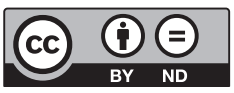
LaTeX pro pragmatiky

Pavel Satrapa

Dokument vznikl za podpory Technické univerzity v Liberci a sdružení CESNET. Jeho aktuální verzi najdete na adrese
http://www.nti.tul.cz/~satrapa/docs/latex/

Verze: 1.0 první veřejná verze, červen 2011
1.1 drobné doplňky a rozšíření, září 2011



Dokument je volně šiřitelný pod licencí Creative Commons BY-ND. Můžete jej šířit a používat pro komerční i nekomerční účely, musí však být uveden autor a dokument nelze měnit.

Sázeno XeLaTeXem písmy Comenia Serif Pro a Vida Mono 32 Pro

Předmluva

Cílem tohoto textu je pomoci vám zorientovat se v typografickém systému LaTeX. Snažil jsem se, aby byl pokud možno krátký, začínal naprostými základy, ale zároveň alespoň naznačil některé složitější konstrukce.

V žádném případě jej neberte jako kompletní či referenční příručku. Řadu věcí jsem zjednodušil, některé zcela vynechal. Nejedná se také o typografickou učebnici. Píši, jak sazební prvky technicky realizovat pomocí LaTeXu, nikoli kdy a proč to dělat.

Jak název napovídá, snažil jsem se o pragmatický přístup. Existující volně šiřitelné texty na podobné téma se zpravidla omezují na holý LaTeX. Jenže už poměrně nezkušený uživatel může začít pokukovat po sazebních prvcích, které vyžadují rozšiřující balíky (vkládání obrázků, živé odkazy v PDF, vícesloupcová sazba, barvy a podobně). Chtěl jsem popsat alespoň základy, jak toho dosáhnout.

Dá se čekat, že vám text postupně přestane stačit a začnete se rozhlížet po další literatuře, která by vám objasnila podrobnosti a dovolila nahlédnout pod kapotu. Standardní dokumentací programu TeX je kniha [Knu86], v češtině určitě stojí za přečtení [Olš01]. Pokud se LaTeXu týče, lze čerpat přímo od pramene z [Lam94]. Pro přizpůsobování a rozšiřování jeho chování je nedocenitelná kniha [MiG04] – kdybyste si měli koupit jedinou publikaci o LaTeXu, doporučil bych tuhle. Jejími souputnicemi jsou [GMR07], která je orientována na práci s grafikou, a [GRG99] pro on-line publikování. V češtině je nejoblíbenější kniha [Ryb02]. A pokud se samotné typografie týče, určitě byste neměli minout publikace [Pec11] a [Što08].

V textu hojně cituji různé `příkazy` a další prvky zdrojového textu. Jsou sázeny neproporcionálním písmem a barevně odlišeny. Rozsáhlejší ukázky kódu jsou navíc ohraničeny. Poměrně časté jsou také příklady, kdy

na pravé straně najdete zdrojový text a na levé výsledek jeho zpracování.

```
na pravé straně najdete
zdrojový text a~na levé
výsledek jeho zpracování.
```

Pavel Satrapa
Liberec, červen 2011

Obsah

1 Úvod, základní pojmy a principy

Koncem 70. let byl americký profesor informatiky Donald E. Knuth natolik nespokojen se sazbou jedné ze svých knih, že se rozhodl napsat typografický program, který bude sázet pořádně, a to včetně složitých matematických vzorců. Vznikl TEX (čtěte „tech“, kořeny názvu pocházejí z řečtiny).

Patří do rodiny tak zvaných značkovacích jazyků (markup languages) a dal by se zjednodušeně charakterizovat jako programovací jazyk pro sazbu textů. Jeho základním vstupem je textový soubor, který obsahuje jak sázený dokument, tak příkazy ovlivňující sazbu. Určité znaky mají přiřazen speciální význam a jejich prostřednictvím jsou v textu odlišeny řídicí konstrukce. Typickým příkladem je zpětné lomítko, jímž začínají příkazy.

Původní Knuthovou ideou bylo, aby TEX fungoval identicky na všech možných platformách. Zpracováním vstupního dokumentu proto vznikl soubor typu DVI (DeVice Independent), který obsahoval jeho vysázenou podobu, nikoli však konkrétní tvary jednotlivých znaků. DVI je abstraktní a obsahuje jen informace typu „na souřadnicích (x, y) se nachází znak Q o velikosti v sázený písmem p“. K zobrazení či vytištění DVI potřebujete specializovaný program, který disponuje použitými písmy. Tyto programy byly původně jedinou součástí závislou na konkrétním výstupním zařízení.

Takové uspořádání není příliš praktické. Dnes proto uživatelé obvykle dávají přednost implementacím TEXu, které na výstupu generují soubor ve formátu PDF, jako je pdfTEX či XƎTEX.

TEX definuje přibližně 300 vestavěných (tzv. primitivních) příkazů, které jsou ovšem dost jednoduché a kdybychom měli sázet dokumenty jen pomocí nich, udřeli bychom se. Naštěstí jsou k dispozici nástroje, jak si z existujících příkazů stavět nové – tak zvaná makra. Sám Knuth vytvořil sadu sofistikovanějších maker pod názvem PlainTEX. Ve své knize [Knu86], základní příručce pro TEX, popisuje jak primitivní příkazy, tak makra PlainTEXu.

Leslie A. Lamport vytvořil pro TEX jinou sadu maker a pojmenoval ji LATEX. Snažil se v ní vyjít vstříc běžným potřebám při sazbě dokumentů, proto zařadil příkazy pro členění textu do kapitol, generování obsahu či vkládání obrázků a tabulek. Považuji LATEX za jednodušší a použitelnější pro každodenní účely, proto se tento text věnuje jemu.

TEX je tedy typografický program a LATEX knihovna maker pro něj, která rozšiřuje jeho jazyk a definuje konstrukce pro prvky obvyklé při sazbě dokumentů. Různých sad maker pro TEX existuje celá řada, nicméně PlainTEX a LATEX jsou jednoznačně nejrozšířenější a nejvýznamnější. Z pohledu vývoje se chovají dost nezvykle.

PlainTEX je velmi konzervativní. V roce 1989 Donald E. Knuth prohlásil, že jej nebude nijak rozšiřovat ani měnit, pouze opravovat chyby. Číslo verze konverguje k π a s každou opravou přibere jedno desetinné místo (aktuálně 3.14159265).

LATEX je pro změnu předmětem nekonečného vývoje. Svého času se masově prosadila verze 2.09, zatížená řadou nedostatků. V roce 1990 byl proto zahájen vývoj verze 3 a jako dočasný mezistupeň k ní vytvořen LATEX 2ε. Jak už tak bývá, dočasnost se stává poněkud trvalou a po dvaceti letech je LATEX 3 stále v nedohlednu. Tento text proto vychází z verze LATEX 2ε, která je současným standardem.

2 Instalace

Existuje celá řada implementací a distribucí TEXu pro různé operační systémy. De facto standardem se stala distribuce TEX Live, kterou vyvíjí sdružení uživatelů TEXu – TEX Users Group, TUG.

Pokud používáte Linux, s vysokou pravděpodobností bude TEX a LATEX vycházející z TEX Live dostupný v repozitáři, instalujte jej standardní cestou. Pokud byste nechtěli (občas repozitáře obsahují starší verze) nebo máte jiný operační systém, stáhněte si ze stránky

http://www.tug.org/texlive/acquire-netinstall.html

instalační program *install-tl*, rozbalte jej a spusťte. Verze pro Windows obsahuje obvyklý sled dialogů, ovšem se zcela minimalistickými možnostmi – můžete nastavit instalační adresář a formát papíru, jinak snad nic. Nainstaluje se kompletní distribuce. Stejného chování dosáhnete v Linuxu, pokud spustíte *install-tl -gui wizard*. Pokročilejší možnosti nabídne *install-tl -gui perltk*, kdy si můžete vybírat instalované součásti.

Po dokončení instalace byste měli mít k dispozici příkazy *tex*, *latex* a další, jimiž se program spouští. Případně musíte vhodně upravit proměnnou prostředí *PATH*, aby se našly, nebo pro ně při instalaci nechat vy-

tvořit odkazy ve správných místech (plnohodnotná instalace na to má volbu).

3 První dokument a jeho překlad

Dokument pro LaTeX má pevnou kostru. Vypadá takto:

```
\documentclass[a4paper,12pt]{article}
\begin{document}
Zde je text dokumentu.
\end{document}
```

Úvodní příkaz `\documentclass` deklaruje třídu dokumentu. Zatím berte jako dogma, že jím dokument musí začínat, později se k němu vrátíme. Část mezi `\documentclass` a `\begin{document}` se nazývá preambule. Slouží pro nastavení různých parametrů, definice příkazů a podobně. Nesmí generovat žádný viditelný výstup. Vlastní sázený text je uzavřen mezi `\begin{document}` a `\end{document}`. Na jeho uspořádání příliš nezáleží. Než se TeX pustí do sazby, vstupní soubor si předžvýká podle následujících pravidel:

1. konec řádku nahradí mezerou
2. libovolně dlouhou posloupnost mezer nahradí jednou mezerou
3. jedinou výjimkou je prázdný řádek, který odděluje odstavce

Když se vrátím k výše uvedenému příkladu, text dokumentu ve tvaru

```
Zde          je
      text
                  dokumentu.
```

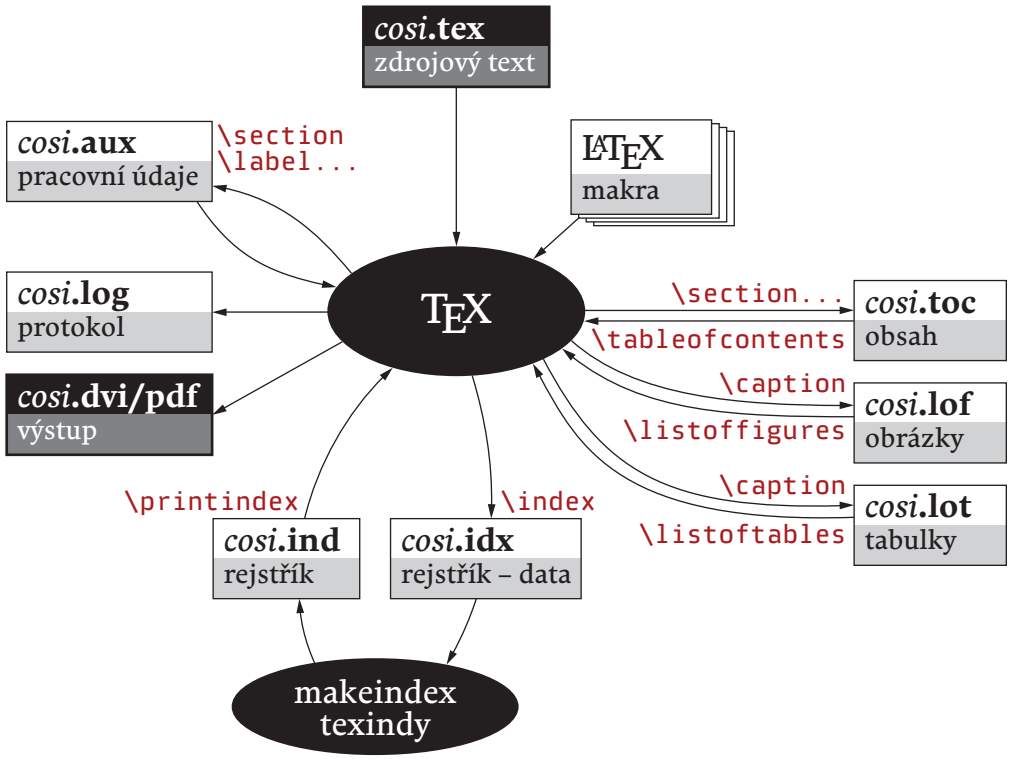
by vedl ke stejnému výsledku jako text původní. V obou případech bude výsledkem sazby jedna stránka obsahující nápis

> Zde je text dokumentu.

Překlad zdrojového textu zajistí příkaz *latex*, kterému jako parametr předáte jméno souboru se zdrojovým textem. Pokud bude výše uvedený zdrojový text uložen v souboru *priklad.tex* (standardní příponou zdrojových textů pro TEX je *.tex*), zajistí jeho překlad

latex priklad

Příponu uvádět nemusíte, program si ji domyslí. Výstupem budou tři soubory: *priklad.dvi* obsahuje vysázenou verzi textu, *priklad.log* protokol o překladu a *priklad.aux* interní informace pro TEX. U složitějších dokumentů se může objevit ještě několik dalších souborů. Celý kolotoč znázorňuje schéma na obrázku 1, jeho části postupně vysvětlím.



Obrázek 1: LaTeX a soubory kolem něj

Chete-li výstup ve formátu PDF, použijte příkaz *pdflatex* nebo *xelatex*. Překlad dokumentu prvním z nich zajistí příkaz

pdflatex priklad

Práce s LaTeXem znamená vytvořit v libovolném textovém editoru zdrojový text dokumentu, přeložit jej, prohlédnout si výsledek, provést úpravy ve zdrojovém textu, znovu přeložit, prohlédnout a tak dále, dokud není dokument hotov.

Vzhledem k tomu, že vstupním souborem je obyčejný text, můžete pro jeho editaci použít libovolný ASCII editor, třeba i *Poznámkový blok* z Windows. Díky hojnosti příkazů v textu si však příliš radosti neužijete. Pokud už máte svůj oblíbený sofistikovanější editor[1], pravděpodobně podporu pro LaTeX už obsahuje nebo se do něj dá snadno doplnit. Například pro můj oblíbený *Vim* existuje *Vim-LaTeX*.

Druhou variantou je sáhnout po editoru určeném speciálně pro LaTeX, jako jsou například *LyX*, který se snaží o pseudoWYSIWYG přístup, *TeXworks*, *Texmaker* a další. Tyto nástroje jsou samozřejmě optimalizovány pro TeX a LaTeX, ovšem na druhé straně v podstatě nepoužitelné pro cokoli jiného. Volba je na vás.

Překlad nemusí pokaždé dopadnout dobře. Dojde-li k chybě, budete vystaveni nepříliš přívětivému způsobu, kterým LaTeX oznamuje problémy. Udělal jsem úmyslně překlep v závěrečném příkazu výše uvedeného souboru a odměnou mi byla následující lamentace:

```
! LaTeX Error: \begin{document} ended by \end{doument}.

See the LaTeX manual or LaTeX Companion for explanation.
Type  H <return>  for immediate help.
 ...

l.4 \end{doument}
```

První řádek informuje, k čemu vlastně došlo. Zde se liší jméno v příkazech `\begin` a `\end`. Na konci najdete číslo a text řádku, v němž došlo k problému (`l.4` znamená 4. řádek). Pokud se chyba nachází kdesi uvnitř, je řádek rozdělen v místě jejího výskytu na dvě části.

Následně TeX přejde do interaktivního režimu, zobrazí výzvu `?` a čeká na vaše instrukce, co má dělat dál. Obvyklou reakcí je stisknout Enter, čímž program vyzvete, aby se s chybou vypořádal jak nejlépe umí a pokračoval v překladu. Některé chyby ovšem mají tendenci vyvolávat další

[1] Tím v žádném případě nemyslím *Microsoft Word* nebo *OpenOffice.org Writer*, řeč je o čistých ASCII editorech.

a další, takže opakované „odklepávání“ nevede k cíli. Pak můžete reagovat dvojím způsobem: `x` sdělí TEXu, že má zanechat marného snažení a svou činnost okamžitě ukončit (eXit). `q` nařídí, aby si přestal stěžovat a dotáhl překlad, jak nejlépe umí (Quiet).

Možnosti jsou širší, ale v běžné praxi si obvykle vystačíte s těmi popsanými. Na webu můžete najít podrobnější popis chyb LATEXu.

Ne každý problém způsobí zastavení překladu. Ty méně závažné (příliš řídká nebo hustá sazba, chybějící písmo, špatný odkaz a podobně) vám program pouze ohlásí a pokračuje dál. Věnujte proto pozornost výstupu z překladu, případně si prostudujte jeho podrobnější verzi v souboru s příponou *.log*.

4 Podpora češtiny

Podpora češtiny v sázených textech zahrnuje dva aspekty. Na nižší úrovni je třeba přimět TEX, aby akceptoval znaky s diakritickými znaménky. Pokud jste v příkladech výše do textu dokumentu zařadili české znaky, budou ve výstupu pravděpodobně chybět, protože standardní LATEX podporuje pouze anglickou abecedu.

Je třeba programu sdělit, že vstupní dokument používá určité kódování. To zajistí balík `inputenc`, jehož parametrem je konkrétní použité kódování. V našich podmínkách připadá nejspíše v úvahu `utf8` (obvyklé v moderních operačních systémech), `cp1250` (kódová stránka 1250 ve Windows) nebo `latin2` (ISO 8859-2 používané dříve v Linuxu a spol.).

Balíky se do dokumentu vkládají v preambuli, typicky hned za úvodním `\documentclass`. Pokud je text v kódování UTF-8, vypadalo by jeho zahájení následovně:

```
\documentclass[a4paper,12pt]{article}
\usepackage[utf8]{inputenc}
\begin{document}
...
```

Druhá část podpory češtiny se týká logicky vyšších vrstev. Je záhodno, aby strojově generované texty byly v češtině (například „Obsah“, nikoli „Table of contents“), slova se dělila podle českých vzorů a obecně se při sazbě dodržovaly konvence české typografické tradice.

Přizpůsobení LaTeXu různým jazykům má na starosti balík `babel`, kterému parametrem určíte cílový jazyk – v našem případě `czech`. Úplné zahájení českého dokumentu, jež nastaví vše potřebné, tedy vypadá

```
\documentclass[a4paper,12pt]{article}
\usepackage[utf8]{inputenc}
\usepackage[czech]{babel}
\begin{document}
...
```

Alternativou je použití CSTeXu (resp. CSLaTeXu), což je adaptace TeXu pro češtinu a slovenštinu vyvinutá Petrem Olšákem. V současné době je podpora češtiny v balíku `babel` už na takové úrovni, že považuji za vhodnější použít tento univerzálně dostupný balík.

5 Příkazy

Klíčovou roli mají příkazy, jimiž řídíte celou sazbu. Příkaz vždy začíná zpětným lomítkem. Podle toho, co následuje, se dělí do dvou kategorií:

Řídicí slova jsou tvořena libovolně dlouhou posloupností písmen (anglické abecedy). Jako příklad může posloužit příkaz `\TeX`, který vysází logo TeX. Řídicí slovo končí prvním nepísmenným znakem. Pokud je tímto znakem mezera, bude řídicím slovem „sežrána“ a ze vstupu zmizí (jiné znaky zůstanou zachovány).

To se někdy hodí, jindy překáží, je zkrátka třeba si zvyknout. Chcete-li zachovat mezeru za řídicím slovem, máte několik možností: můžete za ním použít řídicí mezeru (mezeru předcházenou zpětným lomítkem), nebo příkaz uzavřít do složených závorek (jež vymezují skupinu, více zanedlouho), nebo za něj vložit prázdnou skupinu:

TeX TeX TeX vysází logo TeXu.

```
\TeX\ {\TeX} \TeX{}
vysází logo \TeX u.
```

Řídicí znaky obsahují jediný nepísmenný znak. Například `\#`, kterým se sází znak „#“, je řídicí znak. Taktéž řídicí mezera zmíněná v předchozím bodu je řídicím znakem.

Z hlediska účinku lze charakterizovat tři typy příkazů:

Vkládací příkazy vloží do místa svého výskytu příslušnou typografickou konstrukci, jako například `\TeX` logo nebo `\today` dnešní datum.

Přepínače změní určitý parametr sazby. Změna je trvalá a platí až do doby, kdy příslušný parametr změníte jiným přepínacím příkazem nebo skončí skupina, v níž ke změně došlo. Typickým příkladem je `\itshape`, *který přepne až do odvolání na kurzívu*.

Příkazy s parametrem vytvoří určitou konstrukci kolem svého parametru. Ta se typicky týká jeho podoby při sazbě, může ale být i dost složitá, například příkaz `\section{Příkazy}`, kterým jsem zahajoval tuto sekci, vysází nadpis, vloží jej do obsahu a podobně.

Parametr se uvádí bezprostředně za jménem příkazu a uzavírá se do složených závorek. Pokud má příkaz parametrů více, má každý své složené závorky. Některé příkazy mají i nepovinný parametr, bývá uveden jako první a zapisuje se v hranatých závorkách. Například úvodní příkaz `\documentclass`

```
\documentclass[a4paper,12pt]{article}
```

má jeden nepovinný parametr s hodnotou `a4paper,12pt` a jeden povinný, jehož hodnotou je `article`.

6 Znaky a jiné základní konstrukce

O běžné znaky v textu se nemusíte nijak zvlášť starat. Pokud jste správně deklarovali vstupní kódování použitím balíku `inputenc`, měly by se bez problémů sázet. Potřebujete-li vložit znak ležící mimo náš obvyklý sortiment – třeba v cizím jméně – můžete využít pestrou nabídku akcentovacích příkazů, které shrnuje tabulka 1. Z podobného soudku jsou i různé národní znaky v tabulce 2.

`\'{o}`	ó	`\`{o}`	ò	`\"{o}`	ö	`\H{o}`	ő
`\v{e}`	ě	`\u{o}`	ŏ	`\^{o}`	ô	`\~{o}`	õ
`\c{c}`	ç	`\k{a}`	ą	`\={o}`	ō	`\b{o}`	o̲
`\.{c}`	ċ	`\d{c}`	ọ	`\r{a}`	å	`\t{oo}`	o͡o

Tabulka 1: Akcentové příkazy LaTeXu

`\oe`	œ	`\OE`	Œ	`\ae`	æ	`\AE`	Æ
`\o`	ø	`\O`	Ø	`\aa`	å	`\AA`	Å
`\l`	ł	`\L`	Ł	`\ss`	ß		

Tabulka 2: Mezinárodní znaky

Tím jsme se zvolna dostali do oblasti různých speciálních symbolů a nezvyklých znaků, jako jsou §, © a další. Ty nejčastější najdete v tabulce 3. Nemá smysl snažit se o jejich kompletní výčet, najdete jej třeba v úplném přehledu symbolů LaTeXu, který zahrnuje kromě základu i myriády symbolů v různých nestandardních písmech a rozšiřujících balících. Hledat v nich není příliš zábavné. Vaší pozornosti proto doporučuji online vyhledávač znaků, kterému nakreslíte přibližnou podobu symbolu a on se vám odvděčí příslušným příkazem.

©	`\copyright`	§	`\S`
®	`\textregistered`	†	`\dag`
…	`\ldots`	‡	`\ddag`

Tabulka 3: Vybrané symboly

Ne pro každý symbol existuje příkaz, všechny však lze vysázet, pokud znáte kód příslušného znaku. Použijte `\char`*číslo*, kde *číslo* je kód znaku zadaný v desítkové, šestnáctkové (pak mu předřaďte znak `"`) nebo osmičkové (předřaďte `'`) soustavě. Všechny implementace akceptují osmibitové znaky (kód nanejvýš 255), implementace podporující UTF-8 i šestnáctibitové (kód do 65 535):

¶¶¶ `\char182 \char"B6 \char'266`

Samostatnou kategorii představují znaky se speciálním významem. Ty jsou sice na klávesnici dostupné, ale byla jim přiřazena určitá funkce,

například \ zahajuje příkazy[2]. Pokud chcete takový znak vysázet, musíte použít speciální příkaz. Často mu jednoduše předřadíte zpětné lomítko, ale ne pro všechny to platí. Jejich přehled uvádí tabulka 4. Obsahuje jednotlivé znaky, jejich speciální význam a způsob, jak je vysázet.

\	zahajuje příkazy	`\textbackslash`
{ }	vymezují skupiny	`\{` a `\}`
&	odděluje sloupce tabulky	`\&`
%	zahajuje komentář	`\%`
~	nezlomitelná mezera	`\textasciitilde`
$	zahajuje/ukončuje matematický režim	`\$`
#	odkaz na parametr makra	`\#`
^	horní index	`\textasciicircum`
_	dolní index	`\_`

Tabulka 4: Speciální znaky

Ve zdrojovém textu lze používat komentáře. Jsou zahájeny znakem % a TEX při zpracování ignoruje vše od znaku % až po konec řádku (včetně). Není k dispozici žádná konstrukce, která by vyznačila začátek a konec komentáře[3] – u dlouhých komentářů je třeba zahájit každý jejich řádek znakem %. Kromě vkládání interních poznámek se komentáře někdy využívají k vynechání konce řádků. V některých konstrukcích by mezera vytvořená koncem řádku vadila, proto jsou řádky v nich ukončeny znakem %, díky němuž bude znak konce řádku ignorován a začátek následujícího naváže bezprostředně na předchozí.

Příjemnou kategorií speciálních znaků jsou ligatury neboli slitky. Jedná se o dvojice znaků, které by bezprostředně za sebou nevypadaly dobře a proto se nahrazují hezčím dvojznakem. Typickým příkladem jsou „fi“ a „fl“, které lépe vypadají jako „fi“ a „fl“. Dobrou zprávou je, že o slitky se nemusíte nijak starat. Kdykoli TEX narazí ve vstupu na příslušnou kombinaci znaků, nahradí je slitkem.

Interně se slitky používají i v některých dalších případech, například k rozlišení pomlček. Typografové totiž znají tři: krátký silný *spojovník* (-) pro dělení slov, zvratné -li a složená slova, *pomlčku* (–) ve větách a intervalech a *dlouhou pomlčku* (—), kterou používají ve větách ame-

[2]Toto přiřazení lze změnit a zahajovat příkazy místo \ třeba znakem !. Snad netřeba dodávat, že není rozumné to dělat bez velmi vážných důvodů.

[3]Rozšiřující balík `verbatim` definuje prostředí `comment`, které takové možnosti poskytuje.

ričtí typografové. Ve zdrojovém kódu se zapisují jako jeden, dva nebo tři po sobě jdoucí znaky `-`, které se při sazbě slijí do příslušné pomlčky.

Běžná pomlčka ve větě – je poloviční proti—dlouhé.

```
Běžná pomlčka ve větě~-- je
poloviční proti---dlouhé.
```

Potíž je s uvozovkami. V $\mathcal{CS}$TEXu pro ně existoval příkaz `\uv`, který svůj argument uzavřel do uvozovek. V moderních verzích české podpory ale chybí. Existují pro ně sice příkazy – `\quotedblbase` pro zahajovací a `\textquotedblleft` pro ukončovací – ale komu by se chtělo s nimi psát? Můžete si definovat příkazy pro svůj editor, kterým je vložíte, nebo si sami doplnit definici příkazu `\uv` – později se k němu vrátím. Při vstupu v kódování UTF-8 lze také rovnou zadávat příslušné znaky U+201E a U+201C.

„Těžká práce“ s českými „uvozovkami“.

```
\quotedblbase Těžká
práce\textquotedblleft\
s~českými „uvozovkami“.
```

Angličané to mají jednodušší, protože pro jejich verze uvozovek jsou definovány slitky: ``` `` ``` zahájí a `''` ukončí “uvozený” text.

7 Skupiny a prostředí

Koncept skupin zavádí už samotný TEX. Skupina začíná ve zdrojovém kódu znakem `{` a končí znakem `}`. Umožňují například seskupit řetězec znaků a prohlásit je za parametr určitého příkazu. Hlavní silou skupin ovšem je, že v okamžiku svého začátku uloží aktuální parametry sazby a při ukončení je opět obnoví. Jinými slovy, veškeré změny (velikost a typ písma, nastavení různých vlastností, ale i definice příkazů) provedené uvnitř přestanou v okamžiku ukončení skupiny existovat:

dvě *velká kurzívní* slova

```
dvě {\Large\itshape velká
kurzívní} slova
```

LATEX zavedl prostředí jako nadstavbu skupin. Také prostředí v okamžiku ukončení obnoví stav sázecího mechanismu a ukončí platnost všech

změn provedených uvnitř. Kromě toho ovšem nějakým způsobem ovlivní sazbu textu uvnitř.

Každé prostředí má své jméno a jeho vymezení zajistí dvojice konstrukcí `\begin{`*jméno*`}` a `\end{`*jméno*`}`. Může mít i parametry, které ovlivňují jeho chování. Ty se nacházejí v obvyklém tvaru bezprostředně za zahájením, například `\begin{tabular}{ll}`.

Prostředí do sebe lze vnořovat. Vnoření samozřejmě musí být korektní – to, které bylo zahájeno jako poslední, musí skončit nejdříve, teprve pak lze uzavřít prostředí, které je obklopuje. LaTeX kontroluje jména v příkazech `\begin` a `\end` a pokud narazí na nesoulad, ohlásí chybu.

Řada předdefinovaných prostředí je přímo součástí LaTeXu – už jste se třeba setkali s prostředím `document`, které obaluje vlastní text dokumentu. Zde se podívejme na některá jednodušší.

TeX a jeho parta implicitně sází text do bloku, tedy s oběma okraji zarovnanými. Pro změnu máte k dispozici prostředí `flushleft` (na prapor vlevo), `center` (centrovat) a `flushright` (na prapor vpravo):

doleva

na střed

doprava

```
\begin{flushleft}
doleva
\end{flushleft}

\begin{center}
na střed
\end{center}

\begin{flushright}
doprava
\end{flushright}
```

K dispozici jsou i tři zúžená prostředí. Pro rozsáhlejší citace z jiných zdrojů (například výňatek ze zákona či literárního díla) slouží `quote` nebo `quotation`. Obě mají rozšířené okraje a lehce se liší vzhledem (první neodsazuje počáteční řádek odstavce, druhé ano). Oficiálně je `quote` určeno pro kratší citáty, zatímco `quotation` pro delší, které obsahují více odstavců.

Toto je normální text sázený na celou šířku řádku.

> Text v prostředí quote je z obou stran zúžený.

```
Toto je normální text sázený
na celou šířku řádku.

\begin{quote}
Text v~prostředí quote je
z~obou stran zúžený.
\end{quote}
```

Pokud byste inklinovali k sazbě poezie, sáhněte po prostředí `verse`. Jednotlivé verše ukončujte příkazem `\\`, mezi strofami vynechte řádek jako mezi odstavci.

Zcela mimo rámec běžného chování leží prostředí `verbatim`. Přejde na neproporcionální písmo a svůj obsah vysází tak jak je – neinterpretuje příkazy, zachovává mezery i konce řádků. Hodí se pro ukázky různých konfiguračních souborů či příkazy vlastního LaTeXu (v této roli je používám i zde). Někdo jím sází zdrojové texty programů, pro tento účel ovšem dává podstatně hezčí výsledky rozšiřující balík `listings`.

Běžný text.

```
Příkaz \textbf{nebude}
vykonán. Jen    se opíše.
```

```
Běžný text.
\begin{verbatim}
Příkaz \textbf{nebude}
vykonán. Jen    se opíše.
\end{verbatim}
```

V rámci řádku lze podobného efektu dosáhnout příkazem `\verb`. Jeho parametr se neuzavírá do složených závorek, místo toho lze použít libovolný znak, který následuje bezprostředně za `\verb`. Druhý výskyt stejného znaku pak ukončí parametr. Existuje také prostředí `verbatim*` a příkaz `\verb*`, v nichž se navíc zvýrazňují mezery.

Do␣␣␣řádku{ se vloží část kódu příkazem \verb.

```
\verb*#Do   řádku{# se vloží část
kódu příkazem \verb:\verb:.
```

8 Seznamy

LaTeX zahrnuje tři druhy seznamů: s odrážkami, číslované a s nadpisy. Seznam s odrážkami vytvoříte pomocí prostředí `itemize`. Uvnitř pak každou položku zahajte příkazem `\item`:

Řádek před seznamem.

- První položka.
- Druhou uděláme delší, aby bylo vidět, jak ji LaTeX bude formátovat.

  Může obsahovat několik odstavců.
- Třetí položka.

```
Řádek před seznamem.
\begin{itemize}
\item První položka.
\item Druhou uděláme delší,
aby bylo vidět, jak ji
\LaTeX\ bude formátovat.

Může obsahovat několik
odstavců.
\item Třetí položka.
\end{itemize}
```

Číslovaný seznam se vytváří stejně, jen místo `itemize` použijete prostředí `enumerate`. Seznamy pochopitelně lze vnořovat, styl vyznačování jednotlivých položek se automaticky změní. Vysázet test je proto zcela snadné:

1. První otázka.
   (a) buď
   (b) nebo
   (c) anebo jinak
2. Druhá otázka.
   (a) třeba
   (b) nebo ne
   (c) a co tohle?

```
\begin{enumerate}
\item První otázka.
  \begin{enumerate}
  \item buď
  \item nebo
  \item anebo jinak
  \end{enumerate}

\item Druhá otázka.
  \begin{enumerate}
  \item třeba
  \item nebo ne
  \item a co tohle?
  \end{enumerate}
\end{enumerate}
```

Pro seznam s nadpisy slouží prostředí `description`. Každá položka má určitý zvýrazněný pojem, k němuž se váže vysvětlující text – například slovníková hesla s výkladem. Nadpis položky vložíte jako nepovinný parametr (tedy v hranatých závorkách) příkazu `\item`:

TEX je typografický program, jehož autorem je Donald E. Knuth.

LATEX je nadstavba TEXu, vytvořil ji Leslie A. Lamport. Snažil se o vyšší úroveň abstrakce a podporu běžně používaných konstrukcí.

```
\begin{description}
\item[\TeX] je typografický
program, jehož autorem je
Donald~E. Knuth.

\item[\LaTeX] je nadstavba
\TeX u, vytvořil ji
Leslie~A. Lamport. Snažil
se o vyšší úroveň
abstrakce a podporu běžně
používaných konstrukcí.
\end{description}
```

Také v ostatních typech seznamů můžete příkazu `\item` předat nepovinný argument. V tom případě jím bude nahrazen implicitně generovaný symbol nebo číslo.

9 Mezery a rozměry

Mezery mají v sazbě velmi důležitou roli, společně s použitými znaky tvoří její jádro. Běžnou textovou mezeru vložíte do zdrojového textu obyčejnou mezerou (nebo koncem řádku či skupinou mezer, viz pravidla zpracování zdrojového textu výše). Mezery mají určitou standardní velikost, nicméně jsou pružné a typografický algoritmus s nimi pracuje, aby vyrovnal pravý okraj.

Důležitou variantou běžné mezery je mezera nezlomitelná, která v místě svého výskytu zakazuje rozdělit řádek. Jinak se chová stejně jako běžná mezera, včetně stlačování/roztahování. Do zdrojového textu ji vložíte znakem `~`. V souvislosti s nimi je záhodno zmínit program *vlna*, který do zdrojového textu automaticky doplní nezlomitelné mezery za jednopísmenné předložky a spojky (ve výchozím nastavení s výjimkou malého „a“ a „i“, lze změnit volbou *-v*).

Kromě základních mezer máte k dispozici i několik dalších příkazů pro mezery užší či širší. Jejich přehled obsahuje tabulka 5. Úzké mezery (`\,`) využijete například k oddělování řádů ve velkých číslech, široké (`\quad`, `\qquad`) k výraznému oddělení, třeba mezi číslem a jménem kapitoly. Zapomeňte na zlozvyky z textových procesorů, kde vynechat

‖	úzká mezera	`\,`
‖	standardní mezera	␣ nebo `\␣`
‖	nezlomitelná mezera	`~`
| |	čtverčíková mezera	`\quad`
| |	dvojčtverčíková mezera	`\qquad`
| |	libovolná mezera	`\hspace{`*rozměr*`}`
| |	nekonečná mezera	`\hfil`, `\hfill`, `\hfilll`
|.....|	mezera s tečkováním	`\dotfill`

Tabulka 5: Vodorovné mezery

velké vodorovné místo znamená opřít se o mezerník. Zde je třeba použít odpovídající příkaz.

Maximální volnost vám poskytne `\hspace`, jímž lze vytvořit vodorovnou mezeru libovolné velikosti. Navíc existuje jeho verze `\hspace*`, kterou LaTeX nesmí vypustit. Po rozdělení řádku totiž standardně dochází k odstranění mezer, které se ocitly na začátku následujícího řádku, aby viditelný text začínal vždy na úrovni levého okraje. Mezera vytvořená pomocí `\hspace*` musí ve výstupu zůstat:

Rozdíl v chování mezer:
obyčejná a
neodstranitelná

```
Rozdíl v~chování mezer:\\
\hspace{3mm}obyčejná a\\
\hspace*{3mm}neodstranitelná
```

Poněkud nezvyklá je mezera vložená příkazem `\hfil`. Má přirozenou šířku 0, ovšem je nekonečně roztažitelná. Zarovnání řádku funguje tak, že pokud se v něm objeví nekonečně roztažitelná mezera, všechny ostatní si zachovají svůj původní rozměr a veškeré zbývající volné místo pohltí tato mezera:

raz dva tři
čtyři

```
raz dva\hfil tři\linebreak
čtyři
```

Pokud je jich více, rozdělí si volné místo rovným dílem. Například centrovaný text v prostředí `center` je interně implementován vložením `\hfil` na začátek a konec každého řádku.

Situace je ve skutečnosti ještě o něco komplikovanější, protože existují tři úrovně nekonečnosti a jim odpovídající příkazy `\hfil`, `\hfill`

a `\hfilll`. Čím více `l` v názvu, tím nekonečnější je příslušná mezera. Roztahují se vždy jen mezery s nejvyšší úrovní nekonečnosti, všechny ostatní zůstanou v přirozené šířce (která je u těchto mezer nulová):

razdva tři `raz\hfil dva\hfill tři`

Vnitřní mechanismy LaTeXu používají první úroveň nekonečnosti. Pokud je chcete „přetlačit“, sáhněte po druhé. Třetí je doporučeno se vyhýbat a nechávat si ji v záloze pro případ nouze.

Speciální variantu nekonečně roztažitelné mezery ztělesňuje příkaz `\dotfill` (k dispozici je jen verze se dvěma „l“), který se chová podobně jako `\hfill`, výslednou mezeru ovšem vyplní tečkováním na úrovni účaří. Uplatnění najde například ve formulářích nebo v obsahu.

Také pro svislé mezery, které se vkládají mezi odstavce či řádky[4], existuje sada příkazů – viz tabulka 6. Jsou definovány obecně, jako malá, střední a velká, konkrétní velikost závisí na základním stupni písma v dokumentu. Opět je k dispozici `\vspace` pro vložení mezery libovolné velikosti. Analogie existuje i k odstraňování mezer: ty, které se po stránkovém zlomu ocitnou na začátku nové stránky, zmizí. Chcete-li vložit neodstranitelnou mezeru, nasaďte `\vspace*`.

malá	`\smallskip`
střední	`\medskip`
velká	`\bigskip`
libovolná	`\vspace{`*rozměr*`}`
nekonečná	`\vfil`, `\vfill`, `\vfilll`

Tabulka 6: Svislé mezery

Rozměry, které jsou součástí příkazů `\hspace` a `\vspace` a mohou se vyskytovat i jinde, se zadávají v obvyklé podobě jako číslo následované jednotkou. Může a nemusí mezi nimi být mezera, obvykle se vynechává.

Různých jednotek je k dispozici přehršel, jak dokládá tabulka 7. Pocházejí jak z různých měrných soustav (metrické a imperiální), tak z tradičních typografických systémů. Poslední dvě jednotky jsou relativní a odvozují svou velikost z aktuálního písma. `1em` odpovídá stupni písma

[4] Pokud se generující příkaz vyskytne uvnitř řádku, bude výsledná svislá mezera vložena za aktuální řádek.

(v typografštině je to jeden čtverčík), zatímco `1ex` jeho střední výšce, tedy výšce minusek (malých písmen). Interně TEX používá škálované body, na než vše převádí. Jsou menší než vlnová délka světla, takže i když udělá zaokrouhlovací chybu, nedokážeme to poznat.

`mm`	milimetr
`cm`	centimetr
`in`	palec, 1 in = 2,54 cm
`pt`	typografický bod, 72,27 pt = 1 in
`pc`	pica, 1 pc = 12 pt
`bp`	„velký“ bod, 72 bp = 1 in
`dd`	didotův bod, 1 dd = 0,376 mm
`cc`	cicero, 1 cc = 12 dd
`sp`	škálovaný bod, 65 535 sp = 1 pt
`em`	čtverčík, odpovídá stupni písma
`ex`	střední výška písma, výška „x“

Tabulka 7: Jednotky podporované TEXem

10 Poznámky

Než se pustím do složitějších věcí, dovolím si krátkou poznámku o poznámkách. Ty pod čarou se vytvářejí příkazem `\footnote{`*text poznámky*`}`. Používají se především pro komentáře či upřesnění textu.

Tento text je doprovázen poznámkou[a].

[a] Která jej doplňuje…

```
Tento text je doprovázen
poznámkou\footnote{Která
jej doplňuje\ldots}.
```

Příkaz v místě svého výskytu vysází značku poznámky (typicky pořadové číslo v horním indexu) a do spodní části stránky umístí její text, sázený menším písmem a od běžného textu vizuálně oddělený. Existují konstrukce (například tabulky), v nichž poznámky nejsou přípustné. V takovém případě je třeba poznámku rozdělit do dvou částí: příkaz `\footnotemark` (bez parametrů) vygeneruje jen značku poznámky a bude použit v místě, kde by se normálně nacházel `\footnote`. Hned za konstrukci blokující poznámku pak umístěte `\footnotetext{`*text poznámky*`}`, který vytvoří text poznámky ve spodní části stránky.

poznámky na okraji

Poznámky na okraji se vkládají příkazem `\marginpar{`*text poznámky*`}`. Slouží především jako navigační nástroj – upozorňují na místa, kde jsou popsány klíčové informace. Poznámka vedle tohoto odstavce byla vložena zdrojovým textem

```
Poznámky\marginpar{\textbf{poznámky\\na okraji}} na..
```

Jsou sázeny na boční okraj vedle řádku, na kterém se vyskytl příkaz `\marginpar`. U jednostranného textu jsou sázeny doprava, u dvoustranného na vnější okraje a u dvousloupcového vždy na přilehlý okraj. Umístění lze změnit příkazem `\reversemarginpar`, po jehož použití budou umisťovány na opačný okraj než normálně.

11 Písmo

Práce s písmem prodělala v LaTeXu nezanedbatelný vývoj. Přístup uplatňovaný v současné verzi nese název New Font Selection Scheme (NFSS) a je postaven na čtyřech základních kategoriích, jimiž je písmo charakterizováno:

Rodina (family) určuje základní charakter písma. LaTeX k ní přistupuje zjednodušeně a rozlišuje jen tři rodiny a jim odpovídající příkazy:

`\rmfamily` antikva (písmo serifové, anglicky RoMan)
`\sffamily` grotesk (písmo bezserifové, Sans seriF)
`\ttfamily` `písmo neproporcionální (TypewriTer)`

Duktus (series) se týká tloušťky jednotlivých tahů. K mání jsou jen dva stupně:

`\mdseries` běžné písmo (MeDium)
`\bfseries` **tučné písmo (BoldFace)**

Tvar (shape) vybírá tvarovou variantu písma. Existují čtyři alternativy:

`\upshape` běžné vzpřímené písmo
`\itshape` *kurzíva (ITalics)*
`\slshape` *skloněné písmo (SLanted)*
`\scshape` kapitálky (Small Capitals)

Kurzíva je písmo, které má skloněnou svislou osu a navíc proti základnímu písmu poněkud pozměněnou kresbu. Hezky je to vidět

při porovnání malého „a“ s „*a*“. Skloněné písmo vznikne prostým nakloněním svislé osy, jinak se tvar nemění. Valná většina písem tuto nepříliš pohlednou variantu nemá a příkaz `\slshape` přepne na kurzívu, stejně jako `\itshape`.

Stupeň (size) rozhoduje o rozměrech písma. LaTeX nepracuje s absolutními hodnotami, místo toho zavádí relativní stupnici danou níže uvedenou skupinou příkazů. Základní velikost (`\normalsize`) je určena třídou dokumentu, ostatní jsou od ní odvozeny:

`\tiny`	nejmenší
`\scriptsize`	velikost horního a dolního indexu
`\footnotesize`	velikost poznámek pod čarou
`\small`	malé písmo
`\normalsize`	normální velikost
`\large`	větší písmo
`\Large`	ještě větší
`\LARGE`	opravdu velké
`\huge`	skoro největší
`\Huge`	největší

Vlastnosti jsou navzájem nezávislé. Změna jedné z nich nijak neovlivní ostatní.

Přejdeme na grotesk, zvětšíme si ho **a přitučníme…**

```
Přejdeme na \sffamily grotesk,
\large zvětšíme si ho
\bfseries a přitučníme\,\ldots
```

Všechny výše uvedené příkazy fungují jako přepínače. Jejich použití změní příslušnou vlastnost písma a tato změna platí, dokud nepoužijete jiný příkaz pro stejnou kategorii, nebo dokud neskončí skupina či prostředí, v němž k ní došlo. Osobně dávám přednost příkazům, které změnu uplatní na svůj argument. Pro první tři kategorie jsou k dispozici alternativní příkazy ve tvaru `\textXY{`*změněný text*`}`, kde XY představuje první dva znaky některého z výše uvedených příkazů. Jejich přehled najdete v tabulce 8.

rodina	duktus	tvar
`\textrm{...}`	`\textmd{...}`	`\textup{...}`
`\textsf{...}`	`\textbf{...}`	`\textit{...}`
`\texttt{...}`		`\textsl{...}`
		`\textsc{...}`

Tabulka 8: Změny vlastností písma

Jedno slovo tučně lze vysázet **takto** nebo **jinak**.

```
Jedno slovo tučně lze vysázet
\textbf{takto} nebo {\bfseries jinak}.
```

Speciálním případem změny písma je `\emph{`*zvýrazněný text*`}`, který se používá, pokud chcete určitý text zvýraznit. Standardně svůj parametr vysází kurzívou, ale sleduje, zda nebyl použit sám v sobě. Pokud zvýrazníte část již zvýrazněného textu, vrátí se ke vzpřímenému písmu (a uvnitř něj případně zase ke kurzívě…).

Část věty *zvýrazníme a uvnitř vypíchneme* jedno *jediné slovo*. Zde už pokračuje běžný text.

```
Část věty \emph{zvýrazníme
a uvnitř vypíchneme
\emph{jedno} jediné slovo}.
Zde už pokračuje běžný text.
```

TEX dostal do vínku unikátní písma, rodinu Computer Modern, kterou ve výchozím nastavení sází i LATEX. Pokud by se vám okoukala a chtěli byste experimentovat s jinými písmy, vznikne problém. Písma jsou dnes obvykle distribuována ve formátu OpenType. Počátkem roku 2011 tento formát podporovaly jen dvě implementace: XETEX a LuaTEX.

V tomto stručném textu nechci zabíhat do podrobností. Berte proto jako dogma, že pokud chcete používat písma ve formátu OpenType, budete muset používat jednu z nich (XELATEX je stabilnější, LuaLATEX progresivnější), zdrojové kódy psát v UTF-8 a pozměnit standardně používané balíky (`inputenc` je zbytečný, `babel` je lépe nahradit jednodušším `polyglossia`).

Pro vlastní práci s písmem je určující balík `fontspec`. Začátek zdrojového kódu by měl vypadat asi takto:

```
\documentclass[a4paper,12pt]{article}
\usepackage{fontspec}
```

```
\usepackage{xltxtra}      % jen pro XeLaTeX
\usepackage{polyglossia}
\setdefaultlanguage{czech}
\setmainfont[Ligatures=TeX,BoldFont={* Bold}]
                                  {Comenia Serif Pro}
\setsansfont[Ligatures=TeX,BoldFont={* Bold}]
                                    {Comenia Sans}
\setmonofont{Vida Mono 32 Pro}
\begin{document}...
```

Příkazy `\setmainfont`, `\setsansfont` a `\setmonofont` (definované v balíku `fontspec`) nastavují písma pro trojici základních rodin. Jejich posledním parametrem je vždy jméno písma, dostupného ve vašem systému – zde například jako antikvu používám písmo „Comenia Serif Pro“. Písma si XeTeX bere z operačního systému. Jejich názvy proto uvádějte v té podobě, ve které je zná váš systém. Veškeré podrobnosti najdete v dokumentaci balíku `fontspec`.

Pro jednorázové přepnutí je k dispozici příkaz `\fontspec`, jehož povinným argumentem je jméno písma a nepovinným případné volby:

Raz dva **3 čtyři.**

```
\fontspec{Gallus Alt}\large
Raz dva \textbf{3 čtyři.}
```

Příkazem `\newfontface\`*`příkaz`*`[`*`volby`*`]{`*`písmo`*`}` si můžete definovat příkaz, kterým kdykoli později přepnete na příslušné písmo, například

Cena **100** až **200 Kč**.

```
\newfontface\bm[Scale=1.3]{Briefmarken}
Cena {\bm 100} až {\bm 200 Kč}.
```

12 Členění dokumentu

Většina běžných textů – jako třeba tento – je členěna do hierarchické struktury kapitol, kapitolek a ještě menších částí. Bývají číslovány, samozřejmostí je, že nadpisy stejné úrovně musí vypadat stejně. LaTeX má pro ně sadu příkazů, jejichž seznam najdete v tabulce 9.

`\part` – nepovinný
`\chapter` – základní pro `report` a `book`
`\section` – základní pro `article`
`\subsection`
`\subsubsection`
`\paragraph`
`\subparagraph`

Tabulka 9: Příkazy pro členění dokumentu

Jsou uvedeny v pořadí od největších logických celků po nejmenší. Rozdělení dokumentu na části příkazy `\part` je nepovinné – zdaleka ne každý dokument je vyžaduje. Další příkazy závisí na tom, jakou třídu jste pro dokument zvolili (hned se k nim dostanu). My jsme zatím používali třídu `article`, ve které chybí úroveň `\chapter` a základním příkazem pro členění dokumentu je `\section`. Pro rozsáhlejší dokumenty s třídou `report` nebo `book` je základní jednotkou kapitola (`\chapter`), zatímco příkazy `\section` označují části kapitol, tedy až druhou úroveň členění.

Všechny příkazy z této skupiny mají jednotný tvar použití:

```
\section[krátký nadpis]{nadpis}
```

Nepovinný argument *krátký nadpis* většinou chybí, typicky se setkáte s příkazy v podobě `\section{Úvod}`. Příkaz vykonává celou řadu činností:

- Vygeneruje číslo dané části textu. Kvůli číslům nesmíte v hierarchii přeskakovat – jestliže jste uvnitř `\section` a chcete ji rozdělit, musíte pro nadpisy jednotlivých částí použít `\subsection`. Kdybyste přeskočili úroveň a sáhli rovnou po `\subsubsection`, objevily by se v automaticky generovaných číslech nuly.
- Vysází číslo a *nadpis* stylem, který odpovídá použité kategorii. To zahrnuje veškeré potřebné kroky, jako je volba písma, vynechání volného místa či dokonce přechod na novou stránku (u kapitol).
- Vloží číslo, *nadpis* a číslo stránky do obsahu. Pokud je *nadpis* příliš dlouhý, nevypadal by v obsahu dobře. Proto je možné určit jeho zkrácenou verzi nepovinným argumentem *krátký nadpis*. V místě

zahájení příslušné části bude vysázen kompletní *nadpis*, do obsahu se ale vloží *krátký nadpis*.

- Upraví záhlaví stránky, pokud jste zvolil styl stránky, který v záhlavích uvádí aktuální názvy částí. Stejně jako v případě obsahu dostane přednost *krátký nadpis*, pokud je uveden.

Nastal čas na malý příklad. Představte si fiktivní uživatelskou příručku k blíže neurčenému programu.

```
\section{Úvod}
Náš skvělý program \emph{BeFeLeMePeSeVeZe}
verze~27.4.3 ...

\section{Instalace}

\subsection{Podmínky instalace}
Před zahájením instalace si ověřte, zda jsou
splněny následující podmínky: ...

\subsection{Postup instalace}
Instalaci zahajte spuštěním ...

\subsection{Konfigurace prostředí}
Po instalaci je vhodné nastavit ...

\section{První spuštění}
Je-li program připraven, můžete ...
```

Ke všem příkazům zahajujícím části textu existují omezené verze, za jejichž jménem bezprostředně následuje hvězdička. Udělají pouze druhý z výše uvedených kroků, tedy vysázejí *nadpis* stylem, který odpovídá příslušné úrovni. Nečíslují jej, nevkládají do obsahu, nemění záhlaví stránek.

Typický příklad jejich využití je předmluva. Chceme, aby byl nadpis *Předmluva* vysázen stejně jako nadpisy sekcí, ale nepovažujeme předmluvu za součást hierarchie vlastního textu, a proto ji nechceme číslovat:

Předmluva

Vážení čtenáři, dostává se vám do rukou publikace ...

```
\section*{Předmluva}

Vážení čtenáři, dostává se vám
do rukou publikace ...
```

Na začátek příloh vložte příkaz `\appendix`, jenž přepne do přílohového režimu. Za ním pokračuje členění textu pomocí `\chapter` či `\section`, které budou ale nyní označovány písmeny místo čísel.

Zcela odlišně je pojat abstrakt dokumentu. K jeho vložení slouží prostředí `abstract`, kterým je třeba jeho text obalit. Je k dispozici pouze pro třídy `article` a `report`.

Když už je řeč o mimořádných prvcích ve struktuře dokumentu, je záhodno zmínit i úvodní titulek či titulní stranu. LaTeX dává na výběr dvě možnosti: můžete se spolehnout na jeho konstrukce, nebo si titulek postavit sami.

Vyrazíte-li první cestou, deklarujte příkazy `\title` název dokumentu, `\author` jeho autora a `\date` datum vytvoření. Všechny tři nevytvářejí žádný viditelný výstup, jen uloží poskytnuté informace. O jejich sazbu se postará `\maketitle`, který vytvoří vlastní titulek. Zahájení dokumentu tedy může vypadat třeba takto:

```
\title{\LaTeX\ pro pragmatiky}
\author{Pavel Satrapa}
\date{červen 2011}
\maketitle
```

Autorů může být několik, v tom případě je oddělte příkazy `\and`. Informace o každém z nich lze navíc formátovat, například pomocí `\\` rozdělit do několika řádků. Mohou také obsahovat příkazy `\thanks`, které se chovají podobně jako `\footnote` v běžném textu.

Chování `\maketitle` závisí na třídě dokumentu. U velkých tříd `report` a `book` vytvoří samostatnou titulní stranu, zatímco pro `article` vytvoří jen úvodní titulek, za nímž pokračuje vlastní text (toto chování lze změnit volbou `titlepage`).

Rozhodnete-li se vytvořit si titulní stranu sami, je rozumné zabalit její obsah do prostředí `titlepage`. To zařídí, že aktuální stránka nebude číslována.

13 Třída dokumentu

Vrátíme se teď na úplný začátek zdrojového textu a podíváme se, co dělají jednotlivé zdejší příkazy a jaké nabízejí možnosti. Klíčovou roli hraje příkaz `\documentclass`, kterým povinně musí zdrojový soubor začínat. V jeho pozadí stojí myšlenka, že potřeby dokumentů se liší v závislosti na jejich určení (kniha je rozdělena do kapitol, dopis nikoli, prezentace potřebuje rozlišovat jednotlivé snímky a podobně). Proto je dokumentu stanovena třída, jež určuje jeho základní charakteristiku.

Název třídy je povinným argumentem příkazu `\documentclass`. LaTeX definuje několik standardních tříd, z nichž nejdůležitější jsou následující:

`article` pro články, tedy strukturované texty menšího rozsahu (jednotky až desítky stránek). Neobsahuje `\chapter`, struktura začíná na úrovni `\section`.

`report` čili zpráva slouží pro středně dlouhé strukturované texty (desítky stránek). Strukturování textu začíná na úrovni `\chapter`. Méně šetří místem než `article`, například každá kapitola začíná na nové stránce.

`book` je určena pro sazbu knih, tedy rozsáhlých textů (stovky stránek). Strukturování opět začíná na úrovni `\chapter`, sazba je opět o něco rozmáchlejší – kapitola zde začíná vždy na pravé stránce, levou případně nechá volnou.

`letter` umožňuje sazbu dopisů. Vůbec nezavádí příkazy pro členění textu a řadu dalších, které v dopisech nebývají potřeba. Zato přidává konstrukce pro generování štítků s adresami a podobně.

`slides` je určena pro sazbu prezentací. Sází velkým bezserifovým písmem, vymezuje hranice snímků a podobně. Upřímně řečeno, nepovažuji LaTeX za ideální nástroj pro tento účel, ale můžete to s ním zkusit.

Chování jednotlivých tříd lze ovlivňovat pomocí voleb. Ty se zadávají v podobě nepovinného parametru příkazu `\documentclass`. Chcete-li jich použít několik, uveďte je všechny jako jeden nepovinný parametr a oddělujte navzájem čárkami. Existuje řada voleb společných pro všechny základní třídy. Za nejčastěji používané lze považovat:

`10pt`, `11pt`, `12pt` určuje základní velikost písma v dokumentu (aneb kolik měří `\normalsize`). Implicitních je 10 bodů, což je vhodné spíše pro menší formáty. Pro papír velikosti A4 je vhodnější `11pt` nebo `12pt`.

`a4paper`, `a5paper`, `b5paper` nastaví velikost papíru, na který se sází. Implicitní je americký formát (`letterpaper`), takže je záhodno některý použít. Existuje několik dalších formátů, které se ovšem v našich končinách nepoužívají.

`landscape` přepne na sazbu na šířku. V podstatě jen navzájem vymění šířku a výšku stránky.

`twoside` způsobí, že LaTeX bude počítat s oboustranným tiskem a při sazbě bude rozlišovat liché (pravé) a sudé (levé) stránky. Implicitně se sází pro jednostranný tisk (`oneside`) a všechny stránky vypadají stejně.

`twocolumn` bude sázet do dvou sloupců. Nemá smysl ji používat, balík `multicol` je mnohem lepší. Implicitní je `onecolumn`.

`draft` deklaruje, že se jedná o pracovní výtisk. LaTeX zvýrazní problémová místa sazby. Implicitní je `final`.

`fleqn` zarovná matematické vzorce nalevo namísto centrování.

`leqno` bude vzorce číslovat vlevo, nikoli vpravo.

`openbib` změní formát seznamu literatury.

`titlepage` existuje pouze pro třídu `article`. Titulek generovaný příkazem `\maketitle` bude umístěn na samostatnou stránku.

`openright`, `openany` řídí, zda kapitola má začínat na pravé stránkce (`openright`) nebo na libovolné stránce (`openany`). Existuje jen pro třídy `report` a `book`.

Dokument třídy `article`, který se má sázet dvanáctibodovým písmem na stránku formátu A4 a má mít samostatnou titulní stránku bychom zahájili příkazem

```
\documentclass[12pt,a4paper,titlepage]{article}
```

14 Rozšiřující balíky

Balíky umožňují významným způsobem rozšířit schopnosti LaTeXu nebo změnit jeho chování. Smí se vkládat jen v preambuli, tedy mezi `\documentclass` a `\begin{document}`. Zvykněte si vkládat je hned na začátku a teprve po nich definovat hodnoty parametrů, nové příkazy a podobně.

Balík použijete příkazem `\usepackage`. Jeho povinným argumentem je jméno balíku. Podobně jako třída může mít balík své volby, jimiž ovlivňujete jeho chování. Jejich názvy a význam je třeba najít v dokumentaci daného balíku. Jelikož se cíle diametrálně liší, neexistují tu žádné společné volby, každý balík je v tomto směru zcela unikátní.

Za příklad poslouží balík `babel`, jímž lze chování LaTeXu přizpůsobit určitému cílovému jazyku. Ten je určen volbou, takže pro češtinu dostáváme již dříve používané

```
\usepackage[czech]{babel}
```

Existují myriády balíků od zcela triviálních (jako `indentfirst`, jehož jediným účinkem je odsazení prvního řádku za nadpisem části textu, který LaTeX standardně neodsazuje) až po komplikovaná monstra typu `babel`. Ty seriózní jsou vystaveny na serverech *The Comprehensive TeX Archive Network (CTAN)*, kde najdete obrovské kvantum kódu a dokumentů souvisejících s TeXem. Dobrá distribuce – jako je TeX Live – je buď rovnou obsahuje, nebo umožní snadno doinstalovat.

Pokud byste snad instalovali ručně, jsou pro balík klíčové dva soubory: ten s příponou *.dtx* je vlastní distribuční soubor balíku a ve stylu „vše v jednom“ obsahuje jak soubory tvořící vlastní balík, tak jeho dokumentaci. Soubor s příponou *.ins* pak říká, jak jej zpracovat. Při instalaci balíku postupujte následovně:

1. Porozhlédněte se, zda neexistuje způsob, jak balík instalovat systémově – instalačním programem vašeho systému či distribuce TeXu. Jen pokud to opravdu nejde, pusťte se do ruční instalace.
2. Obstarejte si soubory *balík.dtx*, *balík.ins* a *balík.pdf*. Doporučuji uložit je do samostatné složky.
3. Spusťte *latex balík.ins*

4. Předchozí krok vygeneruje větší či menší množství souborů. Ten nejdůležitější je *balík.sty* (ale může obsahovat i další součásti). Všechny vytvořené soubory **.sty* zkopírujte do adresáře, kde je vaše instalace TEXu najde. Obvykle mívá každý balík svou složku, aby se v nich dalo snadno orientovat. Doporučuji tuto konvenci dodržovat.

5. Většina instalací si udržuje databázi svých souborů, aby nemusela zdlouhavě hledat na disku. Aktualizujte ji, jinak nové soubory jako by neexistovaly. Například v Linuxu to znamená spustit příkaz *mktexlsr*.

6. Spuštěním *latex balík.dtx* si můžete vytvořit dokumentaci, ale bývá výrazně jednodušší stáhnout si *balík.pdf* (viz krok 2).

Po provedení těchto kroků můžete balík používat. Jinak pokus o překlad dokumentu s `\usepackage{balík}` skončí chybou

```
! LaTeX Error: File `balík.sty' not found.
```

Typickou úlohou balíků je rozšířit možnosti základního LATEXu a doučit jej věci, které původně neuměl. Pro příklad nemusíme chodit nijak daleko, třeba obrázky.

15 Grafika

TEX měl být multiplatformní a produkovat všude přesně stejné výsledky. Do tohoto konceptu grafika příliš nezapadá, protože grafické schopnosti různých systémů a prostředí se výrazně liší. Původní TEX proto grafiku jednoduše nepodporoval.

Jenže uživatelé ji chtěli. LATEX se pokusil o řešení, které by zachovávalo nezávislost na platformě. Přišel s prostředím `picture`, ve kterém jste různými příkazy mohli kreslit úsečky, kružnice a podobně. Grafika měla vektorový základ a opírala se o specializované „písmo“, jež obsahovalo různé grafické prvky a jejich části. Celé to bylo velmi komplikované a použitelné jen se skřípěním zubů (o usnadnění se pokusil kreslicí program *TEXCAD*, jímž lze obrázky kreslit interaktivně).

Obrázek 2: Balík `graphicx` umožňuje vkládat fotografie

Většina implementací postupně nabídla nad rámec původního standardu podporu pro vkládání obrázků v běžně používaných grafických formátech, ovšem každá po svém. K odstranění rozdílů mezi implementacemi vznikly balíky, které nabízejí jednotné prostředky pro práci s grafikou. Nejpoužívanější jsou `graphics` a `graphicx`. Oba mají podobné schopnosti, liší se jen struktura jejich příkazů[5]. Oblíbenější se zdá být `graphicx`, proto se mu budu věnovat.

Chcete-li do svého dokumentu vkládat obrázky, začněte tím, že do preambule přidáte

```
\usepackage{graphicx}
```

Sortiment podporovaných grafických formátů vychází ze schopností konkrétní implementace TEXu, kterou používáte – pročtěte si její dokumentaci, případně experimentujte. Můžete počítat s podporou nejběžnějších rastrových formátů (JPEG, PNG), z vektorových bývá často

[5] `graphicx` interně využívá `graphics`, vlastně jen vytváří sadu alternativních příkazů pro jeho ovládání.

podporováno PDF. Problematičtější je Encapsulated PostScript, ale třeba XeTeX bez problémů umí vložit i *.eps*.

Při přípravě grafiky pro sazbu je důležité zvolit vhodný formát odpovídající charakteru obrázku. JPEG je ideální pro fotografie, pro něž byl vytvořen, ale nehodí se pro diagramy, grafy a obecně obrázky s ostrými hranami. Pro ně bývají nejvhodnější vektorové formáty, případně PNG. Některé implementace nepodporují poměrně populární GIF. Převeďte jej na PNG, například programem *Gimp*.

Pokud používáte implementaci s přímým výstupem do PDF (pdfLaTeX nebo XeLaTeX), balík ji obvykle dokáže automaticky rozpoznat a přizpůsobit se. Kdyby neuspěl, nebo překládáte „čistým“ LaTeXem a do cílového formátu převádíte dokument až později[6], můžete mu ji sdělit buď v nepovinném parametru při vložení balíku, nebo (raději) úpravou konfiguračního souboru *graphics.cfg*.

Vlastní vložení obrázku zajistí příkaz `\includegraphics`, jemuž jako parametr předáte cestu k souboru. Pokud například chcete vložit fotografii ze souboru *foto.jpg*, použijte

```
\includegraphics{foto.jpg}
```

Volitelnými parametry lze přizpůsobit vzhled vloženého obrázku. Mají obecný tvar *vlastnost=hodnota* a chcete-li použít více takových dvojic, oddělte je čárkami. Asi nejčastěji se zásahy týkají velikosti – ta původní vás leckdy překvapí. Máte k dispozici hned několik možností. První z nich je obrázek zvětšit nebo zmenšit proti originálu. V tom případě použijte `scale`, má-li být poloviční, nasaďte

```
\includegraphics[scale=0.5]{foto.jpg}
```

Častěji ale existuje určitá pevně daná cílová velikost. Pomocí parametrů `width` a/nebo `height` ji můžete zadat. Uvedete-li jen jeden, druhý rozměr obrázku se přizpůsobí ve stejném poměru, aby nedošlo k deformaci. Poměrně často potřebujete obrázek stejně široký, jako text. V tom případě vám poslouží předdefinovaná délka `\textwidth` – například obrázek 2 byl vložen příkazem

[6] Například programem *dvips*.

```
\includegraphics[width=\textwidth]{foto.jpg}
```

`\includegraphics` nabízí ještě celou řadu dalších vlastností, které najdete v dokumentaci. Z těch častěji používaných zmíním ještě `angle` pro natočení vložené grafiky. Hodnota se zadává ve stupních, takže obrázek otočený na výšku zajistí

```
\includegraphics[angle=90]{foto.jpg}
```

Vložený obrázek se chová jako písmeno – stane se součástí řádku. To se hodí, pokud chcete třeba do textu vkládat ikony. Častěji ale má být umístěn samostatně, takže mu věnujte samostatný odstavec – před ním a za ním vynechte řádek – a obalte jej prostředím `center` nebo mu předřaďte příkaz `\noindent`, pokud zabírá celou šířku textu.

Zejména v odborné literatuře ale bývá nejčastějším požadavkem, aby obrázek byl umístěn samostatně, opatřen popiskem a číslem, díky němuž se na něj lze odkazovat. Také na tohle LaTeX pamatuje a nabízí prostředí `figure`. Jedná se o tak zvané plovoucí prostředí, jehož obsah nemusí být vysázen v tom místě dokumentu, kde se nachází zdrojový text. LaTeX vyhledá nejbližší vhodné místo a tam plovoucí obrázek vloží. Popisek pak vložíte příkazem `\caption`. Kompletní zdrojový kód obrázku 2 vypadá takto[7]:

```
\begin{figure}[tp]
\includegraphics[width=\textwidth]{foto.jpg}
\caption{Balík graphicx umožňuje vkládat fotografie}
\label{foto}
\end{figure}
```

Prostředí `figure` samo o sobě zajistí jen „plavání“ svého obsahu. Jeho obsah a podobu musíte stanovit příkazy uvnitř něj. Nepovinným parametrem lze ovlivňovat, kam LaTeX smí daný obrázek umístit. Možnosti shrnuje tabulka 10. Zdrojový kód výše tedy připouští umístění plovoucího obrázku buď na začátku stránky s textem, nebo na samostatné

[7] Příkazu `\label` si zatím nevšímejte, definuje návěští pro odkazy a budu se mu věnovat později.

stránce zaplněné plovoucími prvky. Na pořadí písmen v nepovinném parametru nezáleží, jen na jejich složení. Pokud jej neuvedete, použije se implicitní hodnota `tbp`.

`h`	v místě výskytu (here)
`t`	na začátku stránky (top)
`b`	na konci stránky (bottom)
`p`	na samostatné stránce (page of floats)

Tabulka 10: Možnosti pro umístění `figure` a `table`

Při hledání vhodného místa LaTeX dodržuje následující omezení:

1. Obrázek nesmí umístit dříve než na stránku, na které se vyskytlo odpovídající prostředí `figure`.
2. Musí zachovat pořadí podle zdrojového textu.

Pokud se rozhodnete omezit možnosti pro umístění plovoucích obrázků, ponechte vždy dostatečný počet alternativ. Jinak se může stát, že LaTeX nenajde vhodné místo pro daný obrázek a podle pravidla 2 bude jej i všechny následující odkládat a vysází je až na konci kapitoly či dokumentu.

Titulky obrázků vytvořené příkazy `\caption` jsou poměrně nenápadné. Ke změně jejich vzhledu lze použít balík `caption`. Nastavení se provádí v podobě jeho voleb, jež mají tvar čárkami oddělovaného seznamu dvojic *vlastnost=hodnota*. Různých vlastností balík definuje asi 15, zmíním jen několik vybraných.

`labelfont` nastaví písmo generovaného textu „Obrázek N“. Hodnotou je závěrečná dvojice písmen příkazů z tabulky 8 na straně 26). `textfont` určuje písmo pro vlastní název obrázku. Pokud inklinujete k dlouhým názvům, bude vás zřejmě zajímat vlastnost `format` řídící celkovou podobu jmenovky. Standardně je sázena jako centrovaný odstavec. Hodnota `hang` to změní a vysune nápis „Obrázek N“ mimo něj. Takto formátované popisky, navíc s tučným textem generovaného nápisu byste zajistili příkazem (v preambuli):

```
\usepackage[labelfont=bf,format=hang]{caption}
```

`l`	doleva zarovnaný sloupec
`r`	doprava zarovnaný sloupec
`c`	centrovaný sloupec
`p{`*`šířka`*`}`	odstavcový sloupec dané *šířky*
`|`	svislá čára
`@{`*`materiál`*`}`	vloží mezi sloupce *materiál*
`*{`*`počet`*`}{`*`sloupce`*`}`	opakuje definici *sloupců*

Tabulka 11: Specifikace sloupců v prostředí `tabular`

16 Tabulky

Sazba tabulek patří mezi náročnější typografické disciplíny. LaTeX ji zvládá, ovšem s řadou různých omezení. Proto existuje celá řada rozšiřujících balíků, jež doplňují nejrůznější schopnosti.

Základem pro tabulky je prostředí `tabular`, jehož povinným argumentem je specifikace sloupců. V základní podobě každému sloupci v ní odpovídá jedno z písmen `l`, `r`, `c` podle toho, zda má být sloupec zarovnán doleva, doprava nebo na střed. Uvnitř `tabular` je obsah tabulky zapsán po řádcích. Jednotlivé sloupce jsou vždy odděleny znakem `&` a každý řádek je zakončen příkazem `\\`.

Podívejme se na jednoduchý příklad ceníku – dvousloupcová tabulka, jejíž první sloupec s názvy položek je zarovnán doleva a druhý s cenami doprava (proto specifikace sloupců `lr`):

Houska	1,90
Mléko 1,5 l	11,90
Milka mléčná	17,90

```
\begin{tabular}{lr}
Houska & 1,90 \\
Mléko 1,5 l & 11,90 \\
Milka mléčná & 17,90 \\
\end{tabular}
```

Ve sloupcích `l`, `r` a `c` nedochází k rozdělení řádku. Celý jejich obsah bude vysázen do jediného řádku a pokud výsledná tabulka bude příliš široká, jednoduše přesáhne pravý okraj textu. Chcete-li víceřádkový obsah sloupce, použijte sloupec typu `p`, v jehož specifikaci musíte uvést, na jakou šířku má být dotyčný sloupec sázen:

TEX	typografický program vynikající zejména sazbou matematických vzorců
LATEX	nadstavba TEXu s příkazy pro sazbu běžných textů

```
\begin{tabular}{rp{5cm}}
\TeX & typografický program
 vynikající zejména sazbou
 matematických vzorců \\
\LaTeX & nadstavba \TeX u
 s~příkazy pro sazbu
 běžných textů
\end{tabular}
```

Speciální úlohu v definici sloupců má konstrukce `@{}`. Může obsahovat libovolný materiál, který bude vložen v každém řádku tabulky mezi příslušné sloupce. Jako vedlejší efekt zruší vodorovné mezery, které LATEX standardně mezi sloupce vkládá. Podívejte se na příklad, který se od předchozího liší jen specifikací sloupců:

TEX:	typografický program vynikající zejména sazbou matematických vzorců
LATEX:	nadstavba TEXu s příkazy pro sazbu běžných textů

```
\begin{tabular}{r@{: }p{5cm}}
\TeX & typografický program
 vynikající zejména sazbou
 matematických vzorců \\
\LaTeX & nadstavba \TeX u
 s~příkazy pro sazbu
 běžných textů
\end{tabular}
```

U tabulky s větším počtem sloupců oceníte opakování pomocí `*`. Pět doprava zarovnaných sloupců lze specifikovat jako `rrrrr` nebo `*{5}{r}`.

Pravděpodobně budete chtít tabulky různě vylepšovat. Každá buňka se v nich chová jako skupina – změny provedené uvnitř přestávají platit nejbližším `&` či `\\`. Obvyklou dekorací tabulek bývají linky mezi buňkami. Svislé jsou součástí specifikace sloupců: znak `|` v ní znamená „na tomto místě chci mezi sloupci svislou čáru“. Vodorovnou pak zařídí příkaz `\hline` uvnitř tabulky. Má-li být čára dvojitá, jednoduše příslušnou konstrukci zdvojte. Zkusím vylepšit jednoduchý ceník z dřívějšího příkladu:

Zboží	*Cena*
Houska	1,90
Mléko 1,5 l	11,90
Milka mléčná	17,90

```
\begin{tabular}{|l|r|}
\hline
\emph{Zboží} & \itshape Cena \\
\hline \hline
Houska & 1,90 \\
Mléko 1,5 l & 11,90 \\
Milka mléčná & 17,90 \\
\hline
\end{tabular}
```

Případné nepravidelnosti umožňuje do struktury tabulky zanést příkaz `\multicolumn` uvnitř prostředí `tabular`. Má poměrně komplikovaný tvar: jeho první argument udává počet sloupců, které má buňka zabrat, druhý argument určuje formátování (jeho obsah tedy připomíná specifikaci sloupců, ale sloupec je tentokrát jen jeden) a třetí argument obsahuje vlastní text vložený do tabulky.

Opět lehce doplním ukázkovou tabulku. Na konec přidám přání příjemného nákupu, které bude roztaženo přes oba sloupce, první argument proto bude mít hodnotu 2. Chci obsah zarovnat doprava, formát proto bude určen písmenem `r`. Jenže formát v `\multicolumn` zruší veškeré formátovací informace v příslušných sloupcích, včetně přilehlých svislých čar. Abych zachoval levé a pravé ohraničení tabulky, musím je přidat do formátu. Ten proto bude obsahovat `|r|`:

Zboží	*Cena*
Houska	1,90
Mléko 1,5 l	11,90
Milka mléčná	17,90
příjemný nákup	

```
\begin{tabular}{|l|r|}
\hline
\emph{Zboží} & \emph{Cena} \\
\hline \hline
Houska & 1,90 \\
Mléko 1,5 l & 11,90 \\
Milka mléčná & 17,90 \\
\hline
\multicolumn{2}{|r|}{příjemný
nákup} \\
\hline
\end{tabular}
```

Podobně je k dispozici i příkaz `\cline{`*od-do*`}`, který vytvoří vodorovnou čáru přes vybrané sloupce. Jeho parametrem je rozmezí sloupců – čáru přes první dva sloupce by zařídil příkaz `\cline{1-2}`.

Pokud by se vám tabulka zdála příliš hustá, můžete její řádky oddálit pomocí `\arraystretch`. Jedná se o faktor svislého roztažení tabulky, kde hodnota 1 znamená původní rozteč. Takto vypadá roztažení 1,2:

Zboží	*Cena*
Houska	1,90
Mléko 1,5 l	11,90

```
\renewcommand{\arraystretch}{1.2}
\begin{tabular}{|l|r|}
\hline
\emph{Zboží} & \itshape Cena \\
\hline \hline
Houska & 1,90 \\
Mléko 1,5 l & 11,90 \\
\hline
\end{tabular}
```

Buňky roztažené do několika řádků standardní LaTeX neumí. Pokud je potřebujete, použijte balík `multirow`, který zavádí pro obsah tabulky příkaz `\multirow{`*počet řádků*`}{`*šířka*`}{`*text*`}`. První argument obsahuje počet řádků tabulky, které má daná buňka zahrnout, druhý její šířku (nejobvyklejší hodnotou je `*`, což znamená automaticky stanovenou šířku podle obsahu) a třetí obsah buňky. Malý příklad:

	Dva
Raz	Tři
	Čtyři

```
\begin{tabular}{ll}
\multirow{3}{*}{Raz} & Dva \\
 & Tři \\
 & Čtyři \\
\end{tabular}
```

Pokud zjistíte, že vaše tabulky inklinují ke složitému chování ve sloupcích, zvažte nasazení balíku `array`, který rozšiřuje možnosti pro definice sloupců. Zavádí několik jednoduchých alternativ k základním konstrukcím, jako jsou sloupce typu `m` a `b`, které jsou přímou analogií odstavcového sloupce `p` a liší se jen zarovnáním ve svislém směru. U sloupce typu `p` jsou zarovnány horní okraje, u sloupce `m` středy a u sloupce `b` spodní okraje. `!` se pro změnu podobá deklaraci `@`, ale zachovává implicitní mezisloupcovou mezeru.

Zajímavé jsou konstrukce `>{}` a `<{}`, které vkládají na každém řádku před obsah následujícího sloupce (resp. za obsah předchozího) materiál ze svého argumentu. Chcete-li mít v tabulce sloupec tučným písmem, můžete buď do každé jeho buňky vložit příkaz přepínající na tučné písmo, nebo to zařídit v definici sloupců pomocí `>{}` před ním:

`b{`*šířka*`}`	odstavcový sloupec zarovnaný zdola
`m{`*šířka*`}`	odstavcový sloupec zarovnaný na střed
`!{`*materiál*`}`	vloží mezi sloupce *materiál*, zachová mezery
`>{`*materiál*`}`	vloží před následující sloupec *materiál*
`<{`*materiál*`}`	vloží za předchozí sloupec *materiál*

Tabulka 12: Typy sloupců v balíku `array`

Houska	1,90
Mléko 1,5 l	11,90
Milka mléčná	17,90

```
\begin{tabular}{>{\bfseries}lr}
Houska & 1,90 \\
Mléko 1,5 l & 11,90 \\
Milka mléčná & 17,90 \\
\end{tabular}
```

Začnou-li se podobné složitosti ve vašich dokumentech vyskytovat častěji, můžete si příkazem `\newcolumntype` (pocházejícím taktéž z balíku `array`) definovat vlastní typ sloupce – třeba doprava zarovnaný tučný pod názvem `R` – a ten pak používat:

Houska	**1,90**
Mléko 1,5 l	**11,90**
Milka mléčná	**17,90**

```
\newcolumntype{R}{>{\bfseries}r}
\begin{tabular}{lR}
Houska & 1,90 \\
Mléko 1,5 l & 11,90 \\
Milka mléčná & 17,90 \\
\end{tabular}
```

Dalším oblíbeným rozšiřujícím balíkem pro tabulky je `tabularx`. Zavádí stejnojmenné prostředí, jehož povinným parametrem je celková šířka, kterou má tabulka zabrat. Ve specifikaci sloupců se u ní může objevit písmeno `X`, které vytváří odstavcový sloupec (podobně jako `p`), jehož šířka je automaticky určena tak, aby tabulka zabrala požadovanou celkovou šířku. Pokud se objeví několik sloupců typu `X`, daný prostor si rovnoměrně rozdělí.

TEX	typografický program vynikající zejména sazbou matematických vzorců
LATEX	nadstavba TEXu s příkazy pro sazbu běžných textů

```
\begin{tabularx}{7cm}{lX}
\TeX & typografický program
 vynikající zejména sazbou
 matematických vzorců \\
\LaTeX & nadstavba \TeX u
 s~příkazy pro sazbu
 běžných textů
\end{tabularx}
```

Standardní tabulky jsou vždy sázeny na jednu stránku. Jsou-li příliš dlouhé, jednoduše „přetečou“ její rozměry. Pokud potřebujete vícestránkové tabulky, použijte balík `supertabular`, který pro ně definuje prostředí `supertabular`. Používá se úplně stejně jako původní `tabular`, jen si interně hlídá délku tabulky vůči stránce a její pokračování případně přesune na následující stranu.

Další balíky, které by vás v souvislosti s tabulkami mohly zajímat, jsou:

- `hhline` přináší větší kontrolu vodorovných linek v tabulce
- `tabls` pro ovládání svislých mezer
- `colortbl` (nebo volba `table` u balíku `xcolor`) dává k dispozici barvy v tabulkách

Také tabulky je často třeba opatřit popiskem a číslem. Pro tento účel je k dispozici prostředí `table`, které se chová prakticky stejně jako `figure` zajišťující stejnou službu obrázkům (jeho popis najdete na straně 37). Vytvoří plovoucí tabulku, pro niž LATEX najde vhodné umístění, které případně můžete ovlivnit nepovinným parametrem. Poznamenejme, že `table` a `figure` tvoří dvě nezávislé skupiny, uvnitř nichž se nesmí předbíhat (druhý obrázek se nesmí vyskytnout před prvním), ale mezi nimi žádná omezení neplatí. Tabulka proto může předběhnout obrázek a naopak.

Pro ilustraci, tabulka 10 na straně 38 byla vysázena následujícím zdrojovým textem:

```
\begin{table}[htp]
\begin{center}
\begin{tabular}{ll}
h & v~místě výskytu (here) \\
```

```
t & na začátku stránky (top) \\
b & na konci stránky (bottom) \\
p & na samostatné stránce (page of floats)
\end{tabular}
\end{center}
\caption{Možnosti pro umístění figure a~table}
\end{table}
```

17 Odkazy

V textu se hojně odkazuji na obrázky, tabulky a další prvky. Mohl bych samozřejmě vždy napsat konkrétní číslo, jenže takový přístup by byl krátkozraký. Při pozdější úpravě se čísla snadno mohou změnit. Je lepší využívat systémové konstrukce – návěští a odkazy na ně.

Návěští vytvoříte příkazem `\label{`*identifikátor*`}`. Nevysází žádný viditelný výstup, jen si LaTeX interně udělá poznámku o místě jeho výskytu. Každé návěští samozřejmě musí mít jednoznačný *identifikátor*.

Odkázat se na ně můžete příkazem `\ref{`*identifikátor*`}`. Bude nahrazen číslem části textu, ve které se návěští nachází. Jestliže se chcete odkazovat na obrázky či tabulky, vložte příkaz `\label` za `\caption` v prostředí `figure` či `table`. Pokud by byl před `\caption`, bude generovat číslo části textu, nikoli číslo obrázku.

Když odkaz vede do odlehlejší části dokumentu, je velmi žádoucí poskytnout čtenáři i číslo stránky, na které se nachází. K tomu poslouží příkaz `\pageref{`*identifikátor*`}`.

Tento odkaz se nachází v části 17 na straně 45.

```
\label{pokus}Tento odkaz se nachází
v~části~\ref{pokus}
na straně~\pageref{pokus}.
```

Nebuďte překvapeni, že při prvním použití odkazu dostanete místo čísla jen dvojici otazníků a v protokolu o překladu najdete zmínku o nedefinovaném odkazu. Příčinou je, že LaTeX si informace o výskytu odkazů zapisuje do souboru s příponou *.aux* a při použití `\ref` či `\pageref` využívá informace načtené ze souboru *.aux* vytvořeného při minulém průchodu textem. Má tedy zpoždění a je třeba dokument přeložit LaTeXem alespoň dvakrát, aby odkazy byly v pořádku.

Dávejte na konci překladu pozor na varování

```
LaTeX Warning: There were undefined references.
```

jež upozorňuje na použití neznámých odkazů. Pokud přetrvá i po druhém průchodu, máte někde skutečně nedefinovaný identifikátor. Hledejte proto během překladu varování typu

```
LaTeX Warning: Reference `XYZ' on page 31 undefined
```

z nějž se dozvíte, který identifikátor LaTeX nezná a na které straně se nachází odkaz na něj. Může se jednat o opomenutí nebo triviální překlep ve jméně. Další závěrečnou zprávou hodnou pozornosti je

```
LaTeX Warning: Label(s) may have changed.
Rerun to get cross-references right.
```

Říká, že proti předchozímu běhu se změnila čísla částí či stránek u některých odkazů (a protože využívá informace z minula, mohou být odkazy špatně). Jednoduše přeložte dokument znovu a varování zmizí.

18 Matematické vzorce

Příčinou vzniku TeXu byla nekvalitní matematická sazba, takže není překvapující, že v této oblasti opravdu vyniká. LaTeX samozřejmě využívá jeho schopností, k nimž přidal několik obalujících konstrukcí.

Sazba matematiky probíhá ve specializovaném, matematickém režimu. Přesněji řečeno jsou matematické režimy k dispozici hned dva – jeden pro vzorce v textu, a druhý pro samostatné vzorce. Ten první je kompaktnější ve svislém směru, aby výšku obklopujícího řádku změnil co nejméně.

Vzorce v textu jsou oficiálně uzavřeny do prostředí `math`, které ale skoro nikdo nepoužívá, protože jsou k dispozici kratší varianty. Buď můžete před vzorec vložit `\(` a za něj `\)`, valná většina autorů ovšem sáhne po nejstručnější variantě, jíž je znak `$`. Opět použijte po jednom před vzorcem a po něm.

Pro samostatný vzorec je k dispozici prostředí `displaymath`, ale výrazně častěji potkáte kompaktnější formy `\[`...`\]` nebo `$$`...`$$`. Podívejte se na názorný rozdíl mezi vzhledem textového a samostatného vzorce:

Vzorec v textu $\sum_{i=1}^{n} i_n$ je nižší než (totožný) samostatný vzorec

$$\sum_{i=1}^{n} i_n$$

```
Vzorec v~textu
$  \sum_{i=1}^n i_n $
je nižší než (totožný)
samostatný vzorec
$$ \sum_{i=1}^n i_n $$
```

Samostatné vzorce jsou sázeny na vlastní řádek a centrovány. Chcete-li to změnit, použijte volbu `leqno` v úvodním příkazu `\documentclass`. Pak budou zarovnány doleva a odsazeny v konstantní vzdálenosti od levého okraje.

V matematickém režimu jsou zcela ignorovány mezery, typografický algoritmus o nich rozhoduje sám. Je jedno, jestli napíšete `$1+1=2$` nebo `$1 + 1 = 2$`, výsledek bude v obou případech stejný. Ostatně i celá řada dalších činností probíhá automaticky, například text se změní na kurzívu. Pro prvky, které je ve vzorcích zvykem psát vzpřímeně (sin, max, log a další) jsou předdefinovány příkazy (`\sin`, `\max`, `\log`,...). Chcete-li do vzorce vepsat volný text, použijte `\mbox{...}`.

Kromě písmen je k dispozici i řecká abeceda a celá řada speciálních matematických symbolů (odkazy na jejich přehled a vyhledávač najdete v části 6 na straně 13.

Z dalších běžně používaných konstrukcí lze jmenovat horní (`^`) a dolní (`_`) index. Je-li hodnotou jediný znak, lze jej psát bezprostředně za „indexující“ symbol. Pokud je v indexu více znaků, uzavřete je standardně do `{...}`. Příklady obou variant vidíte u sumy výše. Tu vložíte příkazem `\sum`, součin je `\prod` a integrál `\int`, meze jsou určeny horním a dolním indexem. Pomocí indexů lze vyjádřit i další matematické prvky, používají se například u limity:

$$\lim_{x\rightarrow\infty} \frac{1}{x} = 0$$

```
$$ \lim_{x\rightarrow\infty}
\frac{1}{x} = 0 $$
```

Z příkladu jste si jistě domysleli, že `\frac{čitatel}{jmenovatel}` vytvoří zlomek. Pro odmocninu máme `\sqrt{...}`, nepovinným argumentem lze sdělit, kolikátá je. Takže se můžeme podívat třeba na kořeny kvadratické rovnice:

$$x_{1,2} = \frac{-b \pm \sqrt{D}}{2a}$$

```
$$ x_{1,2} =
\frac{-b \pm \sqrt{D}}{2a} $$
```

Speciální kategorií vzorců jsou číslované rovnosti. Používají se tehdy, pokud se chcete na příslušný vzorec v textu odkázat. K jejich vytvoření slouží prostředí `equation` a k systémovému vytvoření odkazu v nich použijte `\label`. Pozdější odkaz na něj příkazem `\ref` bude nahrazen číslem příslušné rovnosti.

Objem kvádru určíme vzorcem

$$V = abc \tag{1}$$

Z 1 plyne...

```
Objem kvádru určíme vzorcem
\begin{equation}
V = abc \label{obj-kvadr}
\end{equation}
Z~\ref{obj-kvadr} plyne\ldots
```

Rovnosti se často vyskytují ve skupinách, kdy se buď vzorec postupně rozvíjí, nebo definujete několik pojmů. Pro takové situace LaTeX nabízí prostředí `eqnarray` (případně `eqnarray*`, pokud je nechcete číslovat). Obsah tohoto prostředí má tvar podobný třísloupcové tabulce – v prvním sloupci jsou levé strany vzorců, ve druhém znaky (ne)rovností a ve třetím pravé strany. Sloupce jsou odděleny `&` a každá z rovností ukončena `\\`. Pokud se některá z nich nemá číslovat, použijte na konci daného řádku (před `\\`) příkaz `\nonumber`. Zkusme ještě jednou kořeny kvadratické rovnice, tentokrát i s diskriminantem:

$$\begin{aligned} D &= b^2 - 4ac \\ x_{1,2} &= \frac{-b \pm \sqrt{D}}{2a} \end{aligned} \tag{2}$$

```
\begin{eqnarray}
D & = & b^2 - 4ac \nonumber\\
x_{1,2} & = &
   \frac{-b \pm \sqrt{D}}{2a}
\end{eqnarray}
```

První sloupec může pochopitelně zůstat prázdný, pokud se jedná o pokračující rozvoj předchozího vzorce.

Když už jsme u tabulek, `tabular` nelze v matematickém režimu používat. Zde je k dispozici prostředí `array`, které se používá úplně stejně, jen své jednotlivé buňky sází v matematickém režimu.

Některé symboly (jako například závorky) potřebují měnit svou výšku podle vzorce, který obklopují. K tomu slouží příkazy `\left`*symbol* a `\right`*symbol*, kde *symbol* je buď znak, nebo příkaz generující příslušný symbol. Příkazy `\left` a `\right` se musí vyskytovat v párech, neuzavřený `\left` způsobí chybu.

$$\left(\frac{a+b}{a-b} \right)^2$$

```
$$ \left( \frac{a+b}{a-b} \right)^2 $$
```

Symboly se v páru mohou lišit a pokud má některý chybět, použijte místo něj tečku, jako třeba v této definici faktoriálu:

$$n! = \begin{cases} 1 & \text{pro } n = 1 \\ n(n-1)! & \text{pro } n > 1 \end{cases}$$

```
$$ n! = \left\{ \begin{array}{ll}
1       & \mbox{pro }n=1 \\
n(n-1)! & \mbox{pro }n>1
\end{array} \right. $$
```

Typickým příkladem využití kombinace pružných závorek a prostředí `array` jsou matice. V nich často uplatníte i tečkovací příkazy pro vodorovnou (`\cdots`), svislou (`\vdots`) a diagonální (`\ddots`) trojici teček:

$$A = \left(\begin{array}{ccc} a_{1,1} & \cdots & a_{1,m} \\ \vdots & \ddots & \vdots \\ a_{n,1} & \cdots & a_{n,m} \end{array} \right)$$

```
$$ A = \left( \begin{array}{ccc}
a_{1,1} & \cdots & a_{1,m} \\
\vdots & \ddots & \vdots \\
a_{n,1} & \cdots & a_{n,m}
\end{array} \right)
$$
```

Pro sazbu složitějších konstrukcí se podívejte na schopnosti balíku `amsmath` a jeho souputníků. Tyto balíky rozšiřují schopnosti matematické sazby i sortiment dostupných symbolů.

Pokud se rozhodnete experimentovat s OpenType písmy, zjistíte, že balík `fontspec` nemá na podobu vzorců žádný vliv. Matematický režim používá svá vlastní písma, do nichž umožňuje mluvit balík `mathspec`. Interně používá `fontspec` – nemusíte tedy v dokumentu uvádět oba, stačí jen `mathspec`. Dokonce v jeho volbách lze uvést i libovolné volby balíku `fontspec` a budou mu předány.

Z vlastních voleb balíku `mathspec` stojí za zmínku `MnSymbol`. Zavede balík `MnSymbol` s rozšířenou sadou matematických symbolů poskytnutých stejnojmenným písmem. Byly vytvořeny pro písmo Minion Pro, nicméně dobře se snesou i s řadou dalších. Nemáme bohužel příliš na výběr, nabídka písem s matematickými symboly je velmi omezená.

Nastavení písma zajistí `\setmathsfont(`*sady*`)[`*volby*`]{`*písmo*`}`, kde *sady* určují, pro které sady znaků má být písmo použito. Jedná se o čárkami oddělovaný seznam, jenž může obsahovat hodnoty `Digits` (číslice), `Latin` (písmena) a `Greek` (řecká abeceda). Je třeba si ověřit, zda všechny příslušné znaky skutečně jsou v písmu obsaženy. Zejména přítomnost řečtiny zdaleka nebývá samozřejmá.

Aby se vybrané písmo objevilo i ve vzpřímených textech (např. lim), je třeba přihodit ještě příkaz `\setmathrm[`*volby*`]{`*písmo*`}`. Písmo Comenia Sans Pro ve vzorcích zajistí kombinace příkazů:

```
\usepackage[MnSymbol]{mathspec}
\setmathsfont(Digits,Latin,Greek){Comenia Serif Pro}
\setmathrm[Ligatures=TeX,BoldFont={* Bold}]
                                  {Comenia Serif Pro}
```

Obvykle budete chtít stejné písmo pro text i vzorce, balík `mathspec` proto nabízí příkaz `\setallmainfonts(`*sady*`)[`*volby*`]{`*písmo*`}`, který provede vše potřebné – zavolá `\setmainfont`, `\setmathsfont` i `\setmathrm`.

19 Dělení slov

LaTeX standardně sází do bloku (tedy se zarovnaným levým i pravým okrajem). Zjednodušeně řečeno si celý odstavec srovná do jednoho řádku, najde si v něm možná místa pro rozdělení řádků a vyzkouší různé jejich kombinace. Za nepodařená místa si uděluje pokuty, za zdařilá si naopak může přiznat odměnu. Pro každou variantu vždy spočítá celkové hodnocení odstavce a na závěr použije tu alternativu, která dosáhla nejlepšího výsledku.

Nejprve se pokusí najít vhodné řešení, aniž by dělil slova. Pokud se mu však nepodaří vejít se do určitého limitu, zkusí to znovu, tentokrát s dělením. Provádí je automaticky a využívá vzory pro daný jazyk, jejichž načtení je součástí činnosti balíku `babel`. Jako každý automat ale občas dělá chyby.

Pokud dojde ke špatnému rozdělení slova, máte několik možností, jak situaci napravit. Nejjednodušší je doporučit vhodné rozdělení přímo na místě příkazem `\-`. Tím říkáte „zde je vhodné rozdělit slovo“ a zároveň

zakazujete jeho dělení kdekoli jinde. Slovo může pochopitelně obsahovat několik příkazů `\-`, které pak představují jediné přípustné alternativy pro rozdělení.

Jakmile slovo obsahuje příkaz `\-`, smí je LaTeX rozdělit jen v tom místě.

```
Jak\-mile slo\-vo ob\-sa\-hu\-je
pří\-kaz \cmd{-}, smí je \LaTeX\
roz\-dě\-lit jen v~tom místě.
```

Nedojde-li k rozdělení, nemá příkaz žádný viditelný účinek. `\-` je ve skutečnosti zkratkou švýcarského nožíku pro dělení, jímž je příkaz `\discretionary{`*před*`}{`*za*`}{`*bez*`}`. Říká, jak má dané místo vypadat *bez* rozdělení a pokud zde k rozdělení dojde, co má být *před* ním (na konci prvního řádku) a co *za* ním. Používá se ve slovech, která při rozdělení mění tvar – například v německém Bettuch při rozdělení přibude jedno „t“: Bett-tuch (`Bett\discretionary{-}{t}{}uch`).

Pomocí `\-` je vhodné napravovat mimořádnosti. Pokud se dotyčné slovo v textu opakuje častěji, je vhodnější oznámit rozdělovacímu mechanismu obecnou výjimku, která bude platit v celém textu. K tomuto účelu slouží příkaz `\hyphenation`. Uveďte jej v záhlaví dokumentu a jako argument zapište mezerami oddělovaný seznam slov, v nichž pomlčkami vyznačíte přípustná místa pro rozdělení.

Častý problém například bývá s anglickými slovy a jmény začínajícími na „ne“, což česká pravidla pojímají jako negující předponu a tudíž velmi vhodné místo k rozdělení. Dělení Ne-wton ovšem není to pravé ořechové, proto nasaďte

```
\hyphenation{New-ton Net-Wa-re}
```

Chcete-li zabránit rozdělení konkrétního slova, sáhněte po konstrukci `\mbox{`*slovo*`}`. LaTeX si parametr `\mbox` vysází předem a vloží pak do řádku jako celek, s nímž už nelze jakkoli manipulovat. Pokud byste chtěli dělení slov plošně omezit, zvyšte pokutu za dělení slov nastavením parametru `\hyphenpenalty`. Jeho implicitní hodnotou je 50, zvýšením se zhorší skóre rozdělených řádků a LaTeX se jim bude snažit vyhýbat. O úplné potlačení se postará

```
\hyphenpenalty=10000
```

protože hodnota 10 000 v TeXu platí jako nekonečno.

20 Řádkový zlom a odstavec

Nejvhodnějšími místy pro rozdělení řádku jsou samozřejmě mezery mezi slovy. Až na výjimky. Chcete-li vložit mezeru, ve které je zakázáno rozdělit řádek, zapište do zdrojového kódu místo ní znak `~`, který mezeru doprovodí pokutou 10 000 za případné rozdělení.

Ne vždy to pomůže. Problémem je ono nekonečno s hodnotou 10 000 – jakmile celkové skóre řádku překročí tuto hodnotu, je TEXu jedno o kolik. 10 050 je pro něj stejně špatné jako 100 000, obě hodnoty jsou nekonečné. Pokud se nepodaří najít žádnou variantu, která by skončila výsledkem řádku pod 10 000, TEX nad ním zlomí hůl a odstavec optimalizuje podle ostatních. Může se pak stát, že řádek rozdělí i v místě, kde jste to explicitně zakázali, a žádná pokuta jej neodradí.

V takovém případě je záhodno si vzpomenout, že kromě biče existuje i cukr a nějaká místa pro rozdělení naopak doporučit. Slouží k tomu příkaz `\linebreak`, který do místa svého výskytu vloží naopak odměnu za rozdělení. Můžete mu předat nepovinný parametr vyjadřující sílu vašeho doporučení. Hodnotou je celé číslo v rozsahu od 0 do 4. Nula znamená jemné doporučení, zatímco 4 (což je implicitní hodnota) motivuje sázecí algoritmus tak silně, že je rozdělení v podstatě jisté. Je to vidět na následujícím příkladu, kde je první řádek extrémně roztažen:

Zde je dobré místo
pro rozdělení řádku.

```
Zde je dobré místo\linebreak[4]
pro rozdělení řádku.
```

Existuje i protipól, příkaz `\nolinebreak`, který rozdělení řádku v daném místě naopak nedoporučuje. Také on má nepovinný argument, který se používá stejně jako u `\linebreak`.

Jinou cestou ke zlomení řádku je příkaz `\\`. Na rozdíl od `\linebreak` ovšem rezignuje na zarovnání pravého okraje – říká, že řádek zde končí a má být vysázen ve své přirozené šířce:

Řádek končí
aniž by se TEX snažil zarovnat jeho pravý okraj.

```
Řádek končí\\
aniž by se \TeX\ snažil zarovnat
jeho pravý okraj.
```

Synonymem `\\` je `\newline`. Na rozdíl od něj však `\\` nabízí několik doplňků. Jednak lze připojit nepovinný argument udávající svislou mezeru, která se má přidat mezi stávající a následující řádek:

Řádek končí

a TEX pod něj přidá svislou mezeru dané velikosti.

```
Řádek končí\\[6pt]
a~\TeX\ pod něj přidá svislou
mezeru dané velikosti.
```

A konečně varianta `\\*` zakazuje za ukončeným řádkem ukončit stránku. Musí se tedy objevit na stejné stránce s řádkem následujícím. Také této variantě příkazu můžete přidat svislou mezeru.

Ne pokaždé se podaří najít vhodný řádkový zlom. Velikost mezer se pohybuje jen v určitém rozmezí, slova nelze dělit kdekoli, občas se vyskytnou delší nedělitelné bloky a také pružnost případných dalších prvků řádku mívá své limity. Pokud se nepodaří najít způsob, jak dodržet omezení a zároveň požadovanou šířku řádku, nabízí LATEX dvě alternativy:

- Podle implicitního chování dodrží meze pružnosti a poruší pravý okraj – inkriminovaný řádek bude delší. V protokolu o překladu je to doprovázeno upozorněním „`Overfull \hbox`“. Pokud v příkazu `\documentclass` použijete volbu `draft`, přidá se na konec takového řádku výrazný černý obdélník, aby upozornil na problém. Toto chování lze případně vyvolat příkazem `\fussy`.
- Druhou možností je, že bude dodržen pravý okraj, ovšem za cenu nadměrného roztažení mezer a vytvoření řídkého řádku (jeden takový jste viděli v prvním příkladu této části). V protokolu se objeví upozornění na „`Undrefull \hbox`“. Pro běžnou praxi mi tento přístup připadá jako vhodnější. Je třeba jej ovšem ručně aktivovat, což zajistí příkaz `\sloppy`. Nejvhodnější místo pro něj se nachází v preambuli dokumentu. Existuje také prostředí `sloppypar`, které v tomto režimu vysází svůj obsah.

V souvislosti s odstavci stojí za zmínku tři důležité parametry, které ovlivňují jejich podobu. Standardně LATEX vkládá vodorovné odsazení na začátek prvního řádku a nevynechává žádnou svislou mezeru mezi odstavci. Zabránit vodorovnému odsazení jednoho konkrétního odstavce lze vložením příkazu `\noindent` na jeho začátek (nebo je naopak pomocí `\indent` vynutit).

Plošně můžete vzhled odstavců ovlivnit pomocí `\parskip` (svislá mezera mezi nimi) a `\parindent` (vodorovná mezera na začátku prvního řádku), jejichž hodnoty lze změnit příkazem `\setlength`. Dávám přednost svislému oddělení odstavců, proto ve svých dokumentech nuluji vodorovné odsazení a jako svislé používám polovinu řádkové rozteče:

```
\setlength{\parindent}{0pt}
\setlength{\parskip}{0.5\baselineskip}
```

Máte-li chuť na hrátky s řádkováním, sáhněte po `\baselinestretch`. Určuje faktor pro roztažení řádkových roztečí. Implicitní hodnotou je 1, která je ponechá v původní podobě. Zvětšením řádky oddálíte, zmenšení rozhodně nedoporučuji. Někdy vyžadované „řádkování 1,5“ lze zařídit pomocí

```
\renewcommand{\baselinestretch}{1.3}
```

21 Stránkový zlom

Sazba stránek používá jednodušší přístup. Na rozdíl od řádků v odstavci se u stránek TEX nesnaží o globální optimum. Jakmile nastřádá dostatek materiálu na stránku, najde nejvhodnější místo pro její zlom (opět na principu pokut/odměn), použije je a pustí se do další stránky. Nedokáže dohlédnout, že pokud by aktuální stránku lehce zhoršil, příštích pět mu vyjde mnohem lépe.

Stránkový zlom lze ovlivňovat příkazy `\pagebreak` a `\nopagebreak`. Jejich význam i použití se podobá výše zmíněnému `\linebreak`. Také `\\` (resp. `\newline`) má svou stránkovou analogii: příkaz `\newpage`, který okamžitě zahájí novou stránku.

Sofistikovanější nadstavbou `\newpage` je dvojice příkazů `\clearpage` a `\cleardoublepage`, které také znamenají přechod na novou stránku (v případě `\cleardoublepage` novou lichou stránku), ovšem ještě předtím vysázejí všechen dosud odkládaný plovoucí materiál (`figure` a `table`). Interně je volá například příkaz `\chapter`, aby se obrázky či tabulky z předchozí kapitoly nezavlékaly do další.

Algoritmus stránkového zlomu sice nedokáže posoudit aktuální stránku v globálním kontextu, autor to ale zvládne a může stroj k lepšímu řešení postrčit. K tomu lze využít `\enlargethispage{`*rozměr*`}`, kterým přičtete zadaný *rozměr* k výšce zrcadla (textové oblasti) aktuální stránky. Díky tomu na ni lze přesunout jeden řádek z následující stránky či naopak jeden, dva řádky odložit na další. Účinek se týká vždy té stránky, na níž se příkaz ocitne. Chcete-li ji zkrátit o řádek, lze využít hodnotu `\baselineskip` obsahující rozteč účaří:

```
\enlargethispage{-\baselineskip}
```

Považuje se za nevhodné, aby na stránce zbyl jen počáteční řádek odstavce (tzv. sirotek) nebo na další stránku přetekl jen jeho poslední řádek (vdova). TEX si za tyto prohřešky udělí pokutu, jejíž implicitní výše ale nezabrání jejich občasnému výskytu. Je vhodné nastavit v preambuli hodnoty `\clubpenalty` a `\widowpenalty` na „nekonečných“ 10 000.

Implicitně se LATEX snaží dodržovat jednotnou výšku stránky – roztahuje či stlačuje svislé mezery tak, aby vyplnil veškeré dostupné místo. Občas to může vést k nepěkně řídkým stránkám, například když byl odstavec odložen až na další stránku, aby nevznikl sirotek. Použijete-li v preambuli příkaz `\raggedbottom`, LATEX rezignuje na zarovnání dolních okrajů a sází stránky v jejich přirozené výšce. Obvykle proto před zahájením dokumentu nasazuji

```
\clubpenalty=10000
\widowpenalty=10000
\raggedbottom
```

22 Obsah

Vytvoření obsahu je dětská hračka. Tam, kde jej chcete mít, jednoduše použijete příkaz `\tableofcontents` a on v místě svého výskytu vytvoří obsah složený z nadpisů použitých v příkazech pro členění textu.

Analogickým způsobem lze do dokumentu vložit seznam obrázků (příkaz `\listoffigures`) a tabulek (`\listoftables`). Vycházejí z příkazů `\caption` v prostředí `figure`, respektive `table`.

Stejně jako v případě odkazů, i obsahy mají jedno kolo zpoždění. Při průchodu dokumentem si LaTeX ukládá informace do souboru, který při příštím zpracování textu načte a informace z něj využije[8]. Každý typ obsahu má svůj samostatný soubor, jejich přípony najdete v tabulce 13.

`toc`	obsah (table of contents)
`lof`	seznam obrázků (list of figures)
`lot`	seznam tabulek (list of tables)

Tabulka 13: Přípony souborů pro různé typy obsahů

Obsahy se sice vytvářejí automaticky, nicméně občas je třeba do nich ručně zasáhnout. Chcete-li přidat regulérní položku, poslouží vám příkaz `\addcontentsline{`*soubor*`}{`*styl*`}{`*text*`}`. *Soubor* určuje, do kterého z obsahových souborů ji chcete přidat. Jeho hodnotou je jedna z přípon podle tabulky 13. Jako *styl* použijte v případě klasického obsahu název části textu, od níž má být odvozen vzhled dotyčné položky. V případě obrázků má *styl* konstantní hodnotu `figure` a pro tabulky `table`. *Text* pak obsahuje text položky. O číslo stránky se nestarejte, LaTeX si je doplní automaticky.

Typický příklad využití: v úvodu chci mít předmluvu, jejíž nadpis má vypadat stejně jako název sekce, nechci ji automaticky číslovat (proto použiji příkaz `\section*`), ale rád bych ji zařadil do obsahu. Zahájím ji proto konstrukcí

```
\section*{Předmluva}
\addcontentsline{toc}{section}{Předmluva}
```

Do obsahu lze vložit prakticky libovolný materiál prostřednictvím příkazu `\addtocontents{`*soubor*`}{`*materiál*`}`. Na místě *souboru* opět figuruje jedna z přípon podle tabulky 13 a *materiál* bude vložen do daného souboru. Řekněme, že se vám příliš nelíbí rozložení obsahu na stránky a chtěli byste, aby kapitola „Praktické zkušenosti“ v obsahu začínala na nové stránce:

```
\addtocontents{toc}{\newpage}
\chapter{Praktické využití} ...
```

[8] Jinak to dost dobře ani nelze udělat. Obsah často bývá na začátku, kdy LaTeX ještě netuší, jak bude následující dokument vypadat.

Výsledkem bude, že před položku odpovídající dané kapitole se do obsahu vloží příkaz `\newpage`, který zahájí novou stránku.

23 Seznam literatury

V odborné literatuře by neměl chybět seznam literatury, případně s citacemi jednotlivých publikací v textu (příklad vidíte na straně 81). Každá položka má přiřazenu určitou viditelnou značku, nejčastěji pořadové číslo nebo zkratku autora a roku vydání ve stylu [Knu94].

Ve zdrojovém textu je celý obklopen prostředím `thebibliography`. U jeho zahájení musíte uvést povinný argument odpovídající nejširší značce. Jedná se o libovolný řetězec, který nevytvoří žádný viditelný výstup. LaTeX si pouze změří, jak je široký, a podle něj dimenzuje vodorovnou mezeru určenou pro značky položek. Pokud je číslujete, obvyklou hodnotou bývá `{99}` (dvě číslice by měly stačit). Používáte-li trojpísmennou zkratku autora a letopočet, doporučuji `{Mmm99}`.

Každá položka v seznamu začíná příkazem `\bibitem{`*identifikátor*`}`, která vytvoří pro nově zahájenou publikaci vizuální značku a zároveň jí přiřadí *identifikátor*, jehož pomocí se na ni můžete odkázat. Standardní generovanou značkou je pořadové číslo. Chcete-li jinou, poručte si ji nepovinným argumentem. Zbytek seznamu literatury je tvořen zcela standardním způsobem pomocí obvyklých příkazů.

Reference

[1] Donald E. Knuth: *The TeXbook*. Addison Wesley Professional, 1994

[La94] Leslie A. Lamport: *LaTeX: A Document Preparation System*. Addison-Wesley Professional, 1994


```
\begin{thebibliography}{Mm99}

\bibitem{knu}
Donald~E. Knuth:
\emph{The \TeX book}.\\
Addison Wesley Professional,
1994

\bibitem[La94]{lam}
Leslie~A. Lamport:
\emph{\LaTeX: A~Document
Preparation System}.\\
Addison-Wesley Professional,
1994

\end{thebibliography}
```

Na publikaci ze seznamu literatury se lze snadno odkázat. Slouží k tomu příkaz `\cite{`*identifikátor*`}`, který v místě svého výskytu vysází značku příslušné publikace. Opět čerpá informace z předchozího průchodu textem, takže novou položku nebude znát hned. Můžete mu také přidat nepovinný argument, jehož obsah bude přidán do značky:

Vše popisuje Knuth v knize [1]. Za pozornost stojí i [La94, strana 49].

```
Vše popisuje Knuth
v~knize~\cite{knu}.
Za pozornost stojí
i~\cite[strana~49]{lam}.
```

Seznam literatury lze vytvářet i automaticky na základě citací v textu a databáze používané literatury. Slouží k tomu program BibTeX a vyplatí se o něm uvažovat, pokud píšete především odborné texty s hojnými citacemi.

24 Rejstřík

Odborná publikace většího rozsahu by měla být doprovázena rejstříkem, který čtenářům usnadní orientaci. Jeho vytvoření patří k těm složitějším úkolům, a to jak pro LaTeX, tak pro vás.

Začněte tím, že v preambuli použijete balík `makeidx` a společně s ním příkaz `\makeindex`:

```
\usepackage{makeidx}
\makeindex
```

Vlastní data pro rejstřík vytvoříte pomocí příkazů `\index`. Vždy když v textu popisujete určitý pojem, který se má vyskytnout jako heslo v rejstříku, připojte k němu příkaz `\index{`*heslo*`}`. Příkaz nevytváří žádný viditelný výstup, jen poznamená, že na dané stránce se vyskytlo uvedené *heslo*:

K vytvoření hesla rejstříku slouží...

```
K~vytvoření hesla\index{heslo
rejstříku} rejstříku slouží\ldots
```

Údaje o obsahu rejstříku si LaTeX zapisuje do souboru s příponou *.idx*. Jednotlivá hesla jsou v něm v pořadí, ve kterém se objevila ve zdrojovém textu. Mohou se pochopitelně vyskytovat opakovaně, pokud byla uvedena v několika příkazech `\index`.

Je třeba je uspořádat – seřadit abecedně a opakované výskyty stejného hesla sloučit do jedné položky s několika čísly stránek. To je ovšem úloha, jejíž naplnění leží za hranicemi LaTeXu. Slouží k tomu specializovaný program *makeindex*, jemuž jako parametr předáte jméno zpracovávaného dokumentu.

Problém může vzniknout s češtinou. Standardní *makeindex* řadí hesla jednoduše podle kódů jednotlivých znaků, takže veškeré znaky s akcenty se ocitnou až za anglickou abecedou. Pokud pracujete v Linuxu nebo jiném systému podporujícím locales, můžete programu volbou *-L* nařídit, aby řadil podle pravidel definovaných v nich. Zpracováváte-li dokument *navod.tex*, zavolejte

```
makeindex -L navod
```

Jestliže volba *-L* ve vašem systému nefunguje, zkuste místo *makeindex* použít jeho českou adaptaci *csindex*, případně ve verzi upravené pro MS Windows. Volbou *-z* jí sdělte, v jakém kódování je zdrojový text zapsán.

Modernější variantou je rejstříkový preprecesor *xindy*, konkrétně jeho varianta *texindy* určená pro TeX. Program byl od počátku vyvíjen pro vícejazyčné prostředí, stačí mu prostřednictvím voleb napovědět, který je ten váš. Jeho použití pro náš ukázkový dokument by vypadalo takto:

```
texindy -I latex -L czech -C utf8 navod.idx
```

Volba *-I* stanoví formát vstupního souboru, *-L* určí jazyk a *-C* kódování.

Program (ať už použijete kterýkoli) načte informace z *navod.idx*, uspořádá je a výsledek zapíše do souboru *navod.ind*, který obsahuje formátovanou podobu rejstříku.

Zbývá už jen poslední krok, vložit rejstřík do dokumentu. To zajistí příkaz `\printindex`, který uveďte na místě, kde se má rejstřík vyskytovat (obvykle na konci publikace). Na jeho místo bude vložen aktuální obsah souboru *.ind*. Pozor, k jeho aktualizaci nedochází automaticky. Na rozdíl od obsahu či odkazů nestačí opakovaně spustit LaTeX, soubor *.ind* bude změněn až programem *makeindex*, *csindex* či *texindy*. Chcete-li mít jistotu, že informace budou aktuální, proveďte tuto čtverylku:

1. LaTeX – vytvoří informace pro obsah, odkazy apod.
2. LaTeX – vloží obsah (tím může posunout stránkování) a aktualizuje informace pro obsah a spol.
3. *makeindex* – vytvoří formátovaný rejstřík
4. LaTeX – finální průchod, využívané informace jsou aktuální

Rejstřík může kromě prostých hesel obsahovat i různé formy zvýraznění a podobně. Rozhodující jsou zde schopnosti programu sestavujícího formátovanou verzi. Všechny zmiňované programy podporují následující konstrukce:

Podhesla v argumentu příkazu `\index` oddělte vykřičníkem. Kdybych v předchozím příkladu chtěl do rejstříku kromě hesla „heslo rejstříku“ přidat také podheslo „heslo“ v rámci hesla „rejstřík“, přidal bych příkaz `\index{rejstřík!heslo}`. Lze používat až tři úrovně.

Zvýraznění čísla stránky obstará konstrukce `|`*příkaz*, kde *příkaz* slouží ke zvýraznění čísla. Zapisuje se bez zpětného lomítka. Má-li být číslo stránky tučné, použijte `\index{heslo|textbf}`. Odkaz na jiné heslo (heslo *viz* jiné) vloží `\index{heslo|see{jiné}}`.

Pokud se heslu věnuje rozsáhlejší část textu, použijte na jejím začátku `\index{heslo|(}` a na konci `\index{heslo|)}`. V rejstříku se pak u hesla objeví interval, který začíná číslem stránky s `|(` a končí číslem stránky s `|)`.

Znak `@` způsobí, že příslušné heslo bude abecedně zařazeno podle textu před ním, ovšem do rejstříku se vloží text za ním. Například logo LaTeXu, které se řadí jako řetězec „LaTeX“ `\index{LaTeX@\LaTeX}`.

25 Uspořádání stránky

Vlastní text bývá doplněn různými orientačními prvky, jako je číslo stránky, název aktuální kapitoly a podobně. Jejich podobu určíte příkazem `\pagestyle`, jehož argumentem je stránkový styl podle tabulky 14. Implicitní je `\pagestyle{plain}`. Změna platí trvale, počínaje aktuální stránkou. Chcete-li změnit styl jen pro aktuální stránku, použijte `\thispagestyle`, jehož argumentem je opět jeden ze stylů. U následující stránky se vrátí zpět styl platný před použitím příkazu.

`plain`	číslo stránky dole uprostřed
`empty`	nikde nic
`headings`	číslo stránky a jméno části nahoře
`myheadings`	jako předchozí, ale obsah řídíte sami

Tabulka 14: Styly pro příkaz `\pagestyle`

Použijete-li styl `myheadings`, obsah záhlaví ovládáte ručně pomocí příkazů `\markboth{`*levé záhlaví*`}{`*pravé záhlaví*`}`, který nastaví odděleně levé a pravé záhlaví při oboustranném tisku, a `\markright{`*pravé záhlaví*`}`, jímž nastavíte pouze pravé záhlaví.

Pokud se chcete opravdu vyřádit, sáhněte po balíku `fancyhdr`, který definuje styl stránky `fancy`. V něm mají záhlaví i zápatí po třech částech (levá L, střed C a pravá R) a ty se rozlišují na lichých (O) a sudých (E) stránkách (při jednostranném tisku jsou všechny stránky formátovány jako liché). K jejich nastavení slouží příkazy `\fancyfoot` a `\fancyhead` s nepovinným argumentem, který určuje prvky záhlaví/zápatí, a povinným obsahem, jenž chcete přiřadit. Například styl stránek tohoto textu vnikl následovně:

```
\usepackage{fancyhdr}
\pagestyle{fancy}

% zrušit implicitní obsah a čáru pod záhlavím
\fancyhead{}
\renewcommand{\headrulewidth}{0pt}

% číslo stránky (\thepage) na venkovní okraje -
% vpravo na lichých stránkách, vlevo na sudých
\fancyfoot{}
\fancyfoot[RO,LE]{\textbf{\color{black!50}\thepage}}

%vystrčit číslo stránky mimo hranice textu
\fancyfootoffset{3em}
```

Pokud chcete v okolí pracovat s názvy částí textu, můžete využít standardní příkazy LaTeXu `\leftmark` a `\rightmark`, v nichž je vždy to, co by LaTeX sázel na levou/pravou strnu při stylu `headings`.

Se stránkovým stylem souvisí i styl číslování. Ten lze stanovit příkazem `\pagenumbering{`*styl*`}`, kde *styl* může mít jednu z hodnot uvedených v tabulce 15. Vedlejším efektem změny stylu číslování je reset čítače stránek – po změně stylu se začíná vždy číslovat od jedničky. To dobře odpovídá konvencím americké typografie, z jejíhož prostředí LaTeX pochází. Zde se úvodní obsah, předmluva a případné další části číslují samostatně malými římskými číslicemi, zatímco stránky vlastního dokumentu jsou číslovány arabsky od jedničky.

`arabic`	arabské číslice (3)
`roman`	římské číslice minuskami (iii)
`Roman`	římské číslice verzálkami (III)
`alph`	písmena – minusky (c)
`Alph`	písmena – verzálky (C)

Tabulka 15: Styly číslování pro příkaz `\pagenumbering`

Pokud byste potřebovali s číslem stránky nestandardně manipulovat, z hlediska LaTeXu se jedná o běžný čítač jménem `page`. Jeho hodnotu lze změnit příkazem `\setcounter{page}{`*hodnota*`}`.

Mohli byste potřebovat číslovat stránky nikoli průběžně, ale v rámci kapitol. O to se postará balík `chappg`.

26 Boxy

Když se začnete hlouběji zajímat o způsob, kterým TeX sází, dříve či později narazíte na pojem box. Z pohledu TeXu je boxem prakticky vše – počínaje jednotlivými písmeny a celou stránkou konče. Během své činnosti postupně skládá z existujících boxů složitější a složitější, až se dopracuje ke stránce. Skládání může probíhat jak ve vodorovném (slova ze znaků, řádky ze slov a mezer), tak ve svislém směru (stránka z řádků a svislých mezer).

Pokud chcete, můžete si box vytvořit sami. Vodorovně skládaný vznikne jedním z příkazů

```
\mbox{text}
\makebox[šířka][zarovnání]{text}
```

`\mbox` je jednodušší – vysází svůj obsah jako vodorovný box, jehož šířku určí automaticky podle obsahu. Naproti tomu v případě `\makebox` si můžete poručit jak celkovou šířku boxu, tak zarovnání jeho obsahu. Implicitně se centruje, písmenem `l` nařídíte zarovnání doleva, `r` doprava a `s` roztažení na celou šířku.

Podobně se chovají příkazy `\fbox` a `\framebox`, které ovšem navíc kolem boxu vykreslí rámeček:


```
\fbox{raz dva}\\
\framebox[5cm]{tři čtyři}\\
\framebox[5cm][l]{pět šest}\\
\framebox[5cm][r]{sedm osm}\\
\framebox[5cm][s]{devět deset}
```

Šířku čáry lze ovlivnit parametrem `\fboxrule` a její odstup od obsahu boxu pomocí `\fboxsep`. V preambuli tohoto dokumentu jsem nastavil

```
\setlength{\fboxrule}{0.75pt}
\setlength{\fboxsep}{6pt}
```

Sestavený box je z pohledu TEXu monolit – jakmile jednou vnikne, už se nezmění. Použijete-li uvnitř odstavce `\mbox`, jeho obsah vytvoří box a ten se pak jako celek bude účastnit řádkové sazby. Tím se dá na jedné straně zabránit řádkovému zlomu uvnitř `\mbox` (box vznikne ještě před hledáním vhodných míst pro řádkový zlom), ale na druhé straně případné mezery uvnitř nebudou pracovat a mohou mít jinou šířku, než běžné mezery na řádku, což působí dost rušivě:

Mezera v boxu si zachová svou velikost.

```
Mezera \mbox{v boxu} si\linebreak
zachová svou velikost.
```

Příkaz `\raisebox{`*posun*`}[`*výška*`][`*hloubka*`]{`*text*`}` posune vodorovný box nahoru či dolů. Vůči *textu* se chová podobně jako `\mbox`: vysází jej v přirozené šířce, ovšem následně jej zdvihne o určený *posun* nad účaří

řádku. *Posun* může samozřejmě být i záporný. Bez volitelných parametrů určí výšku a hloubku automaticky podle velikosti posunutí a rozměrů původního boxu. To může narušit řádkové rozestupy, takže máte možnost *výšku* a *hloubku* výsledné konstrukce předepsat ručně.

Jedno slovo zdvihnu.

```
Jedno slovo \raisebox{1ex}{zdvihnu}.
```

K vytvoření svislého boxu lze použít buď příkaz `\parbox`, nebo prostředí `minipage`. Jejich účinek je podobný, liší se jen způsob použití:

```
\parbox[zarovnání]{šířka}{text}
\begin{minipage}[zarovnání]{šířka}text\end{minipage}
```

V obou případech bude *text* sázen standardním algoritmem pro zlom odstavců, ovšem s vámi definovanou *šířkou*, která je v obou případech povinným argumentem. Neexistuje pro ni žádný automatický výpočet, musíte ji uvést. Výsledný box je součástí aktuálního řádku. Pomocí *zarovnání* lze stanovit, zda má být vůči okolnímu řádku zarovnán jeho horní (`t`) nebo spodní (`b`) okraj, implicitně je centrován.

pár slov
Dáme si do boxu.

```
Dáme si
\parbox[b]{3.7em}{pár slov do boxu}.
```

V obou případech lze za *zarovnání* přidat ještě dva volitelné parametry, které určí cílovou výšku boxu a svislé zarovnání obsahu v něm (hodnoty `t`, `b`, `c` nebo `s`).

27 Definice vlastních příkazů a prostředí

Hlavní předností LaTeXu je, že klíčové prvky sazby definuje logicky, nikoli vizuálně. Ve zdrojovém textu příkazem `\chapter` oznámíte, že zde začíná kapitola s daným názvem. Podobně prostředím `itemize` označíte seznam s odrážkami a příkazy `\item` jeho položky. Co tato informace znamená, jak má vypadat nadpis kapitoly či seznam s odrážkami a co všechno se má provést, to vše je definováno někde stranou.

Tento přístup je koncepční a umožňuje autorovi textu soustředit se na podstatné věci. Druhou jeho velkou výhodou je flexibilita. Pokud se rozhodnete změnit vzhled některé konstrukce, stačí příslušně upravit definici daného příkazu a změna se promítne do celého dokumentu.

Je velmi rozumné držet se tohoto přístupu i pro vlastní texty a připravit si příkazy pro specifické prvky, které se v nich vyskytují. Například v tomto textu hojně cituji různé příkazy. Připravil jsem si tedy pro ně příkaz `\cmd`, kterému jako argument předám jméno příkazu a on se postará o vše potřebné:

Příkazem \newcommand lze definovat vlastní příkazy.

```
Příkazem \cmd{newcommand} lze
definovat vlastní příkazy.
```

K vytvoření nového příkazu slouží příkaz `\newcommand`. Má dva povinné argumenty: jméno definovaného příkazu (včetně úvodního `\`) a jeho význam. Kdykoli později použijete definovaný příkaz, bude jeho výskyt v textu nahrazen významem podle definice. Řekněme, že bych si chtěl definovat zkratku `\ps` pro své jméno a příjmení:

```
\newcommand{\ps}{Pavel Satrapa}
```

Mohu samozřejmě přidat formátování a obecně jakékoli příkazy podle libosti. K jejich interpretaci dojde v okamžiku, kdy je příkaz použit, nikoli během jeho definice:

jsme na straně 65

```
\newcommand{\stranka}{\emph{jsme na
straně \thepage}} \stranka
```

Ve většině případů ale potřebujete příkazu předávat parametry. Pak se tvar jeho definice rozšíří o nepovinný argument udávající počet parametrů. Plný tvar tedy vypadá následovně:

`\newcommand{\`*jméno*`}[`*počet parametrů*`]{`*význam*`}`

Použití parametrů je jednoduché – jsou identifikovány pořadovými čísly od jedničky a kamkoli do *významu* chcete vložit hodnotu *n*-tého parametru, nasaďte `#`*n*. Při volání příkazu pak každý parametr uzavřete do složených závorek.

Příkaz \uv uzavírající svůj argument do uvozovek by se definoval takto:

```
\newcommand{\uv}[1]{„#1“}
```

O chlup složitější by byl \kontakt, který má vysázet kontaktní informace – jméno a za ním adresu v závorce a kurzívou. Jeho prvním parametrem je jméno, druhým adresa pro elektronickou poštu:

A použití:
Jiljí Hustý (*Jilji-Husty@tul.cz*)

```
\newcommand{\kontakt}[2]{#1 (\emph{#2})}
A~použití:\\
\kontakt{Jiljí Hustý}{Jilji-Husty@tul.cz}
```

\newcommand kontroluje, zda je požadované jméno příkazu dosud volné. Nemůže se stát, že byste nedopatřením předefinovali již existující příkaz. Pokud je toto vaším cílem, sáhněte po \renewcommand, kterým změníte význam již existujícího příkazu.

Analogicky lze definovat vlastní prostředí. K tomuto účelu slouží příkaz \newenvironment (resp. \renewenvironment):

\newenvironment{*jméno*}{*zahájení*}{*ukončení*}

Zahájení a *ukončení* obsahuje příkazy, které se mají provést v okamžiku, kdy prostředí začíná a končí. Pokud byste například psali učebnici a chtěli vizuálně oddělit příklady od okolního textu, mohli byste si zavést následující prostředí priklad:

Toto je text před příkladem.

Příklad: A zde máme text příkladu.

Pokračuje běžný text.

```
\newenvironment{priklad}
{\begin{quote}\textbf{Příklad:}}
{\end{quote}}
Toto je text před příkladem.
\begin{priklad}
A~zde máme text příkladu.
\end{priklad}
Pokračuje běžný text.
```

28 Čítače a délky

Vedle vlastních příkazů si můžete definovat i další konstrukce, které lze vhodně využít. Patří mezi ně čítače, které slouží k počítání kusů, jak ostatně napovídá jejich název. Existuje několik standardních čítačů definovaných přímo jako součást LaTeXu, například `page`, který obsahuje aktuální číslo stránky.

Chcete-li si definovat vlastní, použijte příkaz `\newcounter{`*jméno*`}`. Součástí definice může být i navázání nového čítače na již existující nadřízený čítač, které způsobí, že při posunu nadřízeného bude tento čítač resetován. Díky tomu lze například obrázky číslovat v rámci kapitol – kdykoli se zvýší číslo kapitoly, začínají se obrázky číslovat znovu od začátku. Jméno nadřízeného čítače se přidává na konec příkazu jako nepovinný parametr: `\newcounter{`*jméno*`}[`*nadřízený*`]`.

Jednoduché nastavení určité konkrétní hodnoty do čítače zajistí příkaz `\setcounter{`*čítač*`}{`*hodnota*`}`. O přičtení celého čísla k aktuální hodnotě se postará `\addtocounter{`*čítač*`}{`*hodnota*`}`.

Nejzajímavější cestu ke změně hodnoty ovšem představuje dvojice příkazů `\stepcounter{`*čítač*`}` a `\refstepcounter{`*čítač*`}`. Mají společný základ: zvětší hodnotu daného čítače o jedničku a resetují na nulu všechny jeho podřízené. `\refstepcounter` navíc učiní daný čítač aktuálním pro odkazy vytvářené příkazem `\ref`.

Když potřebujete vysázet aktuální hodnotu čítače, použijte `\the`*čítač*, kde *čítač* je jméno čítače, který chcete zobrazit.

Jako ukázku zkusím rozvinout výše definované prostředí o automatické číslování. Na čísla příkladů zavedu nový čítač `priklad`. Hodlám je číslovat v rámci sekcí, proto jej učiním podřízeným čítače `section` obsahujícího číslo aktuální sekce. Zahajující sekvence příkazů prostředí `prikaz` zvětší jeho aktuální hodnotu pomocí `\refstepcounter` a následně vysází číslo sekce a číslo příkladu v ní, podle obvyklých konvencí oddělené tečkou, tedy `\thesection.\thepriklad`:

Toto je text před příkladem.

Příklad 28.1: A zde máme text příkladu.

```
\newcounter{priklad}[section]
\newenvironment{priklad}
{\begin{quote}
 \refstepcounter{priklad}
 \textbf{Příklad
 \thesection.\thepriklad:}}
{\end{quote}}
Toto je text před příkladem.
\begin{priklad}
A~zde máme text příkladu.
\end{priklad}
```

Jestliže čítače uchovávají celá čísla, prostřednictvím délek lze ukládat a měnit údaje o rozměrech. LaTeX obsahuje celou řadu délkových parametrů, ovšem pro své konstrukce si samozřejmě můžete vytvořit vlastní. Použijte příkaz `\newlength{\jméno}`. Všimněte si zpětného lomítka před jménem – na rozdíl od čítačů se délky chovají podobně jako příkazy.

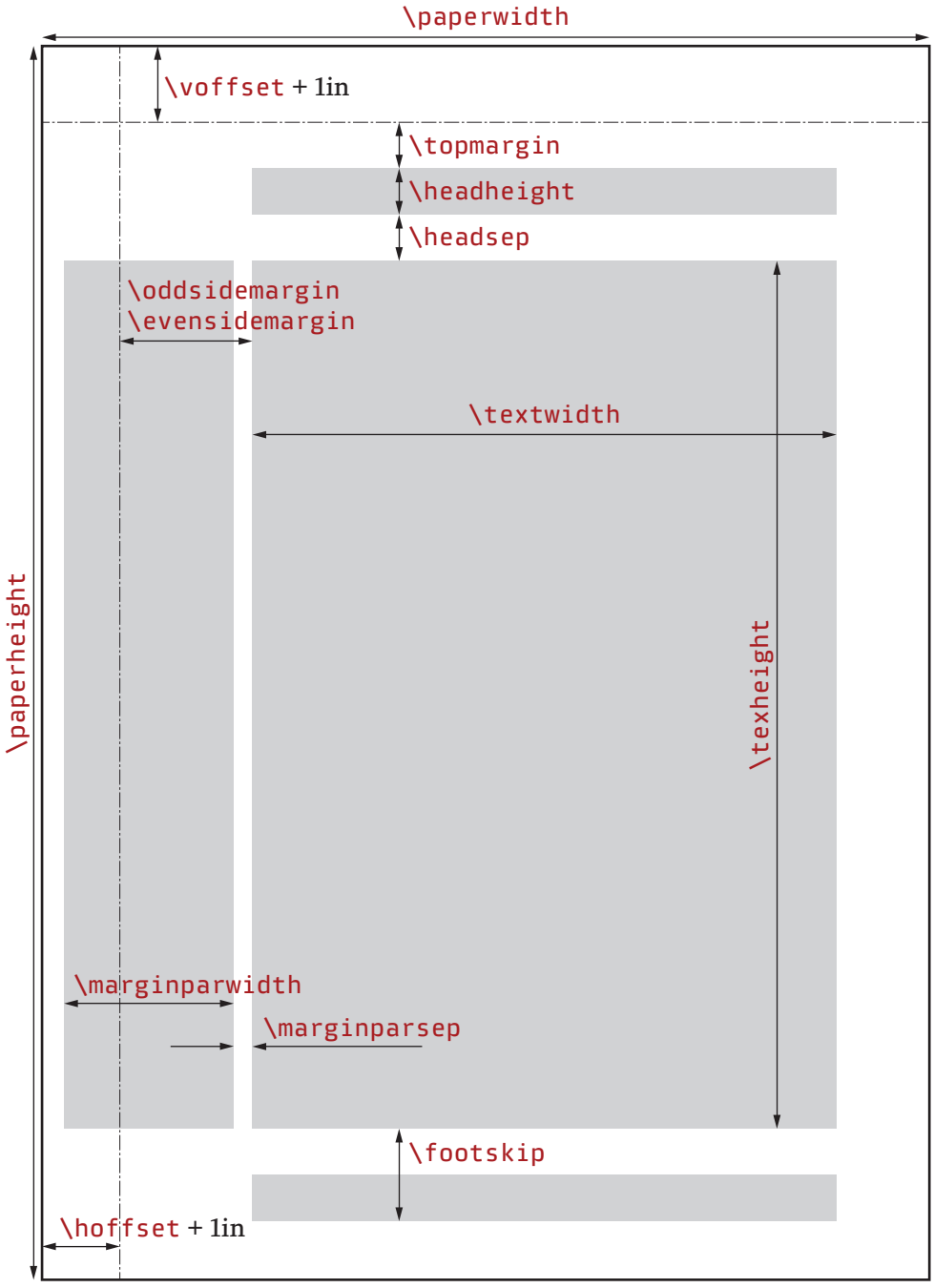
Hodnotu nastavíte příkazem `\setlength{\jméno}{rozměr}` a změníte ji pomocí `\addtolength{\jméno}{rozměr}`. Zajímavou variantou je `\settowidth{\jméno}{text}`, který si interně vysází *text*, změří jeho šířku a tu uloží do délky *\jméno*. Podobně fungují i příkazy `\settoheight` a `\settodepth`, jež do délky uloží výšku, resp. hloubku *textu*. Použít ji můžete ve tvaru *\jméno* všude tam, kde je očekáván délkový údaj:

raz dva
raz dva

```
\newlength{\delka}
\setlength{\delka}{1cm}
raz\hspace{\delka}dva\\
raz\hspace{2\delka}dva
```

Má-li být délkový údaj pružný, uveďte, o jakou vzdálenost jej lze roztáhnout (`plus`) a o kolik naopak stlačit (`minus`). Pokud některá z těchto hodnot má být nekonečná, uveďte jako její jednotku `fil`, `fill` nebo `filll` (podle stupně nekonečnosti). Délku o přirozené velikosti 12 pt, roztažitelnou o další 4 pt a stlačitelnou o 2 pt přiřadíte pomocí

```
\setlength{\delka}{12pt plus 4pt minus 2pt}
```



Obrázek 3: Rozměry na stránce

Skupina předdefinovaných délek určuje rozměry jednotlivých částí stránky. Jejich názorný přehled vidíte na obrázku 3. Celou stránkou lze snadno posunovat změnou `\hoffset` a `\voffset`, jež určují polohu dvou základních os, od kterých jsou odvozeny ostatní délky. K hodnotám `\hoffset` a `\voffset` se interně přičítá jeden palec. Chcete-li mít vodorovnou osu 1 cm pod horním okrajem stránky, nastavte

```
\setlength{\voffset}{-1.54cm}
```

Všechny stránkové rozměry jsou inicializovány na začátku zpracování podle třídy dokumentu a rozměrů papíru. V preambuli je následně můžete změnit podle libosti. Pokud byste se s nimi nechtěli mořit jednotlivě, sáhněte po balíku `geometry`.

29 Vkládání souborů

Texty většího rozsahu je rozumné rozložit do několika souborů, autorovi to ulehčí orientaci ve zdrojovém textu. LaTeX pro tento účel nabízí dvě konstrukce. Tou jednodušší z nich je příkaz `\input{`*soubor*`}`, který prostě do místa svého výskytu vloží obsah *souboru*. Vkládaný soubor může mít libovolnou příponu a kromě jeho vložení se neprovádějí žádné další kroky. Tato konstrukce se hodí zejména pro menší části, pomocí `\input` lze vkládat i definice v preambuli.

Sofistikovanější je `\include{`*soubor*`}`, který je vhodný zejména pro celé kapitoly či větší části textu. Také vloží obsah *souboru* do místa svého výskytu, ovšem před ním a po něm odstránkuje (provede `\clearpage`). Vkládaný soubor musí mít příponu *.tex*, kterou lze v argumentu příkazu vynechat. Hlavní výhodou `\include` je, že pomocí příkazu `\includeonly{`*seznam částí*`}` lze řídit, které soubory budou vloženy. Jakmile se ve zdrojovém textu vyskytne, budou vloženy jen ty *soubory*, které má ve svém seznamu (je oddělován čárkami).

Tím lze podstatně urychlit překlad rozsáhlé publikace – lze se soustředit jen na kapitolu, na které momentálně pracujete, a ostatní vůbec nepřekládat. Kompletní publikaci pak kdykoli vysázíte odstraněním (či lépe zakomentováním) příkazu `\includeonly`.

I když použijete `\includeonly`, řada konstrukcí LaTeXu se chová, jako by text byl kompletní. Vložené části mají správná čísla stránek, obsah

je úplný, fungují i odkazy do momentálně nevložených částí. Důvodem je, že pro každý soubor vložený příkazem `\include` si LaTeX vytváří pracovní soubor stejného jména s příponou *.aux*, do kterého si poznamenává klíčové informace o něm – začátky částí, návěští pro odkazy a podobně. Jestliže soubor není díky `\includeonly` vložen, načte alespoň odpovídající *.aux* a díky němu se dozví vše podstatné.

Soubor *.aux* bude pochopitelně aktualizován až při příštím překladu příslušné části. Není-li momentálně vložena, soubor se nemění a informace z něj nemusí odpovídat aktuálnímu stavu, pokud došlo například k posunu stránkování.

U rozsáhlejších publikací rozhodně doporučuji uložit každou kapitolu do samostatného souboru a tělo hlavního dokumentu minimalizovat:

```
%\includeonly{zaklady}
\begin{document}
...
\include{uvod}
\include{instalace}
\include{zaklady}
...
\end{document}
```

30 Sazba do sloupců

U běžných textů to nebývá obvyklé, ale některé typy publikací bývají uspořádány do sloupců. Typickým příkladem jsou noviny a časopisy, ale potkáte i dvousloupcové sborníky. V LaTeXu má tento způsob sazby na starosti rozšiřující balík `multicol`. Definuje prostředí `multicols`, které svůj obsah vysází do několika sloupců. Jejich počet je povinným argumentem prostředí.

Algoritmus se chová inteligentně, snaží se vyrovnávat výšky sloupců a bez problémů se vypořádá i s přechodem na další stránku. Pokud byste chtěli explicitně ukončit aktuální sloupec a zahájit další, máte k dispozici příkaz `\columnbreak`.

Několika parametry lze ovlivňovat vzhled vícesloupcového textu. Patří mezi ně především dél-

ky `\columnsep` stanovící mezeru mezi sloupci a `\columnseprule`, což je šířka oddělující čáry mezi nimi. Její barvu určíte příkazem `\columnseprulecolor`. Zahájení této sekce bylo sázeno následujícím způsobem:

```
\usepackage{multicol}
\setlength{\columnsep}{18pt}
\setlength{\columnseprule}{0.75pt}
\renewcommand{\columnseprulecolor}{\color{black!30}}
...
\begin{multicols}{2}
U~běžných textů ........ následujícím způsobem:
\end{multicols}
```

Mějte na paměti, že krátký řádek velmi omezuje možnosti sazby, obsahuje málo materiálu pro vyrovnání pravého okraje, což často vede k nedokonalostem. Proto to s počtem sloupců nepřehánějte, na stránce velikosti A4 jen vzácně dopadnou dobře více než dva sloupce.

31 Obtékané obrázky a tabulky

Obrázky a tabulky jsou obvykle sázeny v prostředí `figure` a `table`, které vždy přeruší tok textu. Jsou-li jejich rozměry malé, může výsledek působit nepatřičně. V takovém případě by bylo lepší, aby text obtékal kolem.

`l`	doleva
`r`	doprava
`i`	vnitřní okraj
`o`	vnější okraj

Tabulka 16: Zarovnání

Standardní LaTeX takové chování nenabízí, ale lze je doplnit rozšiřujícím balíkem `wrapfig`. Definuje dvě nová prostředí: `wrapfigure` pro obtékané obrázky a `wraptable` pro tabulky. Svými vlastnostmi se shodují, proto se budu věnovat jen druhému z nich. Pro `wrapfigure` platí totéž.

Prostředí má čtyři parametry, z toho dva nepovinné a dva povinné. Musí být uvedeny v následujícím pořadí:

`\begin{wraptable}[`*řádků*`]{`*umístění*`}[`*přesah*`]{`*šířka*`}`

Nepovinný počet *řádků* udává, kolik řádků má kolem materiálu obtékat. Obvykle se vynechává a počet řádků je určen automaticky. *Umístění* je jedno z písmen podle tabulky 16. Určuje, na kterou stranu má být obtékaný materiál umístěn. Pomocí *přesahu* lze nařídit, o kolik má obtékaný prvek přesahovat okraj stránky. A konečně povinná *šířka* určuje šířku obtékaného prostředí. Zadáte-li hodnotu `Opt`, určí se automaticky podle přirozené šířky svého obsahu.

Uvnitř prostředí `wraptable` lze používat stejné konstrukce, jako v běžném `table`. Všimněte si, že číslování nadpisu generovaného příkazem `\caption` plynule navazuje na číslování „obyčejných“ tabulek.

Obtékaná tabulka 16 byla vytvořena následujícím zdrojovým kódem:

```
\begin{wraptable}{o}[3em]{5cm}
\begin{center}
\begin{tabular}{ll}
\ff{l} & doleva\\
...
\end{tabular}
\end{center}
\caption{Zarovnání}
\label{wrap-umisteni}
\end{wraptable}
```

Na rozdíl od `table` toto prostředí není plovoucí. LaTeX pro ně nehledá vhodné místo, vysází je tam, kde je uvedete ve zdrojovém textu. Jen následující text bude kolem něj obtékat. Stejně jako v případě vícesloupcového textu dbejte na to, aby obtékající řádky byly dostatečně široké, jinak se dočkáte nepříliš pohledných výsledků.

Obtékaná prostředí mají řadu omezení. Nesmíte je použít uvnitř jiného prostředí. Kolem nich by měl obtékat jen hladký text bez komplikovanějších konstrukcí (například se špatně snáší s příkazy na členění textu). Také svislé umístění na stránce je čistě na vás – pokud obtékaný prvek zahájíte příliš pozdě, bez skrupulí přesáhne dolní okraj stránky. Sečteno a podtrženo: Používejte je raději méně a dobře si je hlídejte.

32 Otáčení a změna velikosti

Vrátím se ještě jednou k balíku `graphicx` z kapitoly o vkládání obrázků (strana 34). Vedle příkazů pro zařazení souborů v běžných grafických formátech nabízí i nástroje, jimiž lze změnit velikost a orientaci částí dokumentu.

Pro hrátky s velikostí je k dispozici hned několik prostředků. Za základní lze považovat příkaz `\scalebox{`*vodorovně*`}[`*svisle*`]{`*text*`}`, kde hodnoty *vodorovně* a *svisle* určují měřítko pro změnu velikosti ve vodorovném a svislém směru. Druhé se obvykle vynechává, což vede k použití stejného měřítka v obou směrech. Zadávají se jako reálná čísla, kde hodnota 1 představuje původní velikost.

Zajímavých efektů lze dosáhnout, pokud je některá z hodnot záporná – dochází pak k zrcadlovému obrácení textu. Pro zjednodušení balík zavádí příkaz `\reflectbox{`*text*`}`, který zajistí vodorovné zrcadlení a ve skutečnosti je zkratkou pro `\scalebox{-1}[1]{`*text*`}`.

velké Zrcadlení
velké 

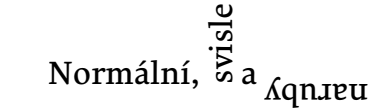
```
velké \scalebox{2}{Zrcadlení}\\
velké \scalebox{2}[-2]{Zrcadlení}
```

Nechcete-li úpravu velikosti zadávat měřítkem, máte k dispozici alternativní příkaz `\resizebox{`*v-rozměr*`}{`*s-rozměr*`}{`*text*`}`, kde zadáváte cílovou velikost ve vodorovném a svislém směru. Všimněte si, že oba údaje jsou povinné. Jeden z nich ale můžete nahradit vykřičníkem, pokud chcete, aby se dopočítal automaticky a nedošlo k deformaci.

velký a malý

```
\resizebox{3cm}{!}{velký} a
\resizebox{!}{6pt}{malý}
```

Otočení svého obsahu zajistí `\rotatebox[`*volby*`]{`*úhel*`}{`*text*`}`. Úhel otočení proti směru hodinových ručiček se zadává ve stupních.

Normální, svisle a 

```
Normální,
\rotatebox{90}{svisle} a
\rotatebox{180}{naruby}
```

Volbami lze především ovlivnit střed otáčení. Standardně se box otáčí kolem svého referenčního bodu, který leží na účaří. Proto se nápis obrácený vzhůru nohama ocitl pod řádkem. Střed otáčení určíte buď zjednodušeně volbou `origin`, jejíž hodnotou jsou až dva ze znaků `l` (vlevo), `r` (vpravo), `c` (uprostřed), `t` (nahoře), `b` (dole) a `B` (na účaří). Kromě toho lze střed zvolit v libovolném místě pomocí `x=`*délka* , `y=`*délka*.

není ʎqnɹɐu
jako naruby

```
není \rotatebox{180}{naruby}\\
jako \rotatebox[origin=c]{180}{naruby}
```

33 Barva

Práce s barvou je dalším prvkem, který leží za hranicemi schopností původního TEXu a LATEXu. Podobně jako v případě vkládané grafiky velmi záleží na schopnostech konkrétní implementace, nicméně ty nejběžnější barvu podporují a interní rozdíly mezi nimi odstíní balík `xcolor` jednotnou sadou příkazů.

První krok na cestě k obarvení dokumentu tedy zní

```
\usepackage{xcolor}
```

Ve volbách balíku se může objevit identifikace ovladače, kterou určíte konkrétní implementaci (a v důsledku toho interní příkazy pro řízení barev). Pokud je takový krok potřeba, je rozumnější nastavit v souboru *color.cfg* výchozí hodnotu pro celý systém. Konkrétní ovladač uvedený v dokumentu komplikuje jeho přenositelnost.

Další zajímavou volbou může být cílový barevný model, do kterého se mají barvy převádět. Balík jich podporuje celou řadu (`rgb`, `cmyk`, `gray`, `hsb`, `HTML` a další). Má-li dokument směřovat do tiskového stroje, doporučuji `cmyk`, pro obrazovky spíše `rgb`.

Třetí zajímavou skupinou voleb jsou vazby na další balíky. Za pozornost stojí především `table` pro barvení tabulek a `hyperref` pro on-line odkazy.

Základním příkazem pro změnu barvy textu je `\color`. Používá se ve tvaru `\color[`*model*`]{`*specifikace*`}`, kde *specifikace* vychází z použitého

barevného *modelu*. Ve většině případů se jedná o čárkami oddělovaný seznam hodnot od 0 do 1, které vyjadřují intenzitu příslušné barevné složky. Vzhledem k všudypřítomnosti webových barev stojí za zmínku model `HTML`, což je varianta `rgb`, ve které se ovšem místo tří čísel od 0 do 1 uvádí šestice číslic v šestnáctkové soustavě:

Barevný text zajistí příkaz `\color`…

```
\color[HTML]{005F00}Barevný text
zajistí příkaz \cmd{color}\ldots
```

Zadávat barvy po složkách není žádná velká zábava, zejména pokud se opakují. Je lepší používat jména: `\color{`*jméno*`}`. Balík definuje několik základních jmen a volbami `dvipsnames`, `svgnames` a `x11names` k nim lze přidat myriády dalších. Nejzajímavější ovšem je definovat si barvy vlastní pomocí

`\definecolor{`*jméno*`}{`*model*`}{`*specifikace*`}`

Pojmenovanou barvu lze navíc nanést s určitou hustotou – připojte za jméno vykřičník a číselnou hodnotu v rozmezí od 0 do 100, která udává, kolik procent barvy se má použít. Samotné jméno barvy je vlastně zkratkou za `barva!100`. Takových dvojic lze uvést několik (oddělují se opět vykřičníky) a barvy tak míchat:

oranžová,
jemná oranžová,
temná oranžová

```
\definecolor{oranz}{rgb}{1, 0.5, 0}
\color{oranz}oranžová,\\
\color{oranz!50}jemná oranžová,\\
\color{oranz!50!black!50}temná oranžová
```

Příkaz `\color` funguje jako přepínač. Změní barvu textu, která platí až do další změny nebo ukončení aktuální skupiny. Dáváte-li přednost příkazu s argumentem, použijte `\textcolor`, který má na konci navíc jeden argument obsahující text, na nějž má být barva uplatněna.

Box s barevným pozadím vytvoříte příkazem `\colorbox`, jehož prvním argumentem je barva pozadí a druhým obsah boxu. Chcete-li navíc přidat barevný rámeček kolem, sáhněte po

`\fcolorbox{`*barva rámečku*`}{`*barva pozadí*`}{`*obsah*`}`

Barvy lze zadávat jmény nebo kombinací [*model*]{*specifikace*} a pokud jsou obě ve stejném modelu, stačí jej uvést jen jednou.

Speciálním případem je pozadí celé stránky, které lze změnit pomocí `\pagecolor`. Jedná se o přepínač, který změní barvu stránek trvale, počínaje aktuální stránkou. Jeho účinek je globální a nelze jej omezit skupinou, musíte použít další `\pagecolor`.


```
\pagecolor{orange!15}
\colorbox{teal}{\color{white}první}
\fcolorbox{teal}{white}{druhý}
```

Barvení podkladu v tabulkách umožňuje balík `colortbl`, který vložíte použitím volby `table` balíku `xcolor`. Definuje příkazy `\cellcolor` pro obarvení jedné buňky, `\rowcolor` pro řádek a `\columncolor` pro sloupec. Mají přednost v uvedeném pořadí – buňkou lze přepsat barvu řádku i sloupce, řádkem sloupec. `\cellcolor` a `\rowcolor` se vyskytují přímo v tabulce, `\columncolor` patří do definice sloupců, kde je třeba jej uvést na začátku příslušného sloupce v konstrukci `>{}`. Nejlepší bude ilustrační příklad:

raz	dva	tři
čtyři	pět	šest
sedm	osm	devět

```
\begin{tabular}{>{\color{white}
\columncolor{teal}}lll}
raz & dva & tři \\
čtyři & pět & šest \\
\rowcolor{teal!40}
sedm & osm & \cellcolor{orange}devět
\end{tabular}
```

K obarvení čar ohraničujících pole tabulky slouží `\arrayrulecolor`. Smí se použít mimo tabulku, v jejím těle i v definici sloupců pomocí `>{}`. Nastaví vždy barvu pro všechny následující linie v tabulce. Díky `\doublerulesepcolor` lze obarvit i mezeru ve dvojité čáře.

Balík `xcolor` pak přidává ještě příkaz `\rowcolors`, který umožňuje pravidelné střídání barev v jednotlivých řádcích tabulky. Má tvar

`\rowcolors`{*kde začít*}{*barva lichých*}{*barva sudých*}

První argument obsahuje číslo řádku, na němž má střídání barev začít, další dva pak uvádějí použité barvy. Příkaz se musí vyskytovat ještě před začátkem tabulky.

34 Vytvoření PDF

Samotná sazba LaTeXem do formátu PDF nepředstavuje žádný problém. Buď rovnou použijete příslušnou implementaci (pdfLaTeX, XeLaTeX, LuaLaTeX), nebo standardní výstup ve formátu DVI zpracujete vhodným konvertorem (*dvipdf*, *dvipdfm*).

Zajímavější výzvou je využití schopností formátu PDF, především aktivace odkazů. K tomu slouží balík `hyperref`. Má desítky voleb pro nastavení různých vlastností, které buď můžete zadat při vložení balíku, nebo kdykoli později nastavit příkazy `\hypersetup`. Kromě toho lze definovat i implicitní chování balíku pro váš systém v konfiguračním souboru *hyperref.cfg*.

Přidáte-li do záhlaví dokumentu

```
\usepackage{hyperref}
```

chování řady konstrukcí se změní. Položky v obsahu, odkazy na jiné části textu, literaturu či poznámky se stanou aktivními a přesunou čtenáře na příslušné místo. U některých lze toto chování vypnout – konkrétně pro poznámky volbou `hyperfootnotes` a pro rejstřík pomocí `hyperindex`. Například deaktivaci poznámek pod čarou by zařídilo

```
\usepackage[hyperfootnotes=false]{hyperref}
```

Ve výchozím nastavení jsou odkazy zvýrazněny barevným rámečkem. Lze barevně odlišit různé typy odkazů, například pro běžné vnitřní odkazy (na části textu či obrázky) definuje barvu rámečku vlastnost `linkbordercolor`. Jejich přehled najdete v tabulce 17.

Barevné rámečky ale estétovo oko nepotěší. Za vhodnější považuji místo nich obarvit text odkazu, jak činím i v tomto dokumentu. Postará se o to volba `colorlinks` s hodnotou `true`, která zároveň vypne rámování. Stejně jako v případě rámečků, i texty lze obarvit různě v závislosti na typu daného odkazu. Příslušné vlastnosti jsou opět uvedeny v tabulce 17. Nepovažuji to za příliš šťastné. V preambuli tohoto dokumentu byste proto našli

odkaz na	*rámeček*	*text*
URL (externí cíl)	`urlbordercolor`	`urlcolor`
soubor	`filebordercolor`	`filecolor`
část textu (interní odkaz)	`linkbordercolor`	`linkcolor`
literaturu	`citebordercolor`	`citecolor`
menu Acrobatu	`menubordercolor`	`menucolor`
spouštěný program	`runbordercolor`	`runcolor`

Tabulka 17: Nastavení barev pro zvýraznění odkazů

```
\usepackage{hyperref}
\definecolor{Odkazy}{HTML}{005555}
\hypersetup{colorlinks=true, linkcolor=Odkazy,
            urlcolor=Odkazy, citecolor=Odkazy}
```

Je slušné vložit do PDF také metainformace o názvu dokumentu a jeho autorovi. Slouží k tomu vlastnosti `pdftitle` (implicitní hodnotou je jméno souboru) a `pdfauthor`:

```
\hypersetup{pdftitle={LaTeX pro pragmatiky}}
\hypersetup{pdfauthor={Pavel Satrapa}}
```

Kromě interních odkazů budete často potřebovat ještě nástroj, kterým byste se odkázali ven, například na webovou stránku. K tomu slouží příkaz `\href{`*URL*`}{`*text*`}`, kde *URL* je cílová adresa odkazu a *text* jeho viditelná podoba:

CSTUG – Československé sdružení uživatelů TEXu bylo založeno roku 1990.

```
\href{http://www.cstug.cz/}{CSTUG~--
Československé sdružení uživatelů
\TeX u} bylo založeno roku 1990.
```

V *URL* znaky #, & a ~ ztrácejí svůj speciální význam, adresy proto můžete psát v původním tvaru. Pokud se cílová adresa shoduje s textem, který chcete vysázet, můžete použít zjednodušený příkaz `\url{`*URL*`}`:

http://www.nti.tul.cz/~satrapa/

```
\url{http://www.nti.tul.cz/~satrapa/}
```

Abyste mohli snadněji dodržovat jednotný vzhled adres v dokumentu, máte k dispozici ještě `\nolinkurl{`*URL*`}`, který svůj argument vysází stejným způsobem jako `\url`, ale neučiní jej aktivním odkazem.

Balík `hyperref` obsahuje i příkazy, jimiž lze generovat PDF formuláře (`\TextField`, `\CheckBox` a další), ale to jsme už mimo dosah tohoto textu.

Reference

[GMR07] Michel Goossens, Frank Mittelbach, Sebastian Rahtz, Denis Roegel, Herbert Voß: *The LaTeX Graphics Companion*. 2nd edition, Addison-Wesley Professional, 2007

[GRG99] Michel Goossens, Sebastian Rahtz, Eitan M. Gurari, Ross Moore, Robert S. Sutor: *The LaTeX Web Companion*. Addison-Wesley Professional, 1999

[Knu86] Donald E. Knuth: *The TeXbook*. Addison-Wesley Professional, 1986

[Lam94] Leslie A. Lamport: *LaTeX: A Document Preparation System*. 2nd edition, Addison-Wesley Professional, 1994

[MiG04] Frank Mittelbach, Michel Goossens: *The LaTeX Companion*. 2nd edition, Addison-Wesley Professional, 2004

[Pec11] Martin Pecina: *Knihy a typografie*. Host, 2011

[Olš01] Petr Olšák: *TeXbook naruby*. 2. vydání, Konvoj, 2001

[Ryb02] Jiří Rybička: *LaTeX pro začátečníky*. 3. vydání, Konvoj, 2002

[Što08] František Štorm: *Eseje o typografii*. Revolver Revue, 2008

Užitečné adresy

www.ctan.org – archiv materiálů pro TeX, LaTeX a spol.

www.cstug.cz – Československé sdružení uživatelů TeXu

cstex@cs.felk.cvut.cz – elektronická konference o TeXu a typografii

www.tug.org – TeX Users Group

Index