pozor na fakt, že příkaz `\mathit` dosadí pouze textovou italiku, matematická se musí zapnout příkazem `\mathnormal`. Chceme-li vysázet tučný matematický symbol, musíme jej nadefinovat tak, že před použitím matematického prostředí použijeme příkaz `\boldmath` a po skončení matematiky pak `\unboldmath`; porovnejte například $\mu\boldsymbol{\mu}$.
- Matematické výrazy ve vysazeném módu mohou být číslovány, a to ve stylech `article` a `report` průběžně, ve stylu `book` po jednotlivých kapitolách.
- Matematické výrazy, které jsou na samostatném řádku, nejsou automaticky zalamovány, musíte je rozdělit sami (pokud je třeba rozdělit výraz v součinu, zpravidla se pro naznačení tohoto součinu používá znak $\times$, dostupný příkazem `\times`). U matematiky, která je na řádku s běžným textem, k dělení ve vhodných místech dochází, ale toto dělení neodpovídá české sazbě, protože nedochází k opakování „operátoru“ na začátku řádku. Je proto vhodné použít soubor `opakuj.tex`, který zajistí splnění pravidel české sazby.
- Sazba desetinné čárky je poněkud problematická. TEX předpokládá anglickou desetinnou tečku, a čárku proto považuje za oddělovač různých skupin, proto za ní vloží malou mezeru. Ve většině případů se tato praxe považuje za vyhovující, pokud to vadí, je možno vrátit mezeru příkazem `\!` nebo lépe čárku uzavřít do složených závorek, např. `1{,}5`. Jako příklad uvedu výraz s destinnou tečkou, čárkou bez korekce a s korekcí: 3.141592, $3,141592$ a 3{,}141592 (jistě jste poznali, že aproximujeme číslo π).

3.1 Matematická prostředí

Prvním matematickým prostředím je sazba matematiky uprostřed textu (matematický textový mód), třeba $\frac{a}{b}$. Tato sazba je uzpůsobena velikosti okolního textu a je vhodná jen pro méně složité výrazy (několikanásobné složené zlomky nevypadají hezky, $\frac{\frac{x-1}{x^2}}{\frac{x+1}{x^2+1}}$). Začátek prostředí se uvozuje znakem `$`[18] a rovněž se tak i končí (tento znak je na české Windows klávesnici dosažitelný Ctrl+Alt+ů). Případná větná interpunkce by měla být až za prostředím, ale bez mezery (např. `$x$,`).

Druhým prostředím je sazba vysazené (s krátkým „a“) matematiky. Zde je již možno uvést složitější vzorce, jejich velikost je větší než v předchozím případě. K označení začátku prostředí se používá znaků `$$`, rovněž se tak i ukončí. Pokud běžný text pokračuje dále, je vysázen na nový řádek. Pro formátování zdrojového textu je vhodné vložit za znak `$` buď mezeru, nebo konec řádku stiskem Enter (jen jednou, další stisk by vynechal prázdný řádek, což by vysázelo nový odstavec) – vzniklá mezera je TEXem ignorována. Pokud běžný text plynně pokračuje, bývá zvykem na konci matematiky používat běžnou interpunkci (čárku, tečku nebo nic) s ohledem na následující text (interpunkci je nutno zapsat **před** ukončením prostředí, tečka na začátku řádku vypadá dost osaměle).

Obě uvedená prostředí lze používat i jiným způsobem. K sazbě matematiky v textu lze použít prostředí `math` (pomocí `\begin` a `\end`) nebo jeho zkratky ve tvarech `\(` pro začátek prostředí a `\)` pro konec prostředí. K sazbě vysazené matematiky lze použít prostředí `displaymath` nebo zkratky `\[` a `\]`.

[18] V tomto a následujícím případě nepoužívejte svorky `\begin` a `\end`, uvedené symboly představují zkratky těchto svorek pro daná matematická prostředí (resp. příkazy TEXu).

Předchozí způsob vložení vysazené matematiky neumožňuje rovnice číslovat. K tomu je určeno prostředí `equation`, které se chová stejně jako předchozí, jen vysadí vpravo na řádku číslo rovnice uzavřené v závorkách.

Žádné z předchozích prostředí neumí vysázet více řádků než jeden a kombinování sledu samostatných vysazovacích prostředí je nevzhledné. Proto existuje prostředí `eqnarray`, které je schopno vysázet více řádků matematiky (rozdělovat musíte sami), přičemž přechodu na nový řádek dosáhnete použitím příkazu `\\`. Každému řádku bude přiřazeno právě jedno číslo; chcete-li, aby byl řádek bez čísla (např. u vzorce, který musel být rozdělen), musíte před přechodem na nový řádek zadat `\nonumber`. Pokud chcete celé prostředí nečíslované, zvolte prostředí `eqnarray*`.

Ve víceřádkovém prostředí jsou běžně výrazy zarovnávány tak, že nejdelší výraz je vystředěn a kratší jsou zarovnány na pravý konec nejdelšího. Chcete-li použít jiného zarovnání (což je většinou), musíte použít dva znaky `&`, mezi nimiž je část vzorce, na kterou má být středováno. Jako středovací symboly se zpravidla používají $=$, $\times$, $+$ a $-$ (například `a+b&=&c`). Požadujete-li, aby toto středování neplatilo pro jeden určitý řádek, napište jej ve formě `\lefteqn{`*výraz*`}\\`.

Poznámka 1: V české technické literatuře se někdy používá zarovnání vzorců vlevo. Viděli-li byste toto rádi i ve svém textu, račte použít volbu `fleqn` v `\documentclass`, ovšem nesmíte používat `$$`, ale `\[` a `\]`. Rovněž není-li vám libo číslování na pravé straně, může býti vlevo – stačí volba `leqno`.

Poznámka 2: Někdy potřebujeme vysázet místo čísla rovnice nějaký jiný „symbol“, který není automaticky generován. V tomto případě použijte příkazy TEXu, a to sice `\eqno` pro sazbu symbolu vpravo a `\leqno` pro sazbu vlevo. Tyto příkazy můžete použít v nečíslovaném vysazeném matematickém prostředí.

Velikost písma V matematických prostředích se používají čtyři velikosti (jsou uvedeny příkazy pro jejich zavedení)

1. `\displaystyle` – standardní velikost vysazené matematiky,
2. `\textstyle` – běžná velikost matematiky v odstavci,
3. `\scriptstyle` – velikost indexů první úrovně,
4. `\scriptscriptstyle` – velikost indexů druhé a vyšší úrovně.

3.2 Základní prvky matematické sazby

V této podkapitole jsou uvedeny některé základní příkazy matematické sazby, které je možno využít ve všech matematických prostředích.

Indexy Pro sazbu horního indexu slouží příkaz `^`, jehož parametr bude vysázen nahoře vedle předchozího symbolu, vysázení dolního indexu se provede příkazem `_`. Velikost vysázeného symbolu je o jeden stupeň menší než běžné matematiky. Příkazy pro sazbu indexů lze do sebe různě vnořovat (je třeba vložit závorky), zmenšování písma se provádí ale jen do druhé indexové úrovně. Příklad sazby x_i^2 je proveden jako `$x^2_i$`.

Všechny uvedené příkazy vkládají indexy **za** symbol, chceme-li vložit index před symbol (např. označení nukleonového čísla prvku), musíme index vztáhnout buď k začátku prostředí nebo k `{}` a případně druhý znak trochu „přitáhnout“, například `$^{57}\!$Co` dává ^{57}Co. Pokud potřebuje použít horní i dolní index zároveň, musíme si sami zajistit zarovnání na pravou stranu. V případě, že chceme index připojit k něčemu vyššímu, musíme to vysázet do složených závorek pomocí neviditelné matematiky, jinak bude index na úrovni odpovídající velkému písmenu (např. `{\vphantom{\int}}^x`).

Pokud chceme indexy použít v běžném textu, musíme psát `text$^{\mbox{index}}$`, což dá text$^{\mbox{index}}$ (úprava velikosti je na vás, v případě jednoslovného indexu lze použít třeba `text$^{\mathrm{index}}`). K dispozici je i příkaz `\textsuperscript{`*text*`}`, který vysází *text* jako $^{\text{horní index}}$ zmenšeným písmem.

Zlomky Zlomky lze vkládat ve dvou tvarech. Prvním je použití šikmé zlomkové čáry (která se dá vložit z číselné klávesnice), tento způsob je vhodný jen pro jednoduché zlomky a převážně v nevysazeném textu. Pro složitější vzorce slouží příkaz `\frac{`*čitatel*`}{`*jmenovatel*`}`, který vysází vodorovnou zlomkovou čáru, nad ní čitatele a pod ní jmenovatele. Tento příkaz je možno i několikrát vnořovat, případně kombinovat s předchozím způsobem sazby zlomku.

Odmocniny Odmocniny se vkládají jednoduše pomocí příkazu `\sqrt[`*n*`]{`*výraz*`}`, který vysází *n*-tou odmocninu z *výraz*u. Není-li *n* uvedeno, vysází se odmocnina bez označení řádu, tedy druhá.

Označení funkcí Funkce jako je např. sin se v textu označují běžnou antikvou (kupodivu i v „azbuce“). Pro jejich sazbu jsou zavedeny příkazy, které vždy před jménem funkce obsahují zpětné lomítko. Při jejich použití bude vytištěno jméno funkce a větší mezera před argumentem. Jsou předdefinovány následující funkce: `\arccos`, `\arcsin`, `\arctan`, `\arg`, `\bmod`, `\cos`, `\cosh`, `\cot`, `\coth`, `\csc`, `\deg`, `\det`, `\dim`, `\exp`, `\gcd`, `\hom`, `\inf`, `\ker`, `\lg`, `\lim`, `\liminf`, `\limsup`, `\ln`, `\log`, `\max`, `\min`, `\Pr`, `\sec`, `\sin`, `\sinh`, `\sup`, `\tan`, `\tanh`, jejichž význam se zdá být zřejmý. Pod každý z těchto výrazů je možno vepsat text jako index, např. $\lim_{x\to\infty}$, který se ve vysazeném prostředí píše přímo pod symbol a v běžném textu jako index vpravo. Chceme-li toto implicitní chování změnit (tak jako v předchozí ukázce), napíšeme před symbol indexu příkaz `\nolimits` nebo `\limits`.

Jako obvykle, v české sazbě budou nějaká specifika. Především se funkce tangens, cotangens a jejich inverze označují jako `tg`, `cotg`, `arctg` a `arccotg` (u inverzních funkcí případně i s mezerou mezi `arc` a zbytkem), a dále se pro označení hyperbolických funkcí používá (nebo spíše má používat) označení `ch`, `sh`, `th`. Chceme-li tyto výrazy používat, musíme si je nadefinovat jako nové matematické operátory, jako příklad budiž uvedeno `\newcommand{\arctg}{\mathop{\rm arctg}}`, čímž jsme nadefinovali nový příkaz `\arctg` pro sazbu funkce inverzní k tangentě. Tento příkaz můžeme používat naprosto stejně jako výše uvedené příkazy.

Nové definice musíme použít také pro běžně používané operátory, které nejsou standardně v LaTeXu zavedeny, např. operátory gradient, rotace a divergence (zde pozor, příkaz `\div` již existuje). Rovněž tak je vhodné jako operátor nadefinovat označení diferenciálu, které by mělo být taktéž provedeno antikvou. Zde je menší komplikace, protože se musíme zbavit mezery, kterou LaTeX vkládá mezi operátor a argument. Možná definice může být `\newcommand{\dd}{\mathop{\rm d\!}}`. Při používání `\mathop` je třeba mít na paměti, že TeX se pokusí jeho obsah vertikálně vycentrovat na matematickou osu, pokud se jedná o jeden jediný znak.

Poznámka: Přestože všechny proměnné a obecná čísla jsou označována mat. italikou, bývá zvykem Eulerovo číslo (e=2,7181…) označovat antikvou. Pokud tedy často používáme výrazy typu e^y, je vhodné zavést si nový příkaz, např. `\def\moce#1{{\rm e}^{#1}}`, který použijeme jako `\moce{y}`. Obdobně to platí pro imaginární jednotku i. Stojatým písmem se také označují například body a indexy, které mají význam slov nebo jejich zkratek, například maximální amplituda A_{m}.

Vložení textu Chceme-li do matematické sazby vložit text, musíme použít příkaz `\mbox{`*text*`}`, který jej vysází ve velikosti a řezu, který byl platný před začátkem matematického prostředí. Chceme-li vložit text například do zlomku či do indexu, musíme pamatovat, že velikost textu nebude automaticky změněna (pokud takový text vkládáme často, je možno jej definovat jako matematický operátor, tím se zmenší automaticky). Jako příklad lze uvést zlomek $\frac{\text{zde zmenšený není}}{\text{ale zde je}}$, který byl vysázen příkazem `$\frac{\mbox{zde zmenšený není}}{\mbox{\scriptsize ale zde je}}$`. Vložení textu, který se automaticky zmenšuje, je možno provést vložením příkazů pro změnu řezu (např. `\mathrm`), ale je zde několik rozdílů – jednak nelze vložit běžné mezery, jednak je nutno zadat přímo určitý řez (při použití `\mbox`u je řez stejný jako před matematikou).

Označování, svorkování Pro matematické výrazy se rovněž může použít podtržení a nadtržení. Příkazy pro jejich použití jsou `\underline{`*výraz*`}` a `\overline{`*výraz*`}`. Tyto příkazy se mohou kombinovat i vnořovat, například $\underline{\overline{x}}$.

V matematické sazbě je často nutno použít označení pro počet členů v některém výrazu. Vhodným prostředkem je použití svorek. Svorku umístěnou pod výrazem lze zadat příkazem `\underbrace{`*výraz*`}`, k němuž lze jako spodní index přidat vhodný text. Obdobně příkazem `\overbrace{`*výraz*`}` se vytvoří svorka umístěná nad výrazem, k níž lze dodat horní index. Příklad použití je

$$n * a = \underbrace{a + \cdots + a}_{n\text{-krát}}, \quad a^n = \overbrace{a a \cdots a}^{n\text{-krát}}.$$

Sazba nad sebe Pokud potřebujeme vysázet dva symboly nad sebe, jako např. u definičních rovností $\stackrel{\text{def.}}{=}$, můžeme použít příkazu `\stackrel{`*horní*`}{`*dolní*`}`, který ovšem sází horní symbol menším písmem. Pro vysázení kombinačních čísel lze použít příkaz `{`*horní* `\choose` *dolní*`}`. Chceme-li dostat totéž bez závorek, nahradíme `\choose` příkazem `\atop`, např.

$$\stackrel{H}{D}, \left(\begin{matrix} H \\ D \end{matrix}\right), \begin{matrix} H \\ D \end{matrix}.$$

Výpustky V příkladě se svorkami jste viděli využití výpustky. Tu používáme tam, kde je potřeba naznačit nějakou pravidelně se opakující „operaci“. K jejímu označení můžeme použít tyto tři příkazy:

- `\ldots` vyprodukuje tři tečky na úrovni řádku, používá se pro označení posloupnosti, např. $n = 0, 1, \ldots, n$, a pro naznačení neúplných čísel, např. $\pi = 3{,}1415\ldots$;
- `\cdots` vytvoří tři tečky přibližně uprostřed řádku a používá se pro označení vypuštěných operací, např. $x_1 + \cdots + x_n$;
- `\vdots` vytvoří svisle tři tečky a využívá se pro označení vypuštěných řádků v maticích;
- `\ddots` vytvoří tři tečky diagonálně (takto $\ddots$), což se používá pro naznačení diagonály v maticích.

Je definován ještě příkaz `\cdot`, který slouží k vytvoření tečky v polovině výšky řádku a používá se k označování vypuštěných argumentů funkcí, např. $f(\cdot)$.

Sazba matic Mnohdy je třeba sázet zarovnaná schémata – matice. K tomu slouží matematická tabulka – prostředí `array`, které má povinný parametr určující počet sloupců a jejich zarovnání (písmena `lrc`) a jeden volitelný, určující polohu matice vzhledem k okolí (`t` připojí matici k okolí horním okrajem, `b` dolním). Jednotlivé sloupce oddělujeme znakem `&`, pro ukončení řádku slouží `\\`.

Prostředí `array` vytvoří pouze uspořádání sloupců a řádků, nikoliv oddělovače. Ty musíme vložit příkazem `\left`*symbol1* pro levou stranu a `\right`*symbol2* pro pravou, přičemž symboly (obecně různé) mohou být následující[19]:

(	(	)	)	[	[	]	]
{	\{	}	\}	/	/	\	\backslash
⌊	\lfloor	⌋	\rfloor	⌈	\lceil	⌉	\rceil
⟨	\langle[20]	⟩	\rangle	|	|	‖	\\|
↑	\uparrow	↓	\downarrow	⇑	\Uparrow	⇓	\Downarrow
↕	\updownarrow	⇕	\Updownarrow				

Tyto symboly lze použít okolo jakéhokoliv výrazu a vždy mění svou velikost podle skutečné výšky výrazu. Oba příkazy `\left` a `\right` musí tvořit vždy pár, a to i na řádku (nelze mezi ně vložit konec řádku). Potřebujeme-li někde pouze jeden oddělovač, použijeme místo druhého tečku (např. `\left.`)[21]. Tímto způsobem ovšem mohou někdy vzniknout problémy při rozdělování řádků, protože výšky obou částí nemusí být stejné, a pak nejsou stejné ani oba oddělovače. Zde je nutno nastavit velikost „ručně“ přidáním vhodného příkazu z řady `\big`, `\Big`, `\bigg` a `\Bigg` před daný oddělovač, např. `\big(`. Druhou možností je uložit „vnitřní“ obsah nejprve do boxu a pak použít jen jeho výšku pomocí `\vphantom`.

Sazba intervalů Intervaly otevřené se zapisují pomocí obyčejných kulatých závorek. Uzavřený interval je nutné naznačit pomocí oddělovačů `\left<` a `\right>`, nestačí pouhé `<>`.[22]

Mezery v matematice V matematice se nepoužívají běžné textové mezery, například mezera vložená mezerníkem je zcela ignorována. Jsou definovány tyto mezery:

- `\;` je velká mezera,

[19] Uvedené symboly samozřejmě můžeme použít i bez `\left` či `\right`.

[20] Lze rovněž použít `<`, obdobně pro pravý symbol.

[21] V jednom z rozšíření TEXu – ε-TEXu – existuje i příkaz `\middle`, který umístí vhodně zvětšený oddělovač i uprostřed mezi `\left` a `\right`.

[22] Dle evropských norem se intervaly stejně sází jinak.

- `\:` je mezera prostřední velikosti,
- `\,` je úzká mezera,
- `\!` je záporná mezera,
- `\quad` je čtverčíková mezera,
- `\qquad` je mezera dva čtverčíky.

Sazba vět a definic V matematice bývá zvykem před definicí či větou uvádět klíčové slovo „Definice“ či „Věta“ a její číslo v tučném řezu a zbylý text je psán italikou. K tomuto účelu slouží příkaz `\newtheorem{`*jméno*`}[`*čítač*`]{`*vypsané jméno*`}[`*oddíl*`]`, kde *jméno* je symbolické jméno nového prostředí (např. `dukaz`), *vypsané jméno* je jméno, které bude vypsáno (např. `Důkaz`) a *čítač* je volitelný parametr, který je totožný s již nadefinovaným teorémem a oba teorémy budou mít společné průběžné číslování. *oddíl* je volitelný odkaz na některý oddíl textu, z něhož bude odvozeno hierarchické číslování (např. `subsection`) ve tvaru `číslo_sekce.číslo_věty`. Příkaz pro definici musí být umístěn v preambuli a nesmí v něm být oba volitelné parametry zároveň (alespoň mi to pak řve). Vlastní věta je vepsána do svorek `\begin{`*jméno*`}[`*název*`]` a `\end{`*jméno*`}`, *název* je text, který bude uveden v závorkách za číslem. LaTeX automaticky vytvoří ke každému samostatnému teorému čítač s názvem `\`*jméno*. Příkladem použití může být

```
%preambule
\newtheorem{defi}{Definice}
\newtheorem{veta}[defi]{Věta}
%text dokumentu
\begin{defi} Kůň je zvíře s nohama.
\end{defi} \begin{veta}[o nohách]
Každý kůň má právě čtyři nohy.
\end{veta} \begin{defi}
Kráva je zvíře s rohama.
\end{defi}
```

Definice 1 *Kůň je zvíře s nohama.*

Věta 2 (o nohách) *Každý kůň má právě čtyři nohy.*

Definice 3 *Kráva je zvíře s rohama.*

Neviditelná matematika V některých případech je vhodno upravit matematické výrazy tak, aby vypadaly opticky lépe. Tento případ může nastat např. při sazbě dvou odmocnin vedle sebe, přičemž jedna je o trochu větší než druhá (například $\sqrt{F}\sqrt{F_{x_x}}$). Zde bychom chtěli mít obě odmocniny stejně vysoké, případně i široké. Můžeme tedy použít příkaz `\phantom{`*výraz*`}`, který stanoví rozměry *výraz*u, započítá je, ale *výraz* **nevytiskne**. V uvažovaném případě nám ovšem stačí pouze svislý rozměr, použijeme tedy příkaz `\vphantom`. Uvedený příklad pak vypadá $\sqrt{F}\sqrt{F_{x_x}}$. Obdobně pracuje `\hphantom` s horizontálními rozměry.

Dalším neviditelným znakem je `\mathstrut`, který vloží čáru nulové šířky a nenulové výšky. Používá se v matematice k výškovému posunu výrazů, např. u složených zlomků – srovnejte $\frac{1}{1+\frac{1}{x}}$, $\frac{1}{1+\frac{1}{x}}$.

3.3 Matematické akcenty

V matematickém textu nelze použít běžné akcenty. Povolené akcenty a příkazy pro jejich dosažení jsou uvedeny v následující tabulce.

$\hat{o}$	`\hat{o}`	$\check{o}$	`\check{o}`	$\breve{o}$	`\breve{o}`	$\tilde{o}$	`\tilde{o}`	$\grave{o}$	`\grave{o}`
$\bar{o}$	`\bar{o}`	$\vec{o}$	`\vec{o}`	$\dot{o}$	`\dot{o}`	$\ddot{o}$	`\ddot{o}`	$\acute{o}$	`\acute{o}`

Příkazy `\hat` a `\tilde` mají k dispozici i široké verze, které je možno uvést přes několik znaků. Jsou jimi `\widehat{`*text*`}` a `\widetilde{`*text*`}`, např. $\widehat{x+y}$. Rovněž jsou k dispozici „šipkové“ akcenty s libovolnou délkou `\overleftarrow{`*výraz*`}` a `\overrightarrow{`*výraz*`}`, např. $\overleftarrow{abcdefgh}$. Pokud potřebujete umístit obyčejný akcent nad složený výraz, je třeba zvolit vhodný způsob zápisu, srovnejte např. $\vec{F_x}$ a $\vec{F}_x$ (`$\vec{F_x}$` a `$\vec{F}_x$`).

Pro označení derivace se používá pouze znak `'`.

3.4 Používané matematické symboly

Pro vkládání různých matematických symbolů má LaTeX předdefinovánu spoustu příkazů. K dispozici je řecká abeceda malá (α `\alpha`, β `\beta`, γ `\gamma`, δ `\delta`, ϵ `\epsilon`, ε `\varepsilon`, ζ `\zeta`, η `\eta`, θ `\theta`, ϑ `\vartheta`, ι `\iota`, κ `\kappa`, λ `\lambda`, μ `\mu`, ν `\nu`, ξ `\xi`, o `o`, π `\pi`, ϖ `\varpi`, ρ `\rho`, ϱ `\varrho`, σ `\sigma`, ς `\varsigma`, τ `\tau`, υ `\upsilon`, ϕ `\phi`, φ `\varphi`, χ `\chi`, ψ `\psi`, ω `\omega`) i velká (Γ `\Gamma`, Δ `\Delta`[23], Θ `\Theta`, Λ `\Lambda`, Ξ `\Xi`, Π `\Pi`, Σ `\Sigma`, Υ `\Upsilon`, Φ `\Phi`, Ψ `\Psi`, Ω `\Omega`). Písmeno, které je stejného vzhledu jako v antikvě, nemá zvláštní příkaz (značná část velkých písmen). Rovněž je k dispozici kaligrafické písmo pro všechna velká písmena ($\mathcal{ABCDEFGHIJKLMNOPQRSTUVWXYZ}$), které se aplikuje na argument příkazu `\mathcal`. Pro malá písmena jej nelze použít. Při použití balíku `eufrak` je k dispozici i $\mathfrak{fraktura}$, na kterou se přepíná příkazem `\mathfrak`.

V dalším budou uvedeny ty symboly, které nelze vysázet přímo z klávesnice. Obecně platí, že libovolný symbol (i textový), lze přeškrknout příkazem `\not` uvedeným před symbolem. Pokud tato operace nevypadá dobře, lze výsledek upravit použitím mezer (i záporných), např. `$\not\kern -.85ex\copyright$` dává $\not\copyright$ (doporučuji používat co nejčastěji[24]).

Větší množství matematických symbolů lze dosáhnout použitím matematického balíku `amssymb.sty`, který používá některé speciální fonty a využívá vpodstatě možností $\mathcal{AMS}$-TEXu. Rovněž je možno využít přímo fontu `msbm`, který obsahuje obrysová písmena ($\mathbb{ABCDEFGHIJKLMNOPQRSTUVWXYZ}$) a některé speciální znaky ($\nleqq\beth\Bbbk\eth\circledast$).

Šipky V následující tabulce jsou uvedeny šipky různých tvarů, přímo dostupné pomocí příkazů.

$\leftarrow$	`\leftarrow`	$\longleftarrow$	`\longleftarrow`	$\uparrow$	`\uparrow`
$\Leftarrow$	`\Leftarrow`	$\Longleftarrow$	`\Longleftarrow`	$\Uparrow$	`\uparrow`
$\rightarrow$	`\rightarrow`[25]	$\longrightarrow$	`\longrightarrow`	$\downarrow$	`\downarrow`
$\Rightarrow$	`\Rightarrow`	$\Longrightarrow$	`\Longrightarrow`	$\Downarrow$	`\Dowarrow`
$\leftrightarrow$	`\leftrightarrow`	$\longleftrightarrow$	`\longleftrightarrow`	$\updownarrow$	`\updownarrow`
$\Leftrightarrow$	`\Leftrightarrow`	$\Longleftrightarrow$	`\Longleftrightarrow`	$\Updownarrow$	`\Updownarrow`
$\mapsto$	`\mapsto`	$\longmapsto$	`\longmapsto`	$\nearrow$	`\nearrow`
$\hookleftarrow$	`\hookleftarrow`	$\hookrightarrow$	`\hookrightarrow`	$\searrow$	`\searrow`
$\leftharpoonup$	`\leftharpoonup`	$\rightharpoonup$	`\rightharpoonup`	$\swarrow$	`\swarrow`
$\leftharpoondown$	`\leftharpoondown`	$\rightharpoondown$	`\rightharpoondown`	$\nwarrow$	`\nwarrow`
		$\rightleftharpoons$	`\rightleftharpoons`		

Relace V této tabulce budou uvedeny relační znaky.

$\leq$	`\leq`	$\geq$	`\geq`	$\equiv$	`\equiv`	$\models$	`\models`
$\prec$	`\prec`	$\succ$	`\succ`	$\sim$	`\sim`	$\perp$	`\perp`
$\preceq$	`\preceq`	$\succeq$	`\succeq`	$\simeq$	`\simeq`	$\mid$	`\mid`
$\ll$	`\ll`	$\gg$	`\gg`	$\asymp$	`\asymp`	$\parallel$	`\parallel`
$\subset$	`\subset`	$\supset$	`\supset`	$\approx$	`\approx`	$\bowtie$	`\bowtie`
$\subseteq$	`\subseteq`	$\supseteq$	`\supseteq`	$\cong$	`\cong`	$\smile$	`\smile`
$\sqsubseteq$	`\sqsubseteq`	$\sqsupseteq$	`\sqsupseteq`	$\doteq$	`\doteq`	$\frown$	`\frown`
$\in$	`\in`	$\ni$	`\ni`	$\neq$	`\neq`		
$\vdash$	`\vdash`	$\dashv$	`\dashv`	$\propto$	`\propto`		

Operátory V této části jsou uvedeny příkazy pro binární operátory.

$\pm$	`\pm`	$\cap$	`\cap`	$\diamond$	`\diamond`	$\oplus$	`\oplus`
$\mp$	`\mp`	$\cup$	`\cup`	$\triangle$	`\bigtriangleup`	$\ominus$	`\ominus`
$\times$	`\times`	$\uplus$	`\uplus`	$\triangledown$	`\bigtriangledown`	$\otimes$	`\otimes`
$\div$	`\div`	$\sqcap$	`\sqcap`	$\triangleleft$	`\triangleleft`	$\oslash$	`\oslash`
$\ast$	`\ast`[26]	$\sqcup$	`\sqcup`	$\triangleright$	`\triangleright`	$\odot$	`\odot`
$\star$	`\star`	$\vee$	`\vee`	$\cdot$	`\cdot`	$\bigcirc$	`\bigcirc`
$\circ$	`\circ`[27]	$\wedge$	`\wedge`	$\bullet$	`\bullet`	$\dagger$	`\dagger`
$\amalg$	`\amalg`	$\setminus$	`\setminus`	$\wr$	`\wr`	$\ddagger$	`\ddagger`

Protože se $\cdot$ a $\bullet$ často používají i v textu, mají definovány i textové příkazy `\textperiodcentered` a `\textbullet`.

[23] Tento příkaz se používá rovněž pro označení Laplaceova operátoru.
[24] Častěji se však pro tyto účely využívá symbol „copyleft“, který obsahuje zrcadlově obrácené písmeno „c“ v kroužku.
[25] Synonymem je kratší příkaz `\to`, pro opačnou šipku `\gets`.
[26] Lze použít i `*` z klávesnice.
[27] Tento příkaz lze použít k označení stupňů (°C), je-li uveden v exponentu.

Velké symboly V této části jsou uvedeny symboly, které se vyskytují ve dvou velikostech podle toho, zda jsou ve vysazeném prostředí či v běžném textu. Pod a nad tyto symboly lze umísťovat různé indexy. Pokud je některý symbol shodný s velkým řeckým písmenem, mělo by být mezi nimi rozlišováno. Tyto symboly nejsou k dispozici ve všech velikostech, proto není vhodné uvádět je např. v indexech. V tomto výjimečném případě je zřejmě lepší např. použít místo znaku pro sumaci velké sigma.

Tyto symboly zpravidla slouží k označení mnohonásobného provádění operací uvedených za nimi, např. nekonečná sumace, direktní součin apod.

$\sum$	`\sum`	$\prod$	`\prod`	$\coprod$	`\coprod`	$\int$	`\int`	$\oint$	`\oint`
$\bigcap$	`\bigcap`	$\bigcup$	`\bigcup`	$\bigsqcup$	`\bigsqcup`	$\bigvee$	`\bigvee`	$\bigwedge$	`\bigwedge`
$\bigodot$	`\bigodot`	$\bigotimes$	`\bigotimes`	$\bigoplus$	`\bigoplus`	$\biguplus$	`\biguplus`		

Ostatní symboly Zde jsou uvedeny symboly, pro které by se těžko našlo smysluplné označení.

$\aleph$	`\aleph`	$\hbar$	`\hbar`	$\imath$	`\imath`	$\jmath$	`\jmath`
ℓ	`\ell`	$\wp$	`\wp`	$\Re$	`\Re`	$\Im$	`\Im`
$\prime$	`\prime`	$\emptyset$	`\emptyset`	∇	`\nabla`	$\surd$	`\surd`
$\top$	`\top`	$\bot$	`\bot`	$\|$	`|`	$\angle$	`\angle`
$\forall$	`\forall`	$\exists$	`\exists`	$\neg$	`\neg`	$\flat$	`\flat`
$\natural$	`\natural`	$\sharp$	`\sharp`	$\backslash$	`\backslash`	∂	`\partial`
∞	`\infty`	$\triangle$	`\triangle`	$\clubsuit$	`\clubsuit`	$\diamondsuit$	`\diamondsuit`
$\heartsuit$	`\heartsuit`	$\spadesuit$	`\spadesuit`				

Rozšířené symboly Tyto symboly jsou použitelné jen tehdy, je-li použit balík `latexsym`.

$\leadsto$	`\leadsto`	$\sqsubset$	`\sqsubset`	$\sqsupset$	`\sqsupset`	$\bowtie$	`\Join`
$\lhd$	`\lhd`	$\rhd$	`\rhd`	$\unlhd$	`\unlhd`	$\unrhd$	`\unrhd`
$\mho$	`\mho`	$\Box$	`\Box`	$\Diamond$	`\Diamond`		

Poznámka 1: V běžných symbolech nejsou vyhrazeny žádné příkazy pro obvyklé označování speciálních množin – dvojité R pro reálná čísla, dvojité N pro přirozená apod. Pro označení těchto množin se pak většinou použije pouze tučné bezpatkové písmeno, jako příklad definujme příkaz `\N` takto `\def\N{\mbox{\sffamily\bfseries N}}` a pak jej můžeme přímo používat (př. `$x\in\N$` dává $x \in \mathbf{N}$).

Poznámka 2: Pokud potřebujete použít nějaký jednodušší symbol, který zde není uveden a je možné ho poskládat z čar a již zavedených symbolů, je vhodné jej nadefinovat jako nový příkaz – lze použít například prostředí `picture` (viz další kapitola). Jako ukázku je možno použít např. $\sqcap\!\!\lrcorner$. Mezi „poskládané“ symboly patří i některé z dříve uvedených, např. $\hookrightarrow$ je poskládán ze symbolů $\leftarrow$ a $\rightarrow$ pomocí `\lhook\joinrel\rightarrow`, kde makro `\joinrel` slouží ke spojování symbolů – „přitáhne“ následující symbol.

Poznámka 3: Spoustu dalších symbolů (většinou už nematematických) lze nalézt v balíčcích z archívu CTANu, které nejsou zahrnuty do běžných distribucí TEXu[28]. Můžeme uvést balíček `ifsym`, který obsahuje časové symboly (hodiny), značky počasí, číslice sedmisegmentových zobrazovačů či stínované čtverce, `bbding` obsahuje různé geometrické symboly (hvězdy, křížky, květinky), ukazující ruce, nůžky apod. V balíčku `cryst` můžete nalézt krystalografické symboly nebo v balíčku `dice` naleznete symboly herních kostek. Dále uvedené symboly se zpravidla v distribucích vyskytují.

Symboly z balíku `amssymb`

Obecné symboly

$\hbar$	`\hslash`	$\Game$	`\Game`	$\lozenge$	`\lozenge`	$\measuredangle$	`\measuredangle`
$\Finv$	`\Finv`	$\blacktriangle$	`\blacktriangle`	$\blacktriangledown$	`\blacktriangledown`	$\blacklozenge$	`\blacklozenge`
$\bigstar$	`\bigstar`	$\complement$	`\complement`	$\eth$	`\eth`	$\varnothing$	`\varnothing`
$\diagup$	`\diagup`	$\diagdown$	`\diagdown`	$\backprime$	`\backprime`	$\angle$	`\angle`
$\ulcorner$	`\ulcorner`	$\urcorner$	`\urcorner`	$\llcorner$	`\llcorner`	$\lrcorner$	`\lrcorner`

[28] Velký přehled symbolů je uveden v „The comprehensive LaTeX symbol list“, který najdete v CTANu v adresáři `info/symbols/comprehensive`.

Šipky

$\dashrightarrow$	`\dashrightarrow`	$\dashleftarrow$	`\dashleftarrow`	$\leftleftarrows$	`\leftleftarrows`	$\rightrightarrows$	`\rightrightarrows`
$\Lleftarrow$	`\Lleftarrow`	$\twoheadleftarrow$	`\twoheadleftarrow`	$\leftarrowtail$	`\leftarrowtail`	$\looparrowleft$	`\looparrowleft`
$\curvearrowleft$	`\curvearrowleft`	$\circlearrowleft$	`\circlearrowleft`	$\curvearrowright$	`\curvearrowright`	$\circlearrowright$	`\circlearrowright`
$\Lsh$	`\Lsh`	$\Rsh$	`\Rsh`	$\twoheadrightarrow$	`\twoheadrightarrow`	$\rightarrowtail$	`\rightarrowtail`
$\rightsquigarrow$	`\rightsquigarrow`	$\leftrightsquigarrow$	`\leftrightsquigarrow`	$\multimap$	`\multimap`	$\nLeftrightarrow$	`\nLeftrightarrow`
$\nleftarrow$	`\nleftarrow`	$\nrightarrow$	`\nrightarrow`	$\nRightarrow$	`\nRightarrow`	$\nLeftarrow$	`\nLeftarrow`

Binární operátory a relace

$\dotplus$	`\dotplus`	$\Cap$	`\Cap`	$\Cup$	`\Cup`	$\intercal$	`\intercal`
$\barwedge$	`\barwedge`	$\veebar$	`\veebar`	$\curlywedge$	`\curlywedge`	$\curlyvee$	`\curlyvee`
$\boxminus$	`\boxminus`	$\boxplus$	`\boxplus`	$\boxtimes$	`\boxtimes`	$\boxdot$	`\boxdot`
$\leftthreetimes$	`\leftthreetimes`	$\rightthreetimes$	`\rightthreetimes`	$\ltimes$	`\ltimes`	$\rtimes$	`\rtimes`
$\leqq$	`\leqq`	$\lessapprox$	`\lessapprox`	$\lessgtr$	`\lessgtr`	$\curlyeqprec$	`\curlyeqprec`
$\succapprox$	`\succapprox`	$\thicksim$	`\thicksim`	$\gtrdot$	`\gtrdot`	$\approxeq$	`\approxeq`
$\lesseqgtr$	`\lesseqgtr`	$\risingdotseq$	`\risingdotseq`	$\fallingdotseq$	`\fallingdotseq`	$\precsim$	`\precsim`
$\bumpeq$	`\bumpeq`	$\ggg$	`\ggg`	$\lll$	`\lll`	$\eqcirc$	`\eqcirc`
$\shortmid$	`\shortmid`	$\shortparallel$	`\shortparallel`	$\blacktriangleright$	`\blacktriangleright`	$\blacktriangleleft$	`\blacktriangleleft`
$\because$	`\because`	$\between$	`\between`	$\therefore$	`\therefore`	$\lessdot$	`\lessdot`
$\backsim$	`\backsim`	$\Vvdash$	`\Vvdash`	$\Bumpeq$	`\Bumpeq`	$\circeq$	`\circeq`
$\doteqdot$	`\doteqdot`	$\backsimeq$	`\backsimeq`	$\triangleq$	`triangleq\`	$\gtrapprox$	`\gtrapprox`
$\Vdash$	`\Vdash`	$\pitchfork$	`\pitchfork`	$\backepsilon$	`\backepsilon`	$\eqslantless$	`\eqslantless`

Symboly z balíku `wasysym`

Tento balíček obsahuje některé symboly z oblasti fyziky, astronomie a některé další. Definuje také nový příkaz `\overstrike{`*co*`}{`*čím*`}`, který slouží k překrytí dvou matematických symbolů (pokud příkaz na začátku odstavce apod. nefunguje tak, jak by měl, dejte před něj příkaz `\leavevmode`). Balíček rovněž předefinovává některé speciální matematické symboly (vypadají ovšem stejně). Až na matematické symboly se používají bez matematického prostředí.

Obecné symboly

♂	`\male`	♀	`\female`	☎	`\phone`	🔔	`\bell`
►	`\RIGHTarrow`	◄	`\LEFTarrow`	▲	`\UParrow`	▼	`\DOWNarrow`
∼	`\AC`	≈	`\HF`	≋	`\VHF`	ϟ	`\lightning`
□	`\Square`	☑	`\CheckedBox`	☒	`\XBox`	¤	`\wasylozenge`
♪	`\eighthnote`	♩	`\quarternote`	𝅗𝅥	`\halfnote`	𝅝	`\fullnote`
♫	`\twonotes`	◆	`\ataribox`	¦	`\brokenvert`	✓	`\checked`
☺	`\smiley`	☹	`\frownie`	☻	`\blacksmiley`	‰	`\permil`
⬡	`\hexagon`	⬠	`\pentagon`	◯	`\octagon`	⬡	`\varhexagon`
✳	`\hexstar`	✳	`\varhexstar`	✡	`\davidsstar`	✠	`\kreuz`
○	`\Circle`	●	`\CIRCLE`	◖	`\Leftcircle`	◖	`\LEFTCIRCLE`
◗	`\Rightcircle`	◗	`\RIGHTCIRCLE`	◐	`\LEFTcircle`	◑	`\RIGHTcircle`
¤	`\currency`	⌀	`\diameter`	⦸	`\invdiameter`	⌕	`\recorder`
☼	`\sun`	🕑	`\clock`	∢	`\varangle`	➪	`\pointer`
ɔ	`\openo`	ə	`\inve`	þ	`\thorn`	Þ	`\Thorn`
¢	`\cent`	~~~~	`\photon`		`\gluon`		

Symboly matematické

≲	`\apprle`	≳	`\apprge`	⌐	`\invneg`	∝	`\wasypropto`
∫	`\varint`	∬	`\iint`	∮	`\varoint`	∯	`\oiint`

Astronomické symboly

♈	`\vernal`	☊	`\ascnode`	☋	`\descnode`	⊙	`\astrosun`
☾	`\leftmoon`	☽	`\rightmoon`	☌	`\conjunction`	☍	`\opposition`
☿	`\mercury`	♀	`\venus`	♁	`\earth`	♂	`\mars`
♃	`\jupiter`	♄	`\saturn`	♁	`\uranus`	♆	`\neptune`
		♇	`\pluto`				

Symboly zvěrokruhu

♈	`\aries`	♉	`\taurus`	♊	`\gemini`	♋	`\cancer`
♌	`\leo`	♍	`\virgo`	♎	`\libra`	♏	`\scorpio`
♐	`\sagittarius`	♑	`\capricornus`	♒	`\aquarius`	♓	`\pisces`

Symboly APL

⍐	`\APLuparrowbox`	⍗	`\APLdownarrowbox`	⍇	`\APLleftarrowbox`	⍈	`\APLrightarrowbox`
⍞	`\APLbox`	⍎	`\APLinput`	⍝	`\APLcomment`	⌹	`\APLinv`
⋆	`\APLstar`	⍟	`\APLlog`	¯	`\APLminus`		

4 Sazba obrázků

Vkládání obrázků do TEXu není zcela uspokojivě vyřešeno. Samotný TEX umí vkládat pouze znaky a vodorovné či svislé čáry, což je na slušný obrázek poměrně málo. Nicméně má zabudovány funkce, které mohou toto omezení obejít.

V LATEXu je pro sazbu čárové grafiky (pérovek) zavedeno prostředí `picture`, které má následující hlavičku: `\begin{picture}(`*x, y*`)(`*x0, y0*`)`, kde x a y jsou rozměry obrázku (kreslící plochy) vyjádřené jako násobek jednotky, která byla předtím zadána příkazem `\unitlength` *vel*. Parametry *x0* a *y0* jsou vztažné body obrázku a většinou se nezadávají. Zadaná velikost obrázku má pro TEX jen ten význam, že z nich určuje místo, které je nutné obrázku vyhradit. Překladač nijak nekontroluje, jestli jsou objekty umísťovány skutečně na vyhrazenou plochu. Bod (0,0) je umístěn v levém dolním rohu obdélníku, souřadnice objektů mohou nabývat i záporných hodnot.

Vkládané objekty mají spoustu omezení plynoucích ze způsobu, jakým jsou vytvářeny: jednotlivé elementy nejsou přímo vykreslovány, ale jsou poskládány ze znaků speciálních fontů. Zmíněná omezení jsou:

- šikmé čáry mohou mít jen konečný počet směrů a mají zespoda omezenou velikost;
- kružnice a plné kroužky jsou kreslitelné jen v určitých průměrech (malých velikostí);
- vkládané objekty mohou mít jen dvě tloušťky čar, mezi nimiž se přepíná příkazy `\thicklines` (0,8 pt) a `\thinlines` (0,4 pt), pouze vodorovné a svislé čáry mohou mít libovolnou šířku *vel* nastavitelnou příkazem `\linethickness{`*vel*`}`;
- rámečky mohou být kresleny jen ve směru vodorovném a svislém.

Jednotlivé objekty se na vhodné místo vkládají příkazem `\put(`*x, y*`){`*objekt*`}`, kde x, y jsou souřadnice vztažného bodu prvku *objekt*. Je možno vkládat tyto prvky:

- Úsečky příkazem `\line(`*m, n*`){`*délka*`}`, kde m, n jsou **celá** čísla z intervalu $\langle -6, 6\rangle$ bez společného dělitele, udávající směr úsečky $\phi = \operatorname{arctg} \frac{n}{m}$, a *délka* je průmět úsečky do vodorovné osy x; v případě svislé úsečky se jedná o celou její délku. Vztažným bodem je počáteční bod úsečky.
- Vektory lze vkládat příkazem `\vector`, který má shodnou syntaxi s příkazem pro úsečku, jen použitelný interval je $\langle 4, 4\rangle$, vztažným bodem je začátek vektoru, na jeho konci bude vykreslena šipka.
- Kružnice se vkládají příkazem `\circle{`*r*`}`, kde r je poloměr kružnice a vztažným bodem je její střed; maximální velikost kružnice je asi 15 mm.
- Vyplněné kroužky (disky) lze vložit příkazem `\circle*{`*r*`}`, kde maximální poloměr je asi 6 mm.
- Příkazem `\oval(`*šířka, výška*`)[`*část*`]` lze vložit ovál o dané výšce a šířce, vztažným bodem je střed oválu. Volitelný parametr *část* udává, která část oválu se skutečně vytiskne a mohou ji tvořit kombinace písmen `b` (spodní), `t` (horní), `l` (levá) a `r` (pravá); uvedeme-li dva parametry, vytiskne se jen čtvrtina oválu.
- Rámečky se vkládají pomocí příkazu `\framebox(`*šířka, výška*`)[`*pozice*`]{`*text*`}`, který vloží rámeček o daných rozměrech a do něj umístí *text* do *pozice*, což je kombinace až dvou písmen, která určují polohu (`rltbc`, písmena mají stejný význam jako výše, `c` je středování), bez jejich zadání bude text vystředěn; vztažným bodem je spodní levý roh rámečku;
- Vložení čistého textu se provede příkazem `\makebox`, který má stejnou syntaxi jako předchozí příkaz;
- Rámeček z přerušovaných čar se vloží příkazem `\dashbox{`*el*`}(`*šířka, výška*`)[`*pozice*`]{`*text*`}`, kde *el* udává délku čárek, z nichž bude čárkování vytvořeno (střídá se mezera a čára dané délky);
- Lze vysázet text do sloupců nad sebou pomocí `\shortstack[`*zar*`]{`*řádky*`}`, kde nepovinný parametr udává zarovnání textu (`r` nebo `l`) a *řádky* jsou vlastní sázené texty, jednotlivé řádky jsou odděleny `\\`.

Pro opakované vkládání stejných objektů je v prostředí vyhražen příkaz `\multiput (`*x, y*`)(`*dx, dy*`){`*n*`}{`*objekt*`}`, který vloží *n*-kráte stejný *objekt* tak, že první vloží do bodu (*x, y*) a další vždy oproti předchozímu posune o (*dx, dy*). Pro toto vkládání je dobré si uvědomit, že souřadnice zde vložených objektů se budou vztahovat k aktuálnímu bodu, tzn. že souřadnice jsou uvnitř *relativní* (např. bude-li *text* obsahovat příkaz `\put(-1,0){\circle{1}}`, vloží se kroužek tak, aby byl jeho střed o jeden dílek vlevo, než jak určuje nastavení příkazu `\multiput`).

4.1 Bèzierovy křivky

Speciálním prvkem prostředí jsou příkazy `\qbezier(`*xA, yA*`)(`*xB, yB*`)(`*xC, yC*`)`, které se vkládají přímo (bez příkazu `\put`) a používají absolutní souřadnice. Tento příkaz vykreslí Bèzierovu křivku druhého stupně s počátečním bodem A, koncovým bodem B a kontrolním bodem C, která je parametricky popsaná množinou bodů

$$P(t) = (1-t)^2 A + 2t(1-t)C + t^2 B,$$

kde $t \in \langle 0, 1\rangle$ je parametr. Bude-li se kontrolní bod shodovat s některým krajním či bude ležet přesně na jejich spojnici, vykreslí příkaz pouze čáru (tímto příkazem je tedy možno vykreslit i čáry, na které není možno použít příkaz `\line`). Chceme-li vykreslit spojitou funkci pomocí těchto křivek, musí být koncový bod prvé a počáteční bod druhé křivky totožný, chceme-li i spojitou první derivaci (hladké navazování) musí být totožné dva body.

Vzhledem ke kompatibilitě se staršími verzemi LaTeXu existuje ještě příkaz `\bezier`, který musí mít jako první parametr uveden ve složených závorkách počet bodů, které se z křivky vykreslí; ostatní parametry jsou stejné jako u `\qbezier`. Příkaz `\qbezier` může rovněž vykreslovat pouze body na křivce, bude-li mu na prvním místě zadán v hranatých závorkách požadovaný počet bodů.

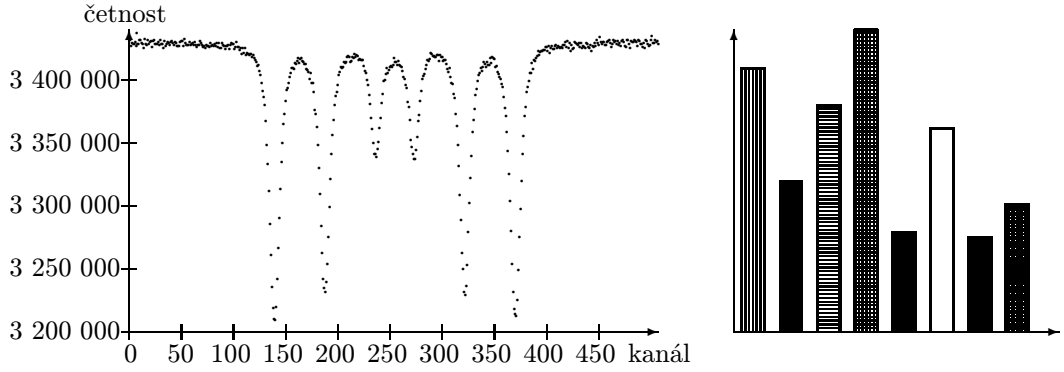
Bèzierovy křivky jsou ovšem vykreslovány tak, že je křivka rozdělena na sled bodů a každý bod je nakreslen jako černý čtvereček. Tato metoda je sice univerzální, avšak zabírá značné místo v paměti a může vést k ukončení překladu z nedostatku paměťové kapacity.

Vložení grafu naměřených hodnot Do textu lze docela obstojně vložit graf sestavený z jednotlivých naměřených hodnot tím, že vytváříme jednotlivé body příkazem `\put` a umísťujeme např. kolečka. Jediný problém představuje přepočet souřadnic jednotlivých bodů. Pokud máme na osách stejná měřítka, lze přepočetní jednotku zahrnout do hodnoty v příkazu `\unitlenght`. Ale ani jinak není problém vytvořit např. v Pascalu krátký prográmek pro převod dat z tabulky do grafu. Pro rozsáhlé soubory je vhodné nevkládat text přímo do zdrojového souboru, ale nechat jej (samozřejmě s příkazy TeXu) v jiném souboru a zde ho jen načíst příkazem např. `\input{obrazek.tex}`. Zde uvedený graf zabírá asi 100 řádků zdrojového textu.

Jednoduchým způsobem lze také vložit do textu sloupcové grafy, ovšem s podmínkou, že budou pouze šrafovány, nikoliv v odstínech šedi. Způsobem uvedeným na obrázku je možno vložit asi 8 druhů sloupců, při větším počtu je nutno spoléhat na jemnější odlišení šrafování, což není dvakrát vhodné.

Velmi jednoduché makro na vložení uvedeného grafu je uvedeno dále. Nejprve byl graf inicializován příkazem `\SetKonst{1}{1}{\circle*{0.33}}`, který nastavuje stejné měřítko 1 jednotka/`\unitlength` na obou osách a bod je označován plným kroužkem. Vlastní data jsou pak vykreslena příkazem `\put(0,0){\data{38.5545 0.138067 37.8048 0.276134 ...}}`.

```
\newdimen\kx\newdimen\ky\newtoks\sto
\def\SetKonst#1#2#3{\global\kx=#1\unitlength\global\ky=#2\unitlength\global\sto={#3}}
\def\data#1{\vynes#1 :;}
\def\vynes#1 #2 #3;{\put(0,0){\raise#2\ky\hbox{\kern#1\kx \the\sto}}%
\let\next=\vynes \if:#3 \let\next=\konec \fi \next#3;}
\def\konec#1;{\relax}
```



Obrázek 1: **a)** Mössbauerovské spektrum — ukázka vložení grafu, **b)** Ukázka sloupcových grafů

Použitím balíku `curves` můžeme vykreslovat i spojité grafy pomocí sítě zadaných bodů. Používá se pro ně příkaz `\curve(`*data*`)`, který vytváří parabolické úseky mezi zadanými body. Vlastní *data* jsou střídavě hodnoty x a y, oddělené čárkou, v jednotkách `\unitlength`. Vystačí-li se s prokládáním přímkami, lze použít balík `curvesls`, který je samozřejmě rychlejší. Pro uzavřené křivky je možno použít podobný příkaz `\closecurve`. Před vykreslením je možno zadat příkaz `\curvewarnfalse`, který zabrání vypisování hlášek o použití přímých úseků místo parabolických (možné zhoršení).

4.2 Vložení externích obrázků

Přestože vlastní TEX neumí zpracovávat např. bitmapové obrázky, umožňuje jejich vložení. To se děje použitím příkazu `\special{`*parametry*`}`. Tento příkaz nevykonává žádnou funkci, pouze do výstupního souboru s příponou `dvi` vloží *parametry* (pokud je ovšem v parametrech nějaký příkaz, pak jej překladač vykoná – proto případné cesty k souborům zadávejte s normálními lomítky `/`). Další zpracování těchto parametrů už je na zobrazovači. A v tom je právě kámen úrazu, protože tyto příkazy nejsou nijak standardizovány. V instalaci emTEXu všechny zobrazovače rozumí příkazům ve tvaru `em:`*příkaz*. Lze tedy použít např. `\special{em:graph soubor.pcx}` k vložení obrázku ze souboru `soubor.pcx`. Problém je ale v tom, že TEX neví, že se bude nějaký obrázek vkládat a nevynechá pro něj žádné místo – to už musíte zajistit sami (např. příkazem `\vspace`). Nejvhodnější místo pro umístění takto vloženého obrázku je uvnitř prostředí `figure`. Dalšími příkazy jsou `em:point`, pomocí něhož můžeme na stránce nadefinovat body, a příkaz `em:line`, který libovolné dva zadané a předem nadefinované body spojí čárou, aniž by pro ni platila jakákoliv omezení (platná v LATEXu).

Aby bylo možno dosáhnout určité přenositelnosti, bývají používané příkazy standardizovány. Nejrozumnější řešení je však použít nějaký balík, který definuje „abstraktnější“ příkaz pro vložení obrázků, který v závislosti na předpokládaném zobrazovači použije odpovídající příkaz `\special` (např. balíky `graphicx`, `pgf`). Ve spojení s vhodným balíkem a zobrazovačem PostScriptu můžete příkazem `\special` provádět např. rotace textu, jeho zvětšení, používat barvy či zrcadlový tisk.

Vložení obrázku vedle textu Někdy chceme dosáhnout toho, aby byl obrázek umístěn vedle textu, ne pod ním. V tomto případě můžeme použít dvě prostředí `minipage`, která budou mít shodná umístění vůči textu (volitelné parametry, např. obě `t`) a součet jejich šířek nepřekročí šířku stránky. Do jednoho prostředí vložíme text, do druhého obrázek (i pomocí `\special`). Obě prostředí **nesmíme oddělit** znakem pro nový odstavec či přechod na nový řádek (např. vynecháním řádku), pro pěknější vzhled (zarovnání obou stran) je vhodné mezi ně vložit příkaz `\hfill`.

Chceme-li navíc dosáhnout začlenění obrázku do odstavce (obtékání obrázku textem), bude to trochu složitější operace, kterou lze vyřešit dvěma způsoby. První použijeme tehdy, chceme-li ponechat několik prvních řádků původní velikosti a ostatní, až do konce odstavce, nějak pozměnit. K těmto účelům slouží příkazy (zadávané na začátku odstavce) `\hangindent` *vel*, kterým udáme délku zkrácení řádku zleva (při záporné hodnotě *vel* zprava) a `\hangafter` *n*, který udává počet *n* řádků, které budou mít původní velikost. Druhým způsobem „tvarování“ odstavce (chceme-li, aby po obrázku měl text opět původní velikost nebo chceme-li vložit obrázek nepravidelného tvaru) je použití příkazu `\parshape` *n a1 b1 a2 b2 ... an bn*, v němž je *n* počet řádků, kterých se zpracování týká (udává zároveň počet dvojic *a, b*), dvojice délek *a, b* pak pro každý řádek udává velikost odsazení zleva a celkovou šířku řádku (chceme-li tedy odsazení zprava, položíme všechny *a* rovny `0 mm` a zmenšíme šířku řádku). Následující řádky až do konce odstavce budou mít tvar zadaný poslední dvojicí *a,b*.

Pokud si chceme ušetřit trochu práce, můžeme použít balíček `picinpar`. Ten vytvoří nové prostředí, které umožní vložit obrázek do *jednoho* odstavce. Zvolený odstavec se začne jako `\begin{window}[`*počet*, *zar*, *obsah*, *popisek*`]` a ukončí `\end{window}`. Na začátku odstavce se vynechá *počet* řádků (smí být i nula) a pak se dle *zar*ovnání (`l` vlevo, `r` vpravo, `c` uprostřed) vloží *obsah* (nemusí být jen obrázek, lze vložit cokoliv), který určuje šířku obrázku. Pod obrázek lze vložit *popisek*, jehož šířka se přizpůsobí šířce *obsah*u. Pokud je výsledek kratší než odstavec, pokračuje odstavec dále v celé šířce, v opačném případě mohou nastat problémy.

Použití pdfTEXu Pokud chceme vložit jednoduše barevný obrázek, nebo chceme mít jako výstup pouze jeden soubor (bez nutnosti přenášet extra obrázky), můžeme použít pdfTEX, který produkuje výstup ve formátu PDF místo DVI. Tento výstup pak lze prohlížet např. programem ACROBAT READER, který je volně šiřitelný a pro několik operačních systémů. Nevýhodou tohoto postupu je poněkud větší

délka výstupního souboru (oproti Dvi) a trošku těžkopádnější práce s prohlížečem Pdf, výhodou je naopak vložení obrázků i fontů do jednoho souboru. Chceme-li využít této možnosti, použijeme k překladu program `pdfcslatex`, který rozumí stejným příkazům jako LaTeX, ale navíc je o některé rozšířen s ohledem na formát Pdf (především obrázky a hypertext – klikací odkazy).

Pro vložení obrázku použijeme příkaz `\pdfximage width` *šířka* `height` *výška* {*soubor*}, který vloží (prozatím jen do paměti) obrázek ze *soubor*u (podporuje formáty Jpeg, Png, Pdf) s danými rozměrovými parametry. Není-li některý rozměr uveden, dopočítá se z druhého při zachování poměru stran, chybí-li oba parametry, vloží se obrázek ve vlastní velikosti. Chceme-li obrázek vložit do výstupu, použijeme k tomu příkaz `\pdfrefximage` *n*, kde *n* je číslo obrázku, který byl načten do paměti některým z příkazů `\pdfximage`. Chceme-li vložit poslední načtený obrázek (asi nejčastěji), lze místo *n* použít příkazu `\pdflastximage`.

Drobnou nevýhodou pdfTeXu je jeho neslučitelnost s dalším způsobem vkládání obrázků a nefunkčnost některých balíčků, které využívají postscriptové příkazy, kterým pdfTeX nerozumí. Konkrétně u obrázků to mnohdy znamená nutnost převodu z formátu Eps do Pdf.

Využití PostScriptu Do TeXu můžete vkládat i obrázky v postscriptové verzi, a to dvěma způsoby. První využívá balíku `graphicx`, který obsahuje příkaz `\includegraphics[`*parametr=hodnota*`]{`*soubor*`}` pro vložení obrázku ze *soubor*u. Vložení obrázku se ovlivní případnými parametry (je možno zadat i více párů parametr=hodnota, které se oddělí čárkou). Zadáme-li parametr `width` nebo `height` obrázek se zvětší tak, aby zaujímal odpovídající rozměry. Pokud zadáme jen jeden z rozměrů, dopočítá se druhý tak, aby se zachoval poměr stran. Vynecháme-li oba rozměry, vloží se v původní velikosti. Zadáme-li parametr `scale`, zvětší se obrázek podle zadané hodnoty. Velikost obrázku můžeme změnit taky tak, že příkaz `\includegraphics` vložíme do druhého parametru příkazu `\scalebox{`*číslo*`}{`*co*`}`, který jej zvětší *číslo*krát. Při tomto způsobu vkládání není zapotřebí vynechávat místo, obrázek jej zaujme automaticky.

Druhý způsob využívá balík `epsf`, který obsahuje příkaz pro vložení `\epsffile[`*výřez*`]{`*soubor*`}` (zde je velikost výřezu udána pouze čísly bez jednotek). Tomuto příkazu mohou předcházet příkazy `\epsfxsize=`*šířka* a `\epsfysize=`*výška*, které zajistí vhodnou velikost obrázku (jsou-li uvedeny oba, může dojít ke zkreslení obrázku), místo pro sebe si obrázek vyhradí automaticky.

Pro oba způsoby vkládání je důležité, aby obrázky byly ve tvaru zapouzdřeného PostScriptu (Eps). Není-li tomu tak, může docházet ke vzájemnému překreslování textu a obrázku. Některé generátory však nevytváří korektní Eps formát a k překreslování dochází také, nejčastěji po převodu Dvi do Ps. Zde je možno si pomoci „přepracováním“ obrázku pomocí Ghostscriptu příkazem `gs -sDEVICE=epswrite -sOutputFile=vystup.eps vstup.eps`. Původní velikost obrázku je v postscriptovém souboru uvedena na řádku obsahujícím text `%% BoundingBox`, za nímž následuje čtveřice čísel, udávající rozměr (v jednotkách $\frac{1}{72}$ palce). Používáme-li balík `epsf`, je vhodné zapnout ořezání obrázku na BoundingBox příkazem `\epsfclipon` (je-li dostupný).

Čtyři čísla, která za textem `BoundingBox` následují, udávají x a y polohu levého dolního a pravého horního rohu (mohou být i záporné). Tento údaj je pak použit k výpočtu místa, které se má pro obrázek vynechat. Změníte-li tyto hodnoty, můžete upravit například velikost mezery mezi textem a obrázkem (po převodu do PostScriptu se obrázek zobrazí celý a může být třeba podkladem následujícího textu, případně můžete snadno odstranit velké okraje, které vkládají některé převodníky).

PSTricks Pokud chceme vytvořit jen jednoduchý obrázek, nemusíme ho vytvářet v externím editoru a vkládat jako Eps, ale můžeme jej vykreslit přímo pomocí postscriptových příkazů. To je však velmi nepohodlné, ale existuje balíček `pstricks`, který nám práci značně ulehčí. Umožní použití barev, definuje příkazy typu `\psline`, `\psellipse`, `\pspolygon` či `\dataplot`. Více se dovíte v dokumentaci k balíčku.

5 Speciální použití

V této kapitolce budou uvedeny některé speciálnější příkazy a funkce, které se tak hojně nepoužívají.

Dělení slov TEX sám provádí dělení slov na konci řádku podle zapnutého jazyka. Pokud se však vyskytne některé slovo, které rozděluje špatně, je možno mu v daném slově nabídnout vhodná místa k rozdělení příkazem `\-`, tedy například `vý\-jim\-ka`. Pokud jde o slovo, které se častěji vyskytuje, je možno jej uvést do seznamu výjimek, který může být uveden v preambuli: `\hyphenation{`*vzory*`}`, kde *vzory* je seznam výjimek oddělených mezerami, v nichž je **přípustné** dělení naznačeno pomlčkou `-`. Pokud je slovo nedělitelné, uvede se bez jediné pomlčky. V běžném textu je možno nedělitelný výraz uvést jako parametr příkazu `\mbox`. Ve speciálních případech lze použít příkaz `\discretionary{`*před*`}{`*za*`}{`*bez*`}`, který část *bez*, vypsanou v případě, že se slovo nebude v daném místě dělit, nahradí v případě dělení částí *před* na konci řádku a *za* na začátku nového řádku. Kupříkladu `Zu\discretionary{k-}{k}{ck}er` způsobí správné rozdělení německého slova „Zucker“. Příkaz lze využít i ke kvalitní sazbě velkých čísel a intervalů. Vložíte-li na začátek dokumentu řádky `\def\tisic{\discretionary{}{tisíc}{\kern.2em000}}` a `\def\az{\discretionary{\kern.3333em až}{}{--}}`, pak můžete psát třeba `345\tisic` a `10\az 20`, což způsobí v prvém případě sazbu textu „345 000“, vejde-li se na jeden řádek, jinak se vysází „345“ na konci řádku a na novém řádku bude text „tisíc“, což je daleko vhodnější než řádek začínající na „000“. Ve druhém případě se vysází buď „10–20“ na jednom řádku, nebo, po rozdělení, bude na prvním řádku „10 až“ a na druhém „20“.

Odkazy V běžném textu se občas potřebujeme odkázat na stránku, kde bylo něco důležitého (nebo taky ne, ale s odkazy to vypadá líp) uvedeno, nebo na vzorec či položku v seznamu. K tomuto účelu slouží v LATEXu **reference**. Označení místa, na které se budeme odkazovat, se provede pomocí příkazu `\label{`*návěští*`}`, kde návěští je symbolické jméno, pomocí kterého se budeme odvolávat (může obsahovat i diakritiku, je citlivé na velikost písmen). Co se konkrétně bude pod referencí rozumět, záleží na umístění příkazu:

- v běžném textu bude odkazovat na číslo oddílu (podle umístění může ukazovat na kapitolu (např. 3), na sekci (např. 3.1) atd.), samozřejmě se uplatní pouze číslované oddíly;
- v matematických prostředích bude odkazovat na číslo vzorce (musí být zvoleno prostředí, které číslování umožňuje – `eqnarray`, při uvedení v obyčejné matematice odkazuje na sekci);
- v plovoucích objektech odkazuje na číslo obrázku či tabulky (**Pozor**, zde musí být `\label` uvedeno až po `\caption`);
- v číslovaných výčtech (`enumerate`) odkazuje na číslo položky;
- v teorémech definovaných uživatelem bude odkazovat na číslo teorému.

K dosazení odkazu na příslušné místo použijeme buď příkaz `\ref{`*návěští*`}` nebo `\pageref{`*návěští*`}`, podle toho, chceme-li se odkázat na číslo objektu (viz předchozí výčet), nebo na číslo stránky, kde se návěští nachází. Vzájemná poloha definice návěští a odkazu může být libovolná, pokud odkaz předchází definici návěští, musí se překlad provést několikrát. Odkazy produkují pouze číslo, všechny závorky či označení „str.“, „obr.“ apod. se musí vložit samostatně (je vhodné mezi zkratku a vlastní odkaz vložit nezalomitelnou mezeru `~`). Abyste si nemuseli pamatovat názvy všech návěští, můžete použít balík `showkeys`, který v místě každého příkazu `\label` připevní obdélník se jménem návěští.

Grafické úpravy textu V některých případech je vhodné, aby byl text nějak graficky upraven. Jako nejjednodušeji použitelné se v TEXu jeví použití orámování a sazba na box pevné šířky. K vytvoření rámečku okolo textu je možno použít příkaz `\frame{`*text*`}`, který vysází rámeček okolo textu bez vynechání mezery (např. ☒). Druhou možností je použití příkazu `\fbox{`*text*`}`, který kousek místa vynechá (např. ☒). Dále existuje příkaz `\framebox[`*šířka*`][`*pozice*`]{`*text*`}`, který orámuje box o dané šířce a do *pozice* (zarovnání `lrc`) vloží *text* (Pozor, tentýž příkaz v prostředí `picture` má jinou syntaxi). Tloušťku rámovací čáry lze nastavit příkazem `\fboxrule`.

Chceme-li vložit text tak, aby byl blokově zarovnán (tzn. aby zabíral stanovenou šířku bez ohledu na skutečné rozměry textu), můžeme použít příkaz `\makebox[`*šířka*`][`*pozice*`]{`*text*`}`, který má tytéž parametry jako `\framebox`.

U všech blokových příkazů musíme sami provést zalomení řádků, přesahuje-li např. orámovaný text přes okraj, bude tak i vytištěn – nedojde k samovolnému rozdělení. K odstranění tohoto problému je možno buď změnit slovosled tak, aby se text na jeden řádek vešel, nebo text vhodně rozdělit do dvou rámečků.

Poslední grafickou úpravou textu může být posunutí písmen. Ve svislém směru lze použít příkaz `\raisebox`, který je popsán u prostředí `tabular`, ve směru vodorovném příkaz `\kern` *vel*, kde *vel* je délková hodnota udávající posunutí, která by měla být nejlépe vyjádřena v relativních jednotkách (vztažených k rozměrům některého znaku, např. `ex`). Vytvoření dvojitého R příkazem `I\kern -0.35ex R` dává IR. Tímto způsobem fungují např. příkazy `\TeX` a `\LaTeX`. Při používání posunutí ve vodorovném směru ovšem nesmíte zapomenout, že se posune i všechen další text, navíc je na vás ohlídání zarovnání textu.

Použití barev Pokud chcete v textu používat barvy, musíte použít balík `color` a vhodný zobrazovač, popř. převést výsledný soubor do PostScriptu (programem `dvips`) či Pdf (nebo program překládat pdfTEXem). Ke zvolení barvy slouží příkaz`\color{`*barva*`}`, kde *barva* je jméno barvy. Může jít o barvu standardně nadefinovanou (`blue`, `red`, apod.) nebo definovanou pomocí příkazu `\definecolor{`*jméno*`}{`*model*`}{`*data*`}`, který vytvoří barvu *jméno* pomocí *model*u (`gray`, `rgb`, `cmyk`). *data* závisí na zvoleném modelu, vždy se jedná o čísla v intervalu $\langle 0, 1\rangle$, popř. oddělená čárkami, která vyjadřují intenzitu příslušné složky. Například `\definecolor{seda}{gray}{.5}` nadefinuje „poloviční“ šeď, totéž lze (přibližně) nadefinovat jako `\definecolor{seda}{rgb}{.5,.5,.5}`. Příkaz `\pagecolor{`*barva*`}` slouží obdobně k nastavení barvy pozadí stránky.

Prolínání písmen Někdy potřebujeme použít jednoduchý přepis písmen, při němž chceme přes sebe vytisknout dva symboly. K tomu můžeme využít příkazu `\rlap{`*text*`}`, který vloží *text* s **nulovou** šířkou tak, aby se levý okraj textu kryl s polohou boxu. Obdobně pracuje příkaz `\llap`. Jako příklad lze uvést `{\sffamily Y\llap{T}}`, který dává Ⅴ, či `$\wr\:\llap{$\sim$}$` pro ≀. Místo takových nesmyslů je možno příkazu využít pro vložení textu na jinak „prázdné“ řádky, vyplněné pomocí pružných mezer. Také můžeme definovat příkaz pro vytvoření podtečkovaného textu nebo přeškrknutého ~~textu~~. Někdy je potřeba použít přeškrtnutých písmen, nejčastěji v kvantové mechanice. Standardně je k dispozici symbol $\hbar$ (symbol pro Diracovu konstantu), ale schází například symbol pro redukovanou vlnovou délku $\lambda\!\!\!^{-} = \lambda/2\pi$. Tento symbol byl dodefinován příkazem `\def\lambar{{\mathchar'26\mkern-9.5mu\lambda}}`.

Uvedené příklady lze použít i k vytváření stínovaných nadpisů, např. `\def\nadpis#1{\leavevmode \rlap{\lower .4ex\hbox{\color{stín}\hskip.2ex{#1}}}{\color{text}{#1}}}`, kde `stín` je jméno barvy stínu a `text` barva textu (samozřejmostí je použití balíku `color`). Konstanty `.4ex` a `.2ex` určují míru posunutí stínu. Uvedený příklad lze použít pouze na text, který neobsahuje konec řádku.

PostScriptové manipulace Je-li použit balík `graphicx` a použitý prohlížeč podporuje PostScript, lze s textem provádět různé kejkle. Příkazem `\reflectbox{`*text*`}` lze napsat *text* zrcadlově (horizontálně) ʇxǝʇ a příkazem `\rotatebox{`*úhel*`}{`*text*`}` lze *text* otočit o *úhel* (ve stupních) ve směru hodinových ručiček.

Černé obdélníky Vyplněné obdélníky je možno vytvořit příkazem `\rule[`*zvýšení*`]{`*šířka*`}{`*výška*`}`, kde *šířka* a *výška* jsou rozměry obdélníku (případně čáry) a parametr *zvýšení* udává, o kolik bude obdélník posunut nad účaří. Čára nad obsahem tohoto textu byla vytvořena příkazem `\rule{\textwidth}{1mm}`. Příkaz `\rule` bohužel není možno vložit do některých jiných příkazů, jako například `\caption`. Zde si musíme pomoci primitivem TEXu `\vrule height` *výška* `width` *šířka* `depth` *hloubka*, který vloží vertikální čáru požadovaných parametrů (nejsou povinné).

Obdélník o šířce 0 pt je neviditelná čára – strut a používá se ke změně velikostí, protože její výška je při překladu započítávána. Může se použít například ke zvýšení orámování textu nebo k většímu odsazení vysokých nápisů v tabulkách.

Hypertextové odkazy Chcete-li ve svém textu (šířeném elektronicky) využívat hypertextových odkazů, je možno použít balíku `hyperref`. Pro vložení odkazu je zde nadefinován příkaz `\href{`*url*`}{`*text*`}`, který způsobí, že *text* bude odkazovat na adresu *url*, bude-li to však podporovat použitý zobrazovač. Uvedený příklad lze použít rovněž ve spojení s pdfTEXem, kde balík vytvoří ještě systém záložek podle struktury sekcí v dokumentu (`section`, `subsection`, ...). Ve standardním Dvi souboru pak vytvoří odkazy z obsahu textu, odkazů na literaturu, k obrázkům a tabulkám. Příkladem použití může být

`\href{http://www.cstug.cz}{odkaz}` odkaz na stránky Československého sdružení uživatelů TEXu. Jsou-li parametry *url* a *text* shodné, lze použít `\url{`*url*`}`, což *url* zároveň vysadí strojopisem.

Odkazy lze použít i uvnitř dokumentu. Cíl odkazu nadefinujeme jako `\hypertarget{`*jméno*`}{`*cíl*`}`, kde *jméno* je symbolický název a *cíl* je obsah cíle (v závislosti na prohlížeči na něj bude „nasměrován“ pohled, zpravidla se nejedná o pouhý přechod na stránku). Odkaz na *cíl* nadefinujeme příkazem `\hyperlink{`*jméno*`}{`*odkaz*`}`, kde *odkaz* je obsah klikacího odkazu.

Styly V předchozím textu bylo používání stylů (balíků) omezeno na jejich uvedení v parametru příkazu `\usepackage`, styly však při načítání umožňují více – použití příkazu s volitelným parametrem, `\usepackage[`*volby*`]{`*styl*`}`, kde *styl* je název stylu (jméno souboru bez přípony `sty`) a *volby* jsou volitelné parametry udávající konfiguraci stylu. Například styl `czech` může mít volitelný parametr `nocaptions`, který zamezí předefinování různých názvů českými ekvivalenty (chapter → kapitola, figure → obrázek) nebo parametr `split`, který umožní přenášení pomlčky při dělení slov typu „propan-butan“ apod.

Uvedený příkaz může mít jako poslední volitelný parametr uvedeno datum vytvoření. Dojde-li k překladu souboru či balíku staršího než uvedené datum, bude vypsáno varování.

Poznámka 1: Chcete-li mít dobrou přenositelnost zdrojových souborů mezi různými systémy, musíte pamatovat na to, že názvy stylů jsou zároveň jména souborů. Některé operační systémy (např. Unix) rozlišují mezi malými a velkými písmeny, je nutno tedy tak dělat i ve zdrojových textech, i když např. v MS DOSu se rozdíl neprojeví. Obdobné problémy se mohou vyskytnout v systémech, které mají omezení délky souboru (např. DOS omezuje soubory na $8+3$ znaků). Zde záleží na konkrétní implementaci TEXu, jakým algoritmem vyhledává soubory podle jména balíku (může požadovat jen názvy kratší než daný limit, používat začátek názvu nebo kousek začátku a konce).

Poznámka 2: Pokud chcete používat styl, který není zahrnut ve vaší distribuci (nalezli jste jej třeba v CTANu), musíte jej nejprve nainstalovat. Instalace závisí na typu stylu. Pokud má stažený balíček v sobě soubor s příponou `sty`, zpravidla jej (spolu s ostatními soubory) stačí nakopírovat do adresáře instalace TEXu (někam do adresáře `tex/latex`, větší styly zpravidla mají svůj vlastní adresář, malé styly společně používají něco jako `misc`) a případně – v závislosti na instalaci TEXu – jej nechat zapsat do databáze souborů (např. programem `mktexlsr`). V případě, že stažený balíček obsahuje soubory s příponami `dtx` a `ins`, instaluje se příkazem `latex název.ins`. Některé distribuce TEXu obsahují nástroje přímo určené pro instalaci balíků.

Načítání souboru Do hlavního souboru lze načítat libovolné množství dalších souborů pomocí příkazu `\input{`*jméno*`}`. Je to vhodné např. při psaní knihy, u níž jednotlivé kapitoly vkládáme do samostatných souborů a v hlavním souboru uvedeme jen základní nastavení a příkazy `\input{kap1.tex}` apod. Výhoda spočívá v možnosti zakomentovat jednotlivé řádky s `\input` a překládat jen kapitolu, na které zrovna pracujeme, čímž si ušetříme čas překladu. Načítaný soubor nesmí obsahovat úvodní hlavičku (`\documentclass`, ...) a jeho načítání se ukončí po dosažení příkazu `\endinput` nebo konce souboru.

Jméno souboru lze uvést s příponou i bez ní, ale algoritmus vyhledávání „správného“ souboru může být implementačně závislý. Načítaný soubor zpravidla nemusí být jen v pracovním adresáři, ale i v jiných, které však závisí na implementaci.

Sazba odstavce Chcete-li změnit začátek sazby odstavce, můžete použít příkaz `\everypar={`*příkazy*`}`. Tento příkaz je proveden na začátku každého odstavce. Pokud má být odstavec odsazen, provede se ovšem nejprve odsazení. Tímto způsobem můžete např. všechny odstavce číslovat, nebo např. první písmeno odstavce udělat větší či kaligrafické. Musíte si ovšem pamatovat, že LATEX tento příkaz po každém použití příkazu pro sazbu oddílu jednak předefinuje, jednak se některý příkaz nesnáší s číslováním sekcí. Proto svou definici musíte také vždy znovu opakovat, případně před číslovanou sekcí zrušit.

Jako příklad nechť slouží tyto dvě definice: `\def\cali#1{$\mathcal{#1}$}`, `\everypar ={\cali}`. Jak to funguje, je vidět v tomto odstavci. V dalších už definice byla zrušena. Obdobně fungují příkazy `\everymath` a `\everydisplay`, které se vykonávají ihned po vstupu do matematického prostředí v odstavci a vysazeném.

Kategorie znaků Když TEX čte vstupní soubor, nezajímá se jen o ASCII hodnoty vstupních znaků, ale i o jejich kategorii, což je číslo 0–13 určující, jak bude se znakem dále nakládáno. Pro naše účely se budeme zabývat jen některými kategoriemi. Kategorie 13 je vyhrazena pro tzv. **aktivní** znaky, které mohou vykonávat nějakou funkci samy o sobě, aniž by je předcházelo zpětné lomítko (vhodným příkladem je znak `~`, který pracuje jako nezalomitelná mezera). Kategorie 9 je určena pro znaky, které se budou v textu zcela ignorovat, kategorie 11 je vyhrazena všem písmenům a konečně kategorie 12 je určena pro (téměř) všechny ostatní znaky. Kategorii libovolného znaku lze změnit příkazem `\catcode‘\`*znak*`=`*číslo*, kde *znak* je buď znak sám o sobě (např. `a`), nebo jeho vyjádření pomocí `^^`S, kde ASCII hodnota znaku S je o 64 větší než hodnota *znak*u (používá se pro znaky, které nelze zapsat přímo na klávesnici, např. `^^M` označuje konec řádku, `^^I` je tabulátor).

Všechny příkazy LATEXu mohou ve svém názvu obsahovat pouze znaky s kategorií 11. Této skutečnosti lze využít ke skrytí některých příkazů před neznalým uživatelem – nastavíme kategorii zvoleného znaku na 11, vytvoříme příkaz, který znak obsahuje a znaku pak vrátíme jeho původní kategorii. Uživatel, který o této možnosti neví, pak tento příkaz nemůže použít, což je zvlášť vhodné pro pomocné, „pracovní“ příkazy. V LATEXu je tímto znakem zpravidla `@` a proto jsou nadefinovány pro změnu jeho kategorie příkazy `\makeatletter` a `\makeatother`.

Tabulky Používání prostředí `tabular` není vždy efektivní, mnohdy je vhodnější použít přímo příkazu TEXu ve tvaru `\halign{`*definice*`\cr`*řádek1*`\cr`...*řádekN*`\cr}`. Jednotlivé řádky odpovídají datovým řádkům tabulky, oddělují se pomocí `\cr`, sloupce se od sebe oddělují znakem `&`. Vlastní sazba je určena řádkem *definice*, který má podobnou strukturu jako datové řádky, ale v každém sloupci kromě vhodných příkazů obsahuje jeden znak `#`, který bude nahrazen obsahem sloupce. Například sloupec s definicí `$10^{#}$\quad&#&\quad\bf #` bude sázet data prvního sloupce jako exponenty, druhý sloupec se vysází přímo a třetí sloupec bude sázen tučně. Například s daty `-3&třetí&deset na mínus třetí` se vysází 10^{-3} třetí **deset na mínus třetí**.

Výhodou uvedeného příkazu je možnost definovat opakování – začíná-li některá položka v *definici* symbolem `&` (tedy je buď na začátku `{&` nebo uprostřed `&&`), bude se TEX k této položce opakovaně vracet, jestliže na datovém řádku bude více sloupců než v definici. Nevýhodou tabulky je nutnost dodělat si případné linky sám.

Zarovnání dat Při zobrazení číselných dat v tabulce často chceme, aby čísla byla zarovnána na desetinou čárku a zároveň aby byla ve snadno čitelném tvaru. K tomu můžeme využít skutečnosti, že ve fontech Computer Modern mají všechny číslice stejnou šířku, a to shodnou s mezerou `\enspace`. Pak můžeme například v bloku, který tabulku uzavírá, předefinovat vlnovku pomocí `\let~\enspace` a čísla vlnovkou vhodně zarovnat (například číslo `9,3` zarovnáme na tři místa před čárkou a dvě desetinná pomocí `~~9,3~`). Pokud se v datech vyskytuje znaménko mínus jen zřídka, není někdy vhodné, aby se zahrnovalo do šířky sloupce a zbytečně jej rozšiřovalo. Pomoci si pak můžeme tím, že budeme psát např. `\llap{$-$}9,3`. Uveďme si krátkou ukázku:

1	10,1
2	9,3
3	−10,1

```
{\let~\enspace\begin{tabular}{cc}
1&10,1\\
2&~9,3\\
3&\llap{$-$}10,1\\
\end{tabular}}
```

Barevné tabulky Potřebujete-li nadefinovat barevné pozadí jednotlivých buněk tabulky, je nejjednodušší použít balík `colortab` (vyžaduje ještě balík `pstricks` a `color`). Pak řádky, ve kterých chcete buňky podbarvit, uzavřete mezi příkazy `\LCC` a `\ECC`, přičemž jako první „řádek“ po `\LCC` jsou příkazy, které nastavují zvolené barvy v každém sloupci. Ukázka:

```
\begin{tabular}{ccc}
\LCC
\gray\bf & & \yellow \\
(1,1) & (1,2) & (1,3)\\
(2,1) & (2,2) & (2,3)\\
\ECC
\LCC
\blue & \yellow & \red\\
(3,1) & (3,2) & (3,3)\\
\ECC
\end{tabular}
```

(1,1)	(1,2)	(1,3)
(2,1)	(2,2)	(2,3)
(3,1)	(3,2)	(3,3)

5.1 Sazba stránky

Stránka je sázena pomocí jednotlivých odstavců. TEX používá odstavec jako nejmenší jednotku, tedy vždy načte celý odstavec a až pak se ho pokusí zalomit na řádky, přičemž se snaží, aby jak řádky, tak celé odstavce na stránce vypadaly co nejlépe. Z toho plyne, že *stránka nemusí mít pevnou délku*, a ani vzdálenosti mezi nadpisy a textem apod. nemusí být konstantní.

Při používání různých textových prostředí je třeba dbát na to, aby text mohl být zalomen. Například v prostředí `\verb` se vůbec neprovádí řádkový zlom, takže pokud bude nějaký text dlouhý, bude „přesahovat“ přes okraj ostatního textu – rozdělení zde musíme vyřešit ručně, třeba změnou slovosledu.

Stránka má nastaven vždy nějaký styl stránkování. Tento je možno v preambuli změnit pomocí příkazu `\pagestyle`, jehož parametr může být

- `plain` – číslo stránky je uvedeno dole uprostřed, hlavička je prázdná (implicitně pro styly `article` a `report`);
- `empty` – není hlavička, ani výpis číslování stránek;
- `headings` – číslo stránky a název kapitoly jsou uvedeny v záhlaví, pata je prázdná (implicitně pro třídu `book`);
- `myheadings` – uživatelem definovaná hlavička, obsah hlavičky je nadefinován příkazem `\markright{`*text*`}`, kde *text* je obsah hlavičky u jednostranného dokumentu, pro oboustranný dokument je nutno použít příkaz `\markboth{`*vlevo*`}{`*vpravo*`}`.

Je možno nastavit i zvláštní styl pro jednu stránku příkazem `\thispagestyle`.

Číslování stránek se nastavuje příkazem `\pagenumbering{`*styl*`}`, kde *styl* je jedna z hodnot `roman`, `Roman`, `arabic`, `alph` nebo `Alph`, což dává postupně výpis s římským číslováním malým, velkým, arabské, písmenné malé a velké. Uvedení příkazu automaticky nastaví číslo stránky na 1.

Parametry (velikosti) stránek[29]:

1. `\textwidth` udává šířku textu;
2. `\textheight` udává výšku textového těla (části bez záhlaví a zápatí);
3. `\pagewidth` udává celkovou šířku stránky;
4. `\pageheight` představuje celkovou výšku stránky;
5. `\headsep` udává svislou vzdálenost spodní části záhlaví od horní části těla;
6. `\footskip` udává vzdálenost spodní části zápatí od spodní části těla;
7. `\topmargin` udává vzdálenost horní části záhlaví od okraje tisknutelné plochy;
8. `\headheight` udává výšku záhlaví;
9. `\oddsidemargin` a `\evensidemargin` udávají vzdálenost od vnitřního tiskového okraje k textovému tělu[30];
10. `\marginparsep` udává vzdálenost mezi tělem a blokem okrajových poznámek;
11. `\marginparwidth` udává šířku bloku okrajových poznámek.

Prodloužení jedné stránky je možno dosáhnout příkazem `\enlargethispage{`*vel*`}`, kde *vel* je velikost prodloužení stránky. Varianta s hvězdičkou provede totéž a navíc zmenší všechny svislé mezery na nejmenší možnou míru. Příkazem `\flushbottom` lze dosáhnout naplnění stránek na stejnou výšku (vkládají se svislé mezery), původní chování lze vrátit příkazem `\raggedbottom`. Implicitní způsob naplnění stránek se liší podle třídy dokumentu, například `book` upřednostňuje pevnou délku stránky.

5.2 Dvousloupcová sazba

Pokud použijete balík `twocolumn`, můžete v textu použít dvousloupcovou sazbu (text bude na stránce rozdělen do dvou sloupců, přičemž text „poteče“ prvním dolů, pak přeteče do druhého nahoru a teprve po dosažení spodní hrany přejde na novou stránku). Zapnutí dvousloupcové sazby se provádí příkazem

[29] Každá stránka má implicitně nastaveny okraje 1 in nahoře a vlevo a vymezují maximální tisknutelnou plochu; s tím nic neuděláte, leda změnit v zobrazovači. Některé zde uvedené parametry lze měnit jen v preambuli.

[30] V plainTEXu se k posunu tiskového zrcadla používá příkazů `\hoffset` a `\voffset`.

`\twocolumn[`*text*`]`, kde volitelný parametr bude umístěn nahoře, nad a přes oba sloupce. Vypnutí této sazby se provede příkazem `\onecolumn`. Balík samozřejmě umožňuje nastavení základních parametrů tohoto režimu: míra `\columnsep` udává velikost mezery mezi sloupci a `\columnseprule` udává tloušťku čáry mezi sloupci (standardně 0 pt, tedy nic).

Pokud chceme ve dvousloupcovém režimu umístit plovoucí objekt (`figure` nebo `table`) přes oba sloupce, musíme k němu přidat hvězdičku, jinak bude sázen jen do jednoho sloupce.

5.3 Čítače a délkové proměnné

Čítače jsou celočíselné proměnné, jejichž hodnotu můžeme nastavovat, zvětšovat o jednu nebo k jejich hodnotě něco přičítat. Používají se např. k číslování oddílů textů (čítače `part`, `chapter`, `section`, ..., `subparagraph`), položek v seznamech (`enumi`, `enumii`, `enumiii`, `enumiv`), vzorců (`equation`), obrázků či tabulek (`figure`, `table`), stránek (`page`) a poznámek (`footnote`). Jejich hodnoty můžeme nastavovat na hodnotu *hodn* příkazem `\setcounter{`*čítač*`}{`*hodn*`}`, kde *čítač* je název čítače, přičítat k němu příkazem `\addtocounter{`*čítač*`}{`*hodn*`}` anebo hodnotu zvýšit o jedničku pomocí `\stepcounter{`*čítač*`}`.

Výpis hodnoty čítače je realizován pomocí příkazu `\the`*čítač*, který se přiřadí každému nově vytvořenému *čítač*i. Tento výpis může být realizován v několika podobách: `\arabic{`*čítač*`}` vypisuje arabské číslice, `\roman{`*čítač*`}` malé římské, `\Roman{`*čítač*`}` velké římské, `\alph{`*čítač*`}` malá písmena a `\Alph{`*čítač*`}` velká písmena. Poslední možností je použití `$\fnsymbol{`*čítač*`}$`, která produkuje jeden z desíti poznámkových symbolů. Změny způsobu výpisu čítače se dosáhne např. příkazem `\renewcommand {\thesection}{\Roman{section}}`, který způsobí tisk čísla sekce římskými číslicemi. Pokud potřebujeme zadat obsah čítače tam, kde překladač očekává číselnou hodnotu, použijeme příkaz `\value{`*čítač*`}`. Zadáváme-li délkový parametr, musíme za příkaz ještě uvést příslušnou jednotku.

Vytvoření nového čítače se provede příkazem `\newcounter{`*čítač*`}[`*nadř*`]`, čímž vznikne i výpisový příkaz `\the`*čítač* (*nadř* je nadřízený čítač, jehož použití příkazem `\[ref]stepcounter{`*nadř*`}` způsobí vynulování čítače *čítač*; příkaz s `ref` vytváří hodnotu snímatelnou příkazem `\ref`).

Je-li čítač spřažen s nějakým výčtovým prostředím, používá se pro jeho zavedení a použití příkaz `\usecounter{`*čítač*`}` – příkaz se uvede v druhém parametru prostředí, používá se automaticky.

Délkové proměnné Délkové proměnné (dimenze) jsou takové proměnné, které nemusí být celočíselné a skládají se z vlastní hodnoty a jednotky (buď absolutní, nebo relativní – vztažené k rozměru některého znaku). Mohou být vyjádřeny *délkovými příkazy* (začínají zpětným lomítkem), což jsou příkazy, které přímo představují délku. Pro práci s délkami jsou zavedeny následující příkazy:

- `\newlength{\`*délka*`}` definuje nový délkový příkaz *délka*, nesmí být definován dvakrát;
- `\setlength{\`*délka*`}{`*hodn*`}` nastaví délku *délka* na hodnotu *hodn* (pozor na jednotku);
- `\addtolength{\`*délka*`}{`*hodn*`}` zvýší proměnnou *délka* o hodnotu *hodn*;
- `\settowidth{\`*délka*`}{`*text*`}` nastaví proměnnou *délka* na hodnotu, kterou by po vysázení zabíral *text*, obdobně pracuje `\settoheight` (určení výšky) a `\settodepth` (vzdálenost od účaří ke spodnímu okraji textu);
- `\the`*délka* vypíše hodnotu proměnné *délka* i s jednotkou, implicitně v bodech (`pt`).

U některých délkových příkazů je možno provést nastavení hodnoty přímo příkazem a hodnotou, např. `\itemsep 0pt`. Použitelné absolutní jednotky jsou např. `mm`, `cm`, `in` a `pt`, mezi relativní patří např. výška znaku „x“ (`ex`), šířka znaku „m“ (`em`).

K zajímavým délkovým příkazům patří následující:

- `\parindent` udává velikost odsazení odstavce;
- `\baselineskip` udává vzdálenost účaří dvou po sobě jdoucích řádků a příkaz `\baselinestretch` je číslo, kterým se násobí `\baselineskip` – jeho změnou se dosáhne různé hustoty řádkování;
- `\fboxrule` udává tloušťku čáry používané k orámování příkazem `\fbox`;
- `\arrayrulewidth` udává tloušťku čar v prostředích `array` a `tabular`;
- `\arraycolsep` určuje vzdálenost mezi sloupci[31] v prostředí `array`, obdobně pro prostředí `tabular` se používá `\tabcolsep`.

[31] Velikost mezery mezi řádky v prostředích `array` a `tabular` určuje příkaz `\arraystretch`, který je definován jako `\def\arraystretch{1}`.

5.4 Definice

TEX i LATEX umožňují provádět vlastní definice příkazů (už několikrát jsme toho využili). Obecná syntaxe těchto příkazů je:

- `\newcommand{\`*jméno*`}[`*n*`][`*impl*`]{`*definice*`}`, kde *jméno* je vlastní název příkazu, *definice* obsahuje popis příkazu pomocí již známých, *n* je počet parametrů, které příkaz vyžaduje a *impl* je implicitní hodnota **prvního volitelného** parametru (tedy taková, jaká se dosadí, bude-li volitelný parametr vynechán; pamatujte, že volitelný parametr je vždy v hranatých závorkách), v definici se na parametry odkazujeme pomocí `#`*m*, kde $m \leq n$. Pokud je v definici matematický text, může nastat chyba překladu, je-li tento příkaz vyvolán mimo matematiku – z toho důvodu se *matematika* vkládá do příkazu `\ensuremath{`*matematika*`}`, který sám vyhodnotí a případně zapne matematické prostředí.
- `\renewcommand` se stejnou syntaxí se používá pro předefinování již existujícího příkazu.
- `\providecommand` způsobí definici příkazu jen tehdy, pokud žádný takového jména neexistuje, syntaxe příkazu je stejná jako u předchozích.
- `\def\`*jméno parametry* `{`*definice*`}` je verze definice TEXu s mocnějším významem, protože *parametry* jsou vkládány ve tvaru *sym0*`#1`*sym1*`#2`*sym2*`#3`*sym3*, kde *sym* zastupuje nic nebo nějaký rozlišující znak (či celou skupinu jakýchkoliv znaků, např. `:#1.\en#2!`), který bude při používání definovaného příkazu vyžadován, např. definice příkazu vypisujícího čas s minutami jako horním indexem může být `\def\tim#1:#2#3{\ensuremath{#1^{#2#3}}}` a volá se např. příkazem `\tim 11:05`, jenž dává 11^{05}. Pokud není příslušný *sym* uveden, je jako daný parametr načten jen jeden znak vyjma mezery (pokud je tímto znakem začátek bloku `{`, načte se celý blok). Pokud je *sym* uveden, načítá se do parametru vše až po *sym* (viz `11` v ukázkovém příkladě).
- `\let\`*příkaz1*`=\`*příkaz2* umožňuje příkazy vzájemně přiřazovat.
- `\newfont{\`*příkaz*`}{`*soubor velikost*`}` zavede nový font, který je uložen jako *soubor* (neudávat příponu), velikost představuje klíčové slovo buď `scaled` a příkaz `\magstep` následovaný číslem stupně zvětšení od základní velikosti, nebo `at` a velikost v bodech. Nový font nelze použít v matematickém režimu a ani nejsou při jeho použití měněny automaticky parametry sazby. Pro jiné řezy je nutno nadefinovat nové příkazy (s jinými soubory), nelze použít běžných příkazů jako `\bf` apod. Příkladem může být `\newfont{\cyr}{wncyr10 at 12pt}`, který zavede azbuku ve velikosti 12 bodů (шишли мишли). Pokud chceme vysázet některý speciální znak z nově zavedeného fontu, na nějž se nelze dostat pomocí zavedených příkazů, použijeme příkaz `\symbol{`*pořadí*`}`, kde *pořadí* je ordinální číslo daného znaku, nebo příkaz `\char`*pořadí*.
- `\newenvironment{`*jméno*`}{`*začátek*`}{`*konec*`}` definuje prostředí *jméno* – při použití `\begin{`*jméno*`}` se dosadí do textu *začátek* a obdobně *konec*. Prostředí může mít také parametry, a to i volitelné.

Platnost definic Všechny definice mají platnost jen v bloku, kde byly zadány (podobně jako platnost změny písma). Chceme-li zavést příkaz s platností pro celý text, je nejvhodnějším místem preambule, případně lze místo `\def` použít příkaz `\gdef`, který dá nově definovanému příkazu globální platnost. Navíc platí, že i nově definovaný příkaz se může objevit v pozdější definici jiného nového příkazu.

Jako příklad zjednodušení práce využitím definic lze uvést Laplaceův operátor `\def\Laplace#1{\frac {\partial^2 #1}{\partial x^2}+\frac{\partial^2 #1}{\partial y^2}+\frac{\partial^2 #1} {\partial z^2}}`, s jehož použitím `$$\Delta\psi=\Laplace\psi$$` dává

$$\Delta\psi = \frac{\partial^2\psi}{\partial x^2} + \frac{\partial^2\psi}{\partial y^2} + \frac{\partial^2\psi}{\partial z^2}.$$

Ukázka Uvedeme si jeden mírně složitější příklad, který slouží k časté činnosti – načítání tabulky dat. Nechť máme v externím textovém souboru uvedena data v exponeciálním tvaru (např. `1.153274e-02`), přičemž sloupce jsou odděleny tabulátorem a řádky symbolem nového řádku. Čísla chceme vytisknout v běžném a počeštěném tvaru (s čárkou místo tečky), tedy jako $1{,}153274 \cdot 10^{-02}$.

```
{\catcode`\^^I=4\catcode`\^^M=13
\makeatletter%
\gdef\tabulka{\def\polozka##1.##2e##3X{$##1{,}##2\cdot10^{##3}$}%
```

```
\gdef\loadtable##1{\@@input ##1}%
\makeatother%
\let^^M=\crcr%
\begingroup \catcode`\^^I=4\catcode`\^^M=13%
\halign\bgroup&\polozka##X\hfill\hskip\tabcolsep\cr}}%
\def\endtabulka{\egroup\endgroup}
\tabulka \loadtable{data.txt} \endtabulka
```

Krátké vysvětlení: nejprve jsme nastavili kategorii konce řádku `^^M` na aktivní, tabulátoru `^^I` jsme nastavili kategorii „oddělovače sloupců“ (kterou má běžně pouze znak `&`) a z `@` jsme udělali písmeno, vše jsme provedli v bloku, takže nastavení budou jen lokální. Dále jsme (globálně) nadefinovali příkaz `\polozka`, který převádí exponenciální tvar na něco lidského (bude-li voláno jako `\polozka1.153274e-02X`, vytvoří z něj posloupnost `$1{,}153274\cdot10^{-02}$`) a příkaz `loadtable`, který načte vlastní soubor s daty (využívá příkaz LaTeXu `\@@input`, kvůli kterému jsme museli z `@` udělat písmeno) a zrušíme použití `@` jako písmene. Pak konec řádku ztotožníme s příkazem `\crcr` (který se v tabulce `\halign` chová stejně jako `\cr`, pouze se snesou dva za sebou) a začneme nový blok (příkazy `\begingroup` a `\bgroup` mají smysl začátku bloku, konci bloku odpovídá `\endgroup` a `\egroup`), ve kterém znovu nastavíme kategorie znaku (první nastavení jsme prováděli proto, abychom mohli příkazy vůbec nadefinovat, druhé nastavení se projeví až při načítání dat), ve kterém pomocí `\halign` vytvoříme tabulku s jedním definičním sloupcem, který se bude stále opakovat (díky posloupnosti `\bgroup&`, která odpovídá `{&` – viz `\halign`). Definice sloupce nejprve „volá“ příkaz `\polozka`, kterému předá jako parametr obsah příslušného sloupce (navíc připojí znak `X`, aby mohl příkaz poznat konec položky – tento znak se v načítaném souboru nesmí vyskytovat) a pak ještě vloží malou mezeru, aby sloupce nebyly namačkány na sebe. Nakonec nadefinujeme příkaz pro ukončení tabulky, které jen uzavírá bloky. Všimněte si `%` na konci řádků, které zde musí být, protože máme konec řádku nastavený jako aktivní a jinak ukončený řádek by nám vkládal nežádoucí `\crcr`. Poslední řádek ukazuje použití nadefinovaných příkazů k načtení dat ze souboru `data.txt`. Můžeme rovněž vložit tabulku rovnou do textu (sloupce je nutno oddělit tabulátorem) a psát např.

```
\tabulka
3.00000e-1    3.10000e0    3.14000e3
3.14100e0     3.14150e-2   3.14156e0
\endtabulka
```

Výsledek:

$3{,}00000 \cdot 10^{-1}$	$3{,}10000 \cdot 10^{0}$	$3{,}14000 \cdot 10^{3}$
$3{,}14100 \cdot 10^{0}$	$3{,}14150 \cdot 10^{-2}$	$3{,}14156 \cdot 10^{0}$

Přetváření LaTeXu Systém LaTeX nám umožní vytvořit dokumenty relativně snadno, ale za cenu určité „uniformity“. Mnohdy by nám vyhovovalo přizpůsobit si vzhled či některé parametry s co nejmenším množstvím práce, aniž bychom museli psát vlastní, složité definice. K tomu účelu se vyplatí prostudovat soubor, který definuje třídu dokumentu (má příponu `cls`, např. `article.cls`) nebo definiční soubor `latex.ltx`, vzít z něj nějakou část a vložit ji mírně pozměněnou před `\begin{document}` v našem textu (popř. předefinování použít jen lokálně, uvnitř bloku). Protože tyto interní definice často obsahují znak `@`, musíme tyto příkazy uzavřít mezi `\makeatletter` a `\makeatother`. Dále uvedeme pár příkladů jednoduchých předefinování.

Změnou obsahu příkazů `\labelitemi` až `\labelitemiv` můžeme nastavit značky, které se používají ve výčtovém prostředí pro označení položek první až čtvrté úrovně. Kupříkladu lze jednoduše změnit barvu „puntíku“ pomocí `\renewcommand{\labelitemi}{{\color{`*barva*`}\textbullet}}`. Podobně lze změnit font, kterým je vypisováno prostředí `verbatim`, na skloněné písmo psacího stroje předefinováním `\renewcommand{\verbatim@font}{\normalfont\ttfamily\itshape}`, příklad `\verb#text#` pak dá *`text`*.

Trošku složitějším příkladem může být snaha o vložení poznámky pod čarou, která bude mít možnost nadefinovat si vlastní značku, ale jinak se bude chovat jako ostatní poznámky. Po nahlédnutí do patřičných souborů můžeme vyprodukovat např.

```
\makeatletter
\def\podcarka#1{\insert\footins{\footnotesize\interlinepenalty%
\interfootnotelinepenalty\splittopskip\footnotesep%
\splitmaxdepth \dp\strutbox \floatingpenalty \@MM\hsize\columnwidth \@parboxrestore%
\edef\@thefnmark{\mojefootmark}%
\edef\@currentlabel{}\@makefntext{\rule{\z@}{\footnotesep}\ignorespaces  #1\strut}}}%
\makeatother
\def\znacka#1{$^{#1}$\edef\mojefootmark{$#1$}}
```

V místě poznámky vložíme jednak příkazem `\znacka{`*značka*`}` libovolnou značku•, jednak příkazem `\podcarka{`*poznámka*`}` vložíme vlastní obsah♣.

Poznámka: Používání nových definic je náročná problematika, protože definicemi se vytvářejí makra, která jsou teprve při překladu rozvinována. Je teda třeba vždy dávat pozor na nadbytečné mezery, správné ukončení skupin (pro ukončení definice je vhodné použít příkaz `\relax`) a hlavně na prioritu expandování maker. Z těchto důvodů se dále problematikou definic v úvodním textu zabývat nebudeme.

Začátky a konce Při psaní složitějších dokumentů můžete občas potřebovat, aby se určitý příkaz vykonal před začátkem nebo na konci dokumentu (např. vypsat na konci počet obrázků). Pro tyto účely jsou k dispozici příkazy `\AtBeginDocument{`*příkaz*`}` a `\AtEndDocument{`*příkaz*`}`. Máme-li například nadefinován příkaz `\konec`, který chceme vykonat na konci textu, použijeme `\AtEndDocument{\konec}`.

5.5 pdfTEX

pdfTEX je jedno z rozšíření TEXu, které vytváří místo souborů DVI soubory PDF (resp. může vytvářet i DVI, záleží na nastavení proměnné `\pdfoutput`). Výhody použití tohoto formátu už byly zmíněny u vkládání obrázků. Zde budou uvedeny některé příkazy této implementace TEXu, které spíše doplňují dokument po formální stránce.

Pro definování rozměrů stránky slouží proměnné `\pdfpageheight` a `\pdfpagewidth` se zcela zřejmým významem. Počátek souřadnicové soustavy lze nastavit pomocí proměnných `\pdfvorigin` a `\pdfhorigin`. Každý PDF soubor může obsahovat informace o autorovi dokumentu, jeho názvu a podobně. Tyto informace se do dokumentu vkládají pomocí příkazu `\pdfinfo{/`*položka* `(`*obsah*`)}`, kde *položka* je některé „klíčové“ slovo (`Title`, `Author`, `Keywords`) s příslušným *obsah*em, dvojic *položka*–*obsah* se může v těle příkazu vyskytovat víc (ale jednoznačně).

U některých dokumentů (např. obrazovkových prezentací) je vhodné přinutit zobrazovač otvírat soubor tak, aby se jeho stránky zobrazily celé do vyhraženého okna. Toho můžeme dosáhnout příkazem `\pdfcatalog{`*volby*`} openaction goto num` *číslo*, který způsobí přechod na stránku *číslo*, která je (v příslušném místě) nadefinována příkazem `\pdfdest num` *číslo způsob*, kde *způsob* upřesňuje způsob vyplnění do okna (`fit` – celá stránka, `fith` – na šířku, `fitv` – na výšku). Parametr *volby* určuje, je-li přítomen, jakým způsobem bude dokument zobrazen. Skládá se z části `/PageMode` a vlastní volby: `/FullScreen` (na plnou šířku obrazovky, bez menu apod.), `/UseThumbs` (na boku budouvidět malé náhledy stránek), `/UseOutlines` (zobrazení záložek) nebo `/UseNone`, kdy nebude vidět nic z předchozího.

Máme-li zájem doplnit dokument o malé náhledy jednotlivých stránek, které by mohly být zobrazeny v části okna, lze použít balík `thumbpdf`. Nejprve vytvoříme celý dokument a po jeho dokončení spustíme příkaz `thumbpdf` *jméno.pdf*. Nyní stačí do souboru doplnit použití zmíněného balíku a soubor ještě jednou přeložit.

•Takto bude vypadat vložená poznámka.
♣V první poznámce jsme vložili puntík, tady to je jen ukázka druhé poznámky.

6 METAFONT

METAFONT je nerozlučným souputníkem TEXu, protože bez něj bychom nemohli nic zobrazit. METAFONT totiž slouží k vytváření fontů, tj. grafického ztvárnění jednotlivých znaků. Je to také asi jediná možnost, jak vložit do textu kvalitní obrázek, aniž by byla narušena kompatibilita a přenositelnost souboru, jen místo jednoho zdrojáku přenesete dva (či více, záleží na počtu obrázku). TEX s takto vytvářenými obrázky zachází jako s jednotlivými běžnými znaky.

Další text bude zaměřen jen na vytváření grafů funkcí, nebude se tedy zabývat dalšími funkcemi programu. Celá tato kapitola je jen letmým přehledem a sleduje průběh [3], z něhož je možné načerpat mnohé další informace. Z této literatury byly rovněž převzaty ukázkové příklady.

Vkládání obrázků z METAFONTu Máme-li vytvořen zdrojový soubor METAFONTu (`*.mf`), necháme jej přeložit při vhodném rozlišení a sbalit do souborů `pk`. Zároveň se vytvoří metrický soubor `tfm`. Nyní ve zdrojovém textu TEXu zavedeme nový font se jménem námi vytvořeného souboru (viz příkaz `\newfont`) ve vhodné velikosti. Nechť se například nový font zavádí příkazem `\grafobr` a nechť požadovaný obrázek má ve fontu pořadové číslo 15. Na místo, kde chceme vložit obrázek do textu, vložíme sekvenci

```
\mbox{}
{\grafobr\char15}
```

Příkaz `\mbox{}` byl vložen jen pro zajištění referenčního bodu (může se použít třeba `\centerline`). Ten potřebujeme tehdy, je-li třeba obrázek doplnit popisem (např. os, dělení). Jednotlivé objekty pak vkládáme příkazem `\put`, jehož souřadnice jsou vztaženy právě k referenčnímu bodu (nezapomeňte nastavit `\unitlength`). Rovněž tak můžeme vkládat i doplňující grafiku (použijeme-li prostředí `picture`), je-li to vhodnější než přímo v METAFONTu.

6.1 Základní práce s programem

Postatným rozdílem proti TEXu je naprostý nedostatek zpětných lomítek, protože METAFONT je přímo programovací jazyk, v němž se texty vyskytují jen zřídka. Jednotlivé příkazy naopak musí mít ukončení středníkem. Při vytváření znaků je nutno pamatovat na některá jeho výpočetní omezení, která však zde nejsou uvedena.

Protože se jedná o grafický program, tak se to v něm jen hemží jednotkami. K dispozici jsou tři typy jednotek:

1. absolutní, které jsou nezávislé na rozlišení výstupního zařízení a jejich označení končí `#` (`pt#`, `mm#`, `cm#`, ...), jsou odvozeny od tiskařského bodu;
2. relativní, které jsou závislé na aktuálním rozlišení (DPI), jsou odvozeny od pixelu (mezibodové vzdálenosti rastru zařízení) a mají označení bez `#` (`mm`, `cm`, `pt`);
3. zavedené uživatelem, které jsou odvozeny od předchozích a slouží jen k umožnění jednoduché změny všech rozměrů najednou; např. `u#=1mm#;` zavede novou jednotku `u#`, pro její nastavení musíme použít příkaz `define_pixels(u);`

Pro nastavení rozměrových parametrů dle přání uživatele slouží příkaz `mode_setup`, který nastaví parametry s ohledem na parametry `mode` a `mag` z příkazové řádky. Tímto příkazem většinou každý soubor začíná, jako ukončení se použije příkaz `end.`.

Definice znaku Definice jednotlivého znaku začíná příkazem `beginchar(`*číslo*`,`*šířka*`,`*výška*`,`*hloubka*`);` a končí příkazem `endchar;`. Parametr *číslo* určuje pořadové číslo znaku, *šířka* určuje šířku znaku, *výška* určuje výšku znaku nad účařím a *hloubka* hloubku pod účařím. Tyto rozměry se uloží do proměnných se jmény `w`, `h` a `d`.

Typy proměnných METAFONT samozřejmě umí pracovat s proměnnými, které mohou být následujících typů:

1. **číselné proměnné** slouží k uložení čísel, jejichž absolutní hodnota musí být menší než 4096. Chceme-li používat např. proměnnou `soucet`, můžeme ji deklarovat pomocí `numeric soucet`, což

je ovšem nepovinné. Pokud však chceme používat pole, musíme ho deklarovat pomocí `numeric soucet[]`. K jednotlivým prvkům pole přistupujeme buď vložením indexu do hranatých závorek `soucet[0]`, nebo přímo `soucet0`.

Hodnoty lze přiřazovat buď přímo přiřazením pomocí `:=`, nebo pomocí `=`, což METAFONT chápe jako soustavu lineárních rovnic, kterou vyřeší, až bude potřebovat hodnotu proměnné. K řešení můžeme rovněž použít neznámou, označovanou jako `whatever`[32].

2. **body** jsou chápány jako místo v rovině, komplexní číslo či vektory a jsou zadávány jako uspořádaná dvojice (x,y). Zpravidla jsou body značeny jako `z`*číslo*`=(x`*číslo*`,y`*číslo*) nebo `z[`*číslo*`]=(x`*číslo*`,y`*číslo*`)`, druhý příklad umožňuje využití např. cyklů, jinak je význam stejný. Jednotlivé souřadnice bodu můžeme získat pomocí `xpart z`, `ypart z`. Použijeme-li goniometrický tvar čísla, lze příkazem `angle z` získat argument bodu (úhly zásadně ve stupních), `abs z` absolutní velikost a příkazem `unitvector z` získáme jednotkový vektor téhož směru. Směr můžeme zadat příkazem `dir`*úhel*, číslo v goniometrickém tvaru zadáme např. `z1=2*dir15`. Význačné směry mají vlastní označení `right`, `left`, `up` a `down`.

   Pro označení dělicích poměrů se používá syntaxe α`[z1,z2]`, např. bod ve středu úsečky se zadá jako `z12=0.5*[z1,z2]`.
3. **cesty** jsou spojité čáry vytvořené jedním tahem, jsou to proměnné typu `path` a musejí být deklarovány, např. `path p[];`. Bývají zadány posloupností bodů, jimiž procházejí a způsobem navázání: `--` způsobí spojení přímkou, `..` spojení Bèzierovou křivkou, `{dir30}..` totéž s výchozím úhlem (zde 30 stupňů), `..{dir30}` totéž s koncovým úhlem, je-li koncový bod cesty stejný jako počáteční, vytvoří se uzavřená křivka, je-li místo posledního bodu uvedeno `cycle`, bude křivka uzavřena s hladkým navázáním a bude ji možno vyplnit. Poslední možností je použít `..controls z1 and z2..`, což spojí hraniční body křivkou s kontrolními body `z1` a `z2` (druhý bod lze případně vynechat a s ním i slovo `and`). Jednotlivé cesty je možno skládat. Pokud bezprostředně navazují, použijeme znak `&`, jinak `..` nebo `--` dle typu navázání.
   Standardně jsou předdefinovány cesty pro vykreslení částí jednotkové kružnice (poloměr 0,5 pixel) `quartercircle`, `halfcircle` a `fulllcircle` a jednotkového čtverce `unitsquare`.
4. afinní **transformace** (označení např. `T`) jsou deklarovány klíčovým slovem `transform T`, jejich definice začínají `T=identity` a pokračují vlastní definicí transformace, což může být

   - `shifted (`x`,`y`)` je posunutí o vektor (x,y);
   - `scaled` *vel* je zvětšení o násobek *vel* (stejnolehlost);
   - `xscaled` *vel* zvětšení ve směru osy x, obdobně `yscaled`;
   - `slanted` a je zkosení ve směru osy x, $(x, y) \to (x + ay, y)$;
   - `rotated` θ je otočení kolem počátku o úhel θ;
   - `rotatedarround((`a`,`b`),`θ`)` je otočení okolo libovolného bodu (a,b);
   - `reflectedabout((a,`b`),(`c`,`d`))` je osová souměrnost podél osy zadané body (a,b) a (c,d).

   Jednotlivé transformace je možno libovolně skládat. Transformaci je možno také určit tak, že ke třem bodům, které neleží v jedné přímce, určíme jejich obrazy.
5. **obrázky** lze pro pozdější použití nebo zpracování ukládat do proměnných typu `picture`, do kterých se ukládají v rastrové podobě. Obrázek, který se vykresluje od začátku `beginchar`, je uložen v proměnné s názvem `currentpicture`, prázdný obrázek bývá v `nullpicture`. Chceme-li uchovat současný obrázek, můžeme použít např. `picture obrazek; obrazek:=currentpicture`. V místě použití jej vložíme pomocí `addto currentpicture also obrazek` a případně doplníme nějakou transformaci[33].

Vykreslování cest a bodů Abychom mohli cesty vykreslit, je nejprve nutno zvolit vhodný štětec příkazem `pickup` *druh* `scaled` *zvětšení*, kde *druh* pera může být `pencircle` (kruhové o průměru 1 pixel), `pensquare` (čtvercové s vodorovnou stranou 1 pixel) nebo `penrazor` (vodorovná úsečka nulové tloušťky a délky 1 pixel). Na pera může aplikovat veškeré transformační příkazy.

Vykreslení bodu `z1` se provede příkazem `drawdot z1`. Vykreslení cest se provede příkazem `draw`, za nímž následuje název cesty a jednotlivé transformace na ní prováděné. Lze vykreslit i několik na sebe

[32] Tato proměnná se může v rovnici vyskytovat i několikrát a jednotlivé výskyty jsou na sobě nezávislé.

[33] Rozsah transformací je však omezený, protože obrázek už je rastrovaný – např. rotaci lze provést jen o násobky 90°.

napojených cest, pokud chceme na některou z nich aplikovat transformaci, musíme to občas dobře „ozávorkovat“. Někdy je nutno nadefinovanou cestu `p` vykreslit v opačném směru pomocí `(reverse p)`. Je-li cesta uzavřena (pomocí `cycle`), lze ji vyplnit příkazem `fill`, pokud vyplňujeme obrys uzavřeného písmena (např. `o`), musíme dávat pozor na smysl orientace vnitřní a vnější cesty. Příkazem `filldraw` cestu vyplníme i obtáhneme zároveň.

Prakticky se k vykreslování písmen často používá makra `penpos`*číslo*`(`*šířka*`,`*úhel*`)`, které vytvoří dvojici bodů `z`*číslo*`r`, `z`*číslo*`l`, které jsou vzdáleny o *šířku* a natočeny o *úhel*. Pak se nastaví jejich poloha pomocí `z`*číslo*`=(`*x*`,`*y*`)` a cesta se vykreslí (po nadefinování a určení dvou dvojic bodů) příkazem `filldraw penstroke z1e--z2e` (všimněte si přípony `e`, makro `penstroke` ji postupně nahradí příponami `r` a `l` a cestu z obou stran uzavře, takže ve výsledku dostaneme totéž jako příkazem `filldraw z1r--z2r--z2l--z1l--cycle`).

Protože vytváření fontů i obrázků tímto způsobem je náročné, potřebujeme vědět, kde jsou ve výsledném obrázku umístěny námi definované body. Existuje proto makro `labels`, kterému v závorce předáme čísla všech bodů, které chceme zobrazit (obdobně pracuje makro `penlabels` pro body vytvořené makrem `penpos`). Ve výsledném obrázku se změna po přeložení neprojeví, ale když na vytvořený soubor typu GF pustíme program `gftodvi`, získáme DVI soubor, který obsahuje kresbu znaku, jeho parametry (šířka, výška, hloubka) a polohu zvolených bodů.

Cykly V METAFONTu můžeme použít i příkaz cyklu, jehož obecná syntaxe je `for` *prom* `=`*h1*`,`*h2*`,...:` *příkazy* `endfor;`, kde *h1*, *h2* je výčet jednotlivých hodnot, kterých proměnná *prom* nabývá a *příkazy* tvoří vlastní tělo cyklu.

Druhým typem cyklu je `for` *prom*`=` *dolní* `step` *krok* `until` *horní* `:` *příkazy* `endfor;`, v němž proměnná *prom* probíhá postupně od *dolní* do *horní* hodnoty, vždy se zvětšením *krok*. Pokud je krok $+1$ nebo -1, je možno mezi krajní hodnoty vložit pouze `upto` nebo `downto`.

Posledním typem cyklu je nekonečný cyklus `forever:` *příkazy* `endfor;`, který probíhá bez ukončení. Vyskočit z něj je možno jen příkazem `exitif` *podmínka*, je-li *podmínka* splněna.

6.2 Vykreslování grafů funkcí

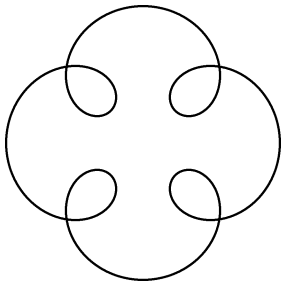
Při vykreslování grafu funkce vycházíme z předpokladu, že při dostatečně husté síti bodů je graf nahraditelný soustavou Bèzierových křivek. Při tom jsme omezeni podmínkou, že cesta nesmí obsahovat více než 300 bodů. Postupujeme tak, že nejprve cestě přiřadíme rovnítkem počáteční bod, pak v cyklu `for` generujeme příslušné dvojice bodů v závorkách a před ně umísťujeme symbol `..` – postupujeme tedy zrovna tak, jako bychom cestu přímo zadávali z již nám známých bodů bez cyklu. Pak příkazem `draw` danou cestu vykreslíme i s příslušnou transformací. Obdobným způsobem lze vykreslovat i parametrické křivky, jen počítáme hodnoty obou souřadnic.

Pro ilustraci způsobu programování uveďme následující příklad: vykreslení epicykloidy, jejíž parametrické rovnice jsou

$$x = R\cos(\Omega t) + r\cos(\omega t), \quad y = R\sin(\Omega t) + r\sin(\omega t),$$

kde v našem případě jsou $R = 12$ mm, $r = 6$ mm, $\omega = 5\Omega = 10$. V ukázce použité funkce `cosd` a `sind` jsou goniometrické funkce s argumentem ve stupních.

```
mode_setup;
u#=1mm#;
define_pixels(u);
beginchar(1,36u#,36u#,0);
pickup pencircle scaled .3u;
draw ((3,0)
for i=1 upto 180:
 ..(2cosd(2i)+cosd(10i),2sind(2i)
 +sind(10i))
endfor)
scaled 6u shifted (w/2,h/2)
endchar;
end.
```



Způsob vykreslování grafů funkcí dvou proměnných je možno nalézt v již citovaném textu [3], kde je uveden jako tzv. makro.

METAPOST

Program METAFONT umožňuje kvalitní kreslení, ale bohužel při využití ke tvorbě obrázků má dvě podstatné nevýhody: nepodporuje práci s barvou a jeho výstup je vždy jen rastrovaný[34]. Vznikl proto nástroj, který se snaží být po stránce zápisu „programu“ co nejvíc slučitelný s METAFONTem, ale bez uvedených nevýhod – METAPOST. Jak již název napovídá, výsledkem jeho práce je soubor v (zapouzdřeném) PostScriptu, který je vektorový. METAPOST se však nesnaží nahradit funkci METAFONTu coby generátoru znaků, ale jen obrázků. Vzhledem k tomu je rozšířen o metody, které umožní vložit do obrázku jakýkoliv text vysázený TEXem.

Pro začátek vystačíme s tím, že METAPOSTu nabídneme zdrojový soubor pro METAFONT, u kterého změníme příponu na `mp`, a odstraníme z něj příkazy, které jsou smysluplné jen pro tvorbu znaků. Zároveň nahradíme příkazy `beginchar` příkazy `beginfig` (tento příkaz má pouze jeden parametr, a to číslo obrázku) a `endchar` nahradíme `endfig`. Zpracujeme-li tento soubor (spustíme program `mpost soubor.mp`), objeví se řada souborů `soubor.`N , kde N je číslo, odpovídající jednotlivým znakům (dle údaje v `beginfig`). Program tedy zpracovává postupně jednotlivé znaky a vytváří z nich samostatné soubory typu Eps (které si lze prohlédnout např. programem GhostView).

Z rozšiřujících funkcí METAPOSTu uvedeme jen následující (ostatní najdete v manuálu):

- deklarace „barevné“ proměnné se provádí jako `color` *jméno*, přiřazení *jméno*`:=(`r`,`g`,`b`)` vloží do proměnné barvu s barevnými složkami r (červená), g (zelená) a b (modrá), což jsou čísla v intervalu $\langle 0, 1\rangle$.
- vykreslení cesty v dané barvě se provede připojením klíčového slova `withcolor`, následovaného zvolenou barvou, do parametrů příkazu `draw`.
- vykreslovaná cesta nemusí být kreslena plnou čarou, ale v případě použití klíčového slova `dashed` může mít vzor definovaný pomocí `dashpattern(on 2 off 2)`. Každé `on` nebo `off` musí být následováno délkou příslušného úseku a úseky se mohou libovolně opakovat; existují předdefinované vzory `evenly` (stejně dlouhé úseky kreslené a nekreslené) a `withdots` (tečkovaná čára). Vzory lze také zvětšovat, například lze psát `draw fullcircle scaled 20 dashed (evenly scaled .5)`.
- každou cestu můžeme vykreslit s šipkou na jednom či obou koncích pomocí příkazů `drawarrow` a `drawdblarrow`.
- jakýkoliv text, který má být vysázen TEXem, se uzavírá do svorek `btex` a `etex` (např. vykreslení integrálu v poloze (0,0) můžeme docílit vložením `draw thelabel(btex $\int$ etex), (0,0)`). Tyto příkazy se však zpracovávají zvláštním způsobem, a proto v nich nemůžeme používat programátorské konstrukce METAPOSTu. Dále nám nebude fungovat čeština, protože se překlad provádí (anglickým) plainTEXem. Chceme-li používat počeštěný LATEX, musíme na začátek souboru napsat

  ```
  verbatimtex
  %&cslatex
  \documentclass{article}
  \usepackage{czech}
  \begin{document}
  etex
  ```

  a na konec `verbatimtex \end{document} etex`.

  Chceme-li napsat jen jednoduchý *text*, můžeme napsat jen `draw thelabel("`*text*`",` *pozice*`)`. Příkaz `thelabel.`*xxx* způsobí zarovnání textu podle hodnoty *xxx*: `top` nahoru, `bot` dolů, `lft` vlevo, `rt` vpravo. „Dvojitého“ zarovnání dosáhneme hodnotami `ulft`, `urt`, `llft` a `lrt`, kde `l` značí dolní a `u` horní roh. Příkaz `draw thelabel` umožňuje při vykreslování aplikovat transformace, změnu barvy apod. Pokud nic takového nepotřebujeme, můžeme použít jednoduchý příkaz `label("`*text*`",`*pozice*`)`. V obou variantách lze přesně určit použitý font pomocí „přídavku“ `infont "`*jméno*`"`, např. `label( "Řešeto" infont "csr10", (0,0))`. Tento příkaz je použitelný jen tehdy, neobsahuje-li *text* mezeru (na jejím místě je ve fontech polské „škrtátko“ nebo „vanička“).
- na začátku dokumentu můžeme nastavit proměnnou `prologues` (má-li hodnotu 1, bude do výstupu vložena i hlavička a soubor lze použít i samostatně, jinak je použitelný jen vnořený do jiného dokumentu), dále lze použít proměnné `defaultscale` (určuje násobící faktor fontů použitých v textech)

[34] To souvisí s jeho určením k tvorbě písma, které bývá černobílé a rastrované.

či `defaultfont` (určuje font, kterým bude sázen text, název fontu musí být uzavřen do uvozovek, vlastní font se do obrázku nevkládá, použijeme-li nějaký TEXovský font, zobrazí se text správně až po vnoření do dokumentu, přímo lze použít jen standardní POSTSCRIPTové fonty, např. Helvetica, Times).

- souřadnice bodů jsou implicitně v POSTSCRIPTových bodech o velikosti $\frac{1}{72}$ in. Pokud chceme jinou jednotku, musíme ji zadat jako `pt` ($= \frac{1}{72{,}27}$ in), `mm`, `cm` apod.

Výsledek METAPOSTu lze do dokumentu vložit způsobem běžným pro vkládání externích obrázků, např. použitím `\includegraphics` (před vložením musí být zdrojový soubor přeložen METAPOSTem). Problematičtější je vložení obrázku do pdfTEXu, který vložení PostScriptu neumožňuje. Zde si vypomůžeme balíčkem `mfpic`, který nám nadefinuje příkaz `\convertMPtoPDF{`*jméno.číslo*`}{`*faktorx*`}{`*faktory*`}` pro načtení obrázku v souboru *jméno.číslo*, jehož velikost může být případně násobena příslušnými faktory. V hlavičce dokumentu musí být uveden příkaz `\usemetapost`.

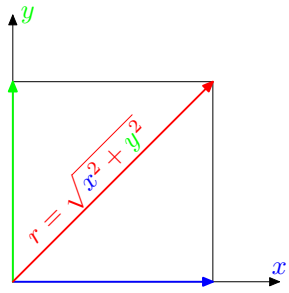
Poznámka 1: Pokud chceme obrázek použít jinde než v TEXu, může se nám někdy hodit spíše ve formátu PDF. Nejprve musíme přeložit původní zdrojový soubor METAPOSTem a pak použít příkaz `mptopdf`, který soubor převede a případně do něj vloží použité fonty.

Poznámka 2: Pokud používáme program `mptopdf`, můžeme v programu používat rozšíření z balíku METAFUN, které nabízí možnost průhledností, přechodů a pokročilejší práci s textem.

Poznámka 3: Možností METAPOSTu můžeme využívat k „ozdobení“ textu v podobě různých rámečků, výplní, a to i takových, které se přizpůsobují textu (např. orámování jednotlivých slov, podbarvení odstavců). Pro tyto účely je však nejlepší místo LATEXu použít formát **ConTEXT**, který je určen pro použití s pdfTEXem.

Poznámka 4: METAPOST lze snadno použít k tvorbě grafů ze zadaných bodů. Je vhodné k tomu použít standardní balík `mpgraph`, který do souboru načteme pomocí `input mpgraph`.

Jako malou ukázku práce v METAPOSTu si uveďme následující vykreslení souřadných os a vyznačení velikosti r. Všimněte si použití příkazů `\phantom` a `\vphantom`, které jsou nutné proto, že METAPOST neumožňuje používat barvu v příkazech `label`. Příkaz `drawoptions` si zapamatuje žádané volby, aby je nebylo nutné opakovat u každého příkazu `draw`.




```
beginfig(1);
pair wid; wid=(100,0);
pickup pencircle scaled .4pt;
z0=(5,5); z1=z0+wid; z2=z0+(wid rotated 90);
z1'=z0+.75wid; z2'=z0+(.75wid rotated 90);
z3=z1'+z2'-z0; z4=.5[z0,z3];
drawarrow z0--z1; drawarrow z0--z2;
draw z1'--z3--z2';
pickup pencircle scaled .8pt;
drawarrow z0--z1' withcolor blue;
drawarrow z0--z2' withcolor green;
draw thelabel.top(btex $x$ etex, z1) withcolor blue;
draw thelabel.rt(btex $y$ etex, z2) withcolor green;
drawarrow z0--z3 withcolor red;
drawoptions(rotated angle(z3-z0) shifted (z4+(0,10pt) rotated 90) withcolor red);
draw thelabel(btex $r=\sqrt{\phantom{x}^2+\phantom{y}^2}$ etex,(0,0)) withcolor red;
draw thelabel(btex $\phantom{r=\sqrt{\vphantom{x^2+y^2}}}x\vphantom{\sqrt{y^2}}
   \phantom{^2+y^2}$ etex,(0,0)) withcolor blue;
draw thelabel(btex $\phantom{r=\sqrt{x^2+\vphantom{y^2}}}y\phantom{^2}$ etex,(0,0))
   withcolor green;
endfig;
end.
```

7 Poznámky

Tabulka fontu Chcete-li se podívat, jak vypadají znaky určitého fontu, můžete si jej zobrazit pomocí souboru `testfont.tex`. Stačí v příkazovém řádku spustit `tex testfont` a program se zeptá na jméno fontu (např. `csr10`). Pak vám nabídne několik možností, co udělat (jejich seznam obdržíte zadáním `\help`, můžete ale napsat jakýkoliv příkaz TEXu). Nejzajímavější z nich jsou vytvoření tabulky rozmístění znaků pomocí `\table` a totéž s vypsáním krátkého textu pomocí `\sample`. Poté můžete buď činnost ukončit (`\end`) nebo přejít na jiný font (`\init`). Výpis fontu bude v souboru `testfont.dvi`.

Kreslení vektorových obrázků Do výstupu z TEXu můžete vložit jakýkoliv obrázek, který se vám podaří dostat do formátu EPS. Takový obrázek můžete získat z libovolného programu, který umí tisknout, pokud si nainstalujete POSTSCRIPTovou tiskárnu[35] (výstup případně musíte upravit, aby odpovídal specifikacím EPS). Mnoho komerčních i nekomerčních programů přímo umožňuje exportovat obrázky do formátu EPS nebo PDF. Z volně dostupných lze zmínit např. programy XFIG a IPE, vhodné pro tvorbu „technických“ obrázků, které nabízejí rozšířené možnosti integrace s TEXem.

Pokud potřebujete vygenerovat EPS obrázek z aplikace v systému MS WINDOWS, ze které lze obrázek zkopírovat přes schránku, můžete vyzkoušet program wmf2eps s grafickým rozhraním, který dává vcelku slušné výsledky.

Pro vykreslení grafů funkcí nebo grafu dat lze použít např. GNUPLOT nebo statistický balík R; existují i další programy, které slouží k „programování“ vektorové grafiky, jedním z nich je například PYX, ve kterém obrázek nakreslíte v PYTHONU. Na ukázku uveďme tuto posloupnost příkazů v GNUPLOTu, která vytvoří soubor `sinusovka.mp` s obrázkem sinusovky:

```
set term mp color latex "csr10"
set output "sinusovka.mp"
plot [x=0:4*pi] sin(x)
quit
```

Na prvním řádku se volí výstup ve formátu METAPOSTu s podporou barev, výstupem vhodným pro LATEX a s českým fontem pro popisky. Parametr `latex` můžete klidně vynechat, pokud jej použijete, musíte spouštět METAPOST příkazem `mpost --tex=latex sinusovka` (nebo něčím podobným v závislosti na vaší instalaci). Graf sinusovky bude doplněn popisem os a orámováním, pro změnu těchto hodnot nastudujte nápovědu programu (`help set`). Uvedený způsob je vhodný tehdy, jestliže chcete výstup z GNUPLOTU dále upravovat, případně zahrnout do jiného souboru. Pokud chcete jen vložit obrázek, můžete přímo vytvořit EPS soubor tím, že první řádek nahradíte `set term postscript eps color` a na druhém změníte příponu.

Náhled znaků Pokud se chcete podívat, jak je nějaký znak navržen, případně vytvořit nový znak, není vhodné zobrazovat jej v běžné velikosti. Výhodnější je získat jej zvětšený spolu s polohami všech bodů, které byly k jeho vykreslení použity. K tomu účelu musíte nejprve METAFONTový soubor přeložit, např. pomocí `mf csr10`, čímž vznikne soubor se jménem `csr10.2602gf` (přípona může být v některých systémech zkrácena). Tento soubor pak pomocí příkazu `gftodvi csr10.2602gf` převedete na soubor `csr10.dvi`, který můžete zobrazit libovolným prohlížečem souborů DVI (prohlížeč bude potřebovat speciální font se jménem `gray`). Na každé stránce bude zobrazen jeden znak, včetně jeho názvu, a polohy všech bodů, které si návrhář fontu přál zobrazit. Pokud je font neskloněný, bude zobrazena i mřížka odpovídající šířkovým a výškovým parametrům znaku.

Převod do jiných formátů Někdy je třeba převést text do jiných formátů, například HTML nebo prostého textu. Dokonalý automatizovaný převod není bohužel možný, ale alespoň velká část práce se zautomatizovat dá. Existují dva základní přístupy – buď převod pracuje se zdrojovým souborem LATEXu, nebo pracuje až s výstupem v DVI nebo PDF. První přístup využívá například program `latex2html` pro převod do formátu HTML, který podporuje standardní LATEX i některé běžné balíky; lze jej však rozšířit o podporu pro další balíky. Druhý přístup používá např. program `catdvi`, který převádí výstup do prostého textu (i UNICODE), volitelně se snaží zachovat základní rozrvžení textu. Při druhém přístupu můžete použít i mezipřevod do formátu PDF, ze kterého získáte text pomocí programu `pdftotext`.

[35] V UNIXových systémech je to standardní volba.

Pokud se jedná o komplikovanější text, který si realizujete sami, lze použít postup, ve kterém použijete oba kroky: používané příkazy LaTeXu si předefinujete tak, aby do výsledného Dvi vkládaly vhodné značky příkazem `\special`, pak soubor přeložíte a výsledné Dvi zpracujete svým vlastním programem (formát Dvi má velmi jednoduchou strukturu, napsat jednoduchý převodník je otázkou pár hodin) a tím dokončíte převod do cílového formátu.

Správa textu Při psaní delšího textu není vhodné vpisovat vše do jednoho souboru, který se pak stane neúměrně dlouhý a bude se dlouho překládat. Lepší je rozdělit text do několika souborů podle logických částí (např. kapitol) a vytvořit jeden hlavní soubor, který bude obsahovat LaTeXové náležitosti (`\begin{document}`, `\end{document}` apod.) a místo vlastního textu bude pouze načítat další soubory pomocí `\input{`*soubor*`}`. Pokud pracujete jen na jedné kapitole, můžete zakomentovat všechny ostatní příkazy `\input` a překládat text rychleji.

Potřebujete-li srovnat dvě verze textu, lze použít program `diff`, který soubory porovnává po řádcích. Jeho výstup však není vhodný v případech, kdy se v textu mění jen jednotlivá slova (např. při korektuře). Pro tyto účely lze použít program s názvem `texdiff`, který se snaží porovnávat text po jednotlivých slovech a měl by být schopen označit části vložené a odstraněné. Bohužel, uvedený program někdy nepracuje vůbec.

Pokud píšete delší text, u kterého předpokládáte časté změny, je vhodné používat nějaký systém pro správu verzí, který vám umožní se kdykoliv vrátit k předchozí verzi textu nebo srovnat dvě verze textu. Nejjednodušším systémem je program Rcs. Pokud pracujete např. se souborem `prace.tex`, vytvoříte pomocí `rcs prace.tex` archív s názvem `prace.tex,v`. Do něho se budou postupně ukládat všechny vámi potvrzené změny dokumentu. Při prvním spuštění `rcs` zadáte nejaký popis projektu a pak se uloží první verze. Další verze vložíte pomocí příkazu `ci prace.tex` – změny se uloží do archívu (spolu s popisem změn, které zadáte po spuštění programu) a původní soubor zmizí. Budete-li chtít na souboru opět pracovat, musíte ho „vytáhnout“ z archívu pomocí `co -l prace.tex`. Při práci s archívem můžete dále používat programy `rcsdiff` pro srovnání poslední verze v archívu se současnou verzí souboru a `rlog` pro výpis všech změn.

Pdfmark V některých případech požadujeme na výstupu formát Pdf, ale musíme vytvořit nejdříve PostScript. Chceme-li i v tomto případě používat klikací odkazy, záložky a podobné prvky, můžeme použít třeba balík `hyperref` s volbou `pdfmark`. Protože balík `hyperref` však kdeco předefinuje, může být někdy výhodnější vkládat odkazy v „pdfmark“ přímo pomocí příkazu `\special`. Informace o dokumentu můžete vložit na začátku textu pomocí příkazu

```
\special{ps:[ /Title (Muj text) /Author (Ja) /Subject (My) /Keywords (klic)
              /DOCINFO pdfmark}
```

Záložku, směřující na „současnou“ stránku, můžete vytvořit pomocí
`\special{ps:[ /Page \the\count0 /Title (Stranka) /View [/XYZ 0 421 null] /OUT pdfmark}`, kde `421` představuje výškovou pozici, na kterou se nastaví pohled po kliknutí na záložku. Nulová hodnota odpovídá spodnímu okraji stránky, maximální hodnota pro stránku formátu A4 je 842. Text `Stranka` se objeví v okně záložek.

Prezentace LaTeX lze použít i k tvorbě prezentací na obrazovce. Jako nejvhodnější se jeví použití formátu Pdf, protože vnitřně podporuje barvu, rastrové obrázky, odkazy a taky pár animovaných přechodů mezi stránkami. Je možné si prezentaci vytvořit s použitím vlastních maker (v podstatě jde jen o jiný formát stránky), ale je lepší použít specializovaných balíčků. Jedna z možností je použít balík `pdfscreen`. Při zavádění balíku pomocí `\usepackage` můžete v nepovinných parametrech specifikovat, zda má prezentace obsahovat tlačítkové menu (`panelleft`, `panelright`, `nopanel`), jaké barevné schéma se bude používat a jestli má být v panelu obsah (`paneltoc`). Po zavedení pak můžete specifikovat obrázek, který bude sloužit jako emblém v panelu (`\emblema{`*soubor*`}`), velikost stránky (`\screensize{`*šířka*`}{`*výška*`}`) a velikost okrajů (`\margins{`*vlevo*`}{`*vpravo*`}{`*nahoře*`}{`*dole*`}`). Příkazem `\overlay{`*pozadí*`}` lze určit pozadí stránky. Jednotlivé stránky lze pak zadávat buď pomocí přechodů na nové stránky, nebo je vkládat do prostředí `slide`.

Další možností je použití novějšího balíku `beamer`, který nabízí volbu témat, možnosti postupného přibývání položek a podporu pro vektorovou grafiku (pomocí dalšího balíku).

Přepisky obrázků Mnoho programů, které generují Eps soubory, nemá typografickou kvalitu na úrovni TEXu. Jimi generované popisky jsou zpravidla jen vedle sebe naskládaná písmenka, o možnosti zapsání rovnic už vůbec nemluvě. Pro tyto účely lze použít balík `psfrag`, který umí nahradit popisky v Eps souboru libovolným LATEXovým blokem. Nahrazování funguje tak, že místo celého popisku v programu, který Eps generuje, použijeme pouze jednoslovný „tag“ (bez akcentovaných znaků). Před načtením obrázku pomocí `\includegraphics` definujeme nahrazení pomocí příkazu `\psfrag{`*tag*`}{`*LATEXovský blok*`}`. Při načtení obrázku pak *LATEXovský blok* nahradí všechny výskyty textu *tag*. Například `\psfrag{x**2}{$x^2$}` nahradí symbolické vyjádření druhé mocniny. Příkaz `\psfrag` můžeme použít několikrát pro různé *tag*y, definice nahrazení platí vždy v daném bloku. Příkaz má další nepovinné parametry, které mohou určovat zarovnání, zvětšení a rotaci nového popisku.

Manipulace se stránkami V některých případech je třeba provádět operace, které jsou snáze realizovatelné na úrovni stránek výstupního formátu než přímo v TEXu. Mezi ně patří kupříkladu umístění více logických stránek na jeden list papíru, jiné seřazení stránek apod. Nejvíce možností poskytuje převod souboru do formátu PostScript. Pak můžeme používat následující programy:

- `psnup` slouží k umístění několika stránek na jeden list papíru. Například `psnup -2 soubor.ps` umístí na list dvě stránky, které zároveň vhodně zmenší;
- `psbook` slouží k seřazení stránek tak, aby se dal dokument po oboustranném tisku složit do tvaru brožury;
- `psselect` slouží k výběru stránek, které se překopírují do výstupního souboru;
- `psmerge` slouží ke spojení několika PostScriptových souborů, které ovšem musí být vytvořeny stejnou aplikací (většinou se ovšem spojení nepodaří).

Všechny tyto programy jsou pouze nádstavbou nad obecným programem `pstops`, který nabízí širší možnosti poskládání stránek. Existují podobné programy, které pracují přímo s formátem Dvi, ale ty v distribucích běžně nebývají zahrnuty.

Manipulace se stránkami lze provádět i v TEXu, budeme-li stránky používat jako obrázky. Existuje např. balík `pdfpages`, který umožňuje do pdfTEXu vložit pomocí příkazu `\includepdf[`*volby*`]{`*soubor*`}` libovolné stránky z Pdf dokumentu. Pomocí voleb je možné určit, které stránky vložit, kolik stránek vložit na jeden list, zvětšení či otočení stránek.

V současných distribucích bývá k dispozici program `texexec`, který slouží k „řízenému“ spouštění TEXu, METAPOSTu a dalších pomocných programů. Lze jej využít taky ke „zpracování“ Pdf dokumentů, například ke spojení několika souborů do jednoho nebo ke tvorbě brožury. Spojení dvou souborů se provede pomocí `texexec --pdfarrange s1.pdf s2.pdf`, výsledný soubor bude mít jméno `texexec.pdf`.

Bibtex Při psaní textů je často třeba citovat jiné práce, přičemž způsob, jakým se citace provádí, se může velmi lišit. Proto se místo přímého kopírování už zformátovaných citací a jejich následného přeformátování používají citační programy, které spravují databázi citovatelných zdrojů a odkazy aktuálně citovaných prací zformátují podle našich požadavků. V případě TEXu se používá program s názvem BibTEX. Program se používá následujícím způsobem. V LATEXovském souboru (např. `main.tex`) se odkazujeme na zdroje příkazem `\cite`, ale příslušnou položku `\bibitem` v textu neuvádíme. Místo toho si udržujeme jednu (nebo několik) databázi v souboru s příponou `.bib`, ve které jsou uvedeny všechny použitelné zdroje. Každý zdroj je uvozen typem práce a jedním klíčem, který pak použijeme v parametru příkazu `\cite`. Na konec textu pak uvedeme příkazy `\bibliography{`*soubor*`}` a `\bibliographystyle{`*styl*`}`, kde *soubor* je jméno databáze (musí mít příponu `.bib`, kterou zde můžeme vynechat) a *styl* určuje způsob formátování citací. Pak soubor přeložíme `cslatex main`, vygenerujeme citace pomocí `bibtex main` a znovu (případně dvakrát) spustíme překlad `cslatex main`. Na výstupu pak obdržíme v textu odkazy na příslušné práce a na konci dokumentu bude seznam všech citovaných zdrojů. Pokud používáme více souborů s databázemi, můžeme je uvést v parametru příkazu `\bibliography` oddělené čárkami.

Stylové soubory BibTEXu určují jednak způsob odkazu (čísla v hranatých závorkách, první autor doplněný rokem publikace apod.), jednak vlastní formátování (jestli bude zobrazen název práce, jestli bude tištěn italikou, jestli bude tučně číslo časopisu, pořadí těchto údajů, formátování jmen autorů apod.). K dispozici je několik standardních stylů[36]:

[36] Můžete si samozřejmě napsat vlastní styl. Styl BibTEXu je v podstatě program, který určuje, jak se mají záznamy z databáze přepsat do souboru s příponou `.bbl`. Psaní těchto programů není jednoduché, ale dají se snadno zvládnout úpravy stávajících stylů, jen je nutno uvést, že styly používají „zásobníkovou“ notaci.

- `unsrt` uvádí citace v pořadí, v jakém jsou citovány v textu a citace označuje číslem;
- `plain` citace abecedně setřídí a označuje je číslem;
- `alpha` citace abecedně setřídí, ale označuje je zkratkou vytvořenou z prvních písmen příjmení prvních tří autorů a z roku vydání;
- `abbrv` pracuje jako předchozí, ale místo plných jmen autorů uvádí jejich první písmena.

Pro práci s citacemi ve tvaru „Autor, rok“ je vhodné použít např. balík `natbib`, který rozšiřuje možnosti příkazu `\cite` a zároveň definuje nové příkazy pro citaci v závorkách či bez závorek, citaci roku publikace či jmen autorů apod. Umožňuje také definovat *aliasy* pro význačné citace.

Záznamy do databáze lze přidávat buď ručně, nebo pomocí celé řady programů (z volně šířených např. PYBLIOGRAPHER a JABREF). Každý záznam začíná na novém řádku „zavináčem“, za kterým je uveden typ zdroje (`ARTICLE`, `BOOK`, `PROCEEDINGS`, `MISC` atd.) a celý záznam je uzavřen ve složených závorkách. Jednotlivé údaje se oddělují čárkou, prvním údajem musí být jedinečný klíč, který záznam identifikuje, další údaje pak mají zpravidla tvar `nazev = {obsah}`, přičemž každý název by se měl vyskytnout jen jednou. Existují předdefinované názvy (`author`, `title`, `year`, `note`), z nichž některé jsou povinné[37]; záznam samozřejmě může obsahovat i další, nestandardní údaje. V údajích je možno používat jakékoliv příkazy LaTeXu, speciálně se zpracovávají jen některé údaje:

- údaje jako `author`, `year` se používají ke třídění, měly by proto obsahovat příkazů co nejméně;
- `author` může obsahovat příkazy pro diakritiku, jednotliví autoři se oddělují pomocí ␣`and`␣, jméno se uvádí za příjmením, odděluje se čárkou a uvádí se do složených závorek;
- v údaji `title` se ignoruje velikost písmen vyjma těch, které jsou uzavřeny do dalších složených závorek.

Typický záznam může vypadat takto:

```
@ARTICLE{BCh+05,
  Author         = {Brbla, Pavel and Chytr\'a, Hor\'akyn\v{e}},
  Title          = {V {K}ocourkov\v e},
  Journal        = {Pohádky \& příběhy},
  Volume         = {25},
  Number         = {1},
  Pages          = {10-18},
  Note           = {Obzvlášt pěkná pohádka},
  Year           = 2005,
}
```

citace ve stylu `plain` bude vypadat následovně:

> [1] Pavel Brbla and Horákyně Chytrá. V Kocourkově. *Pohádky & příběhy*, 25(1):10–18, 2005. Obzvlášt pěkná pohádka.

Jistě jste si všimli, že uvedený styl není počeštěn a mezi autory vkládá anglické „and“.

Program BIBTEX lze samozřejmě používat pro správu i jiných údajů, např. adres, telefonních čísel apod. I pro tyto účely jsou nadefinované vhodné styly.

[37] Různé typy mohou mít různé povinné údaje.

A když LaTeX nechci? Pokud vás LaTeX nijak nezaujal, můžete vyzkoušet některé jiné „programovací“ systémy pro sazbu textu, například:

1. **groff** je jednoduchý sázecí systém, který se používá např. pro formátování Unixových manuálových stránek. Nabízí lámání textu do řádků, použití různých písem, tvorbu tabulek či rovnic (přes externí programy) a poskytuje výstup ve tvaru prostého textu nebo PostScriptu.

2. **lout** [5] je relativně nový systém s širokou škálou možností sazby, který je kromě sazby hladkého textu schopen sazby rovnic, tabulek i grafiky. Podporuje výstup v PostScriptu, Pdf, Dvi i prostém textu.

3. **QuickScript** [6] je systém, který k sazbě využívá interpret PostScriptu. Do psaného textu se vkládají speciální značky, které se interpretují až při tisku (nebo zobrazení pomocí Ghostscriptu). Systém nabízí možnost sazby tabulek, záhlaví, seznamů apod. Pokud je mi známo, neumí přímo vysázet rovnice, ale umožňuje vložení libovolného PostScriptového kódu. Primární výstup je pouze do PostScriptu.

Závěr

Nyní se dostáváme k závěru textu. Tento letmý přehled rozhodně neměl nahradit kvalitnější učebnice, které jsou uvedeny dále, ale jen umožnit základní přehled o způsobu a obtížnosti práce s LaTeXem. Kdyby měl poskytovat úplný zdroj informací, musel by také obsahovat spoustu vzorových příkladů a také rejstřík příkazů. Rovněž jeho struktura by musela odpovídat učebnici – měly by být napsány oddělené celky, aby bylo možno se vrátit k ucelené části. Zde jsem spíše předpokládal, že si přečtete celý text najednou, a tak jsou některé věci uvedeny pouze tam, kde se logicky hodily (to tedy znamená, že hledání, kde zrovna to či ono bylo, může zabrat pěknou spoustu času).

Hodně zdaru při další práci s TeXem vám přeje

Autor

Za vzornou spolupráci autor děkuje tučňáku Tuxovi.

Literatura

[1] Rybička, J.: **LaTeX pro začátečníky**, *Konvoj 1995*

[2] Olšák, P.: **Typografický systém TeX**, *1995*

[3] Šedivý, P.; Brož, M.; Gřondilová, J.; Píše, M.; Houfek, K.: **Kreslíme Metafontem**, *1997* (http://www.cstug.cz/kreslime/)

[4] Olšák, P.: **TeXbook naruby**, *Konvoj 2001* (pro zájemce o označení „pokročilý uživatel“, dostupná na ftp://math.feld.cvut.cz/pub/olsak/tbn/)

[5] Lout, `http://snark.niif.spb.su/~uwe/lout/lout.html`

[6] QuickScript, `ftp://ftp.adfa.edu.au/pub/postscript/Qs_README.html`

Spoustu dalších informací vám nabízí Československé sdružení uživatelů TeXu CSTUG na svých internetových stránkách `http://www.cstug.cz`. Z anglických zdrojů lze doporučit stránky mezinárodní organizace `http://www.tug.org`.