OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

KAPITOLA 2

Typografický systém TEX

V této kapitole se budeme zabývat typografickým systémem TEX a především jeho nástavbou LATEX, ale jen v základních rysech. Struktura obsahu kapitoly je podobná jako v kapitole o MS Wordu – typografická pravidla již vysvětlovat nebudeme, ale ukážeme si, jak v LATEXu provést to, co jsme dělali v MS Wordu.

Čtenář se nemusí obávat přílišné náročnosti této problematiky. Jednotlivé postupy jsou popisovány jako jakási „kuchařka“ a ukazovány na příkladech. Je používán editor, který sice pracuje pouze s čistým textem, ale vyznačuje klíčová (důležitá) slova a téměř vše ze základů zde můžeme provádět klepáním na tlačítka panelů.

Narozdíl od předchozí kapitoly zde nenajdeme tolik úloh. Předpokládá se, že čtenář si znalosti procvičuje na uvedených příkladech.

2.1 Jak to všechno funguje

2.1.1 Princip překládaných dokumentů

MS Word a programy jemu podobné jsou označovány zkratkou WYSIWYG – *What You See Is What You Get*, tedy dostaneš to, co vidíš. Vidíme zformátované stránky s uplatněním stylů, zobrazenými tabulkami, obrázky apod., a totéž dostaneme, když dokument vytiskneme. Ono to až tak doslova někdy nebývá, ale to je zase trochu jiný problém. WYSIWYG programy mají hodně výhod z hlediska koncového uživatele (toho, kdo dokument vytváří), ale také hodně nevýhod plynoucích většinou z toho, že program musí pořád „myslet za uživatele“, někde uvnitř mít uloženy informace o tom, jak má co vypadat a pořád to tak zobrazovat, pamatovat si pozici v dokumentu a neustále „graficky“ reagovat na pokusy uživatele o změny v dokumentu.

Tohle všechno uhlídat a ještě k tomu nabízet možnost „uživatelsky přítulného“ přístupu ke všem možnostem nastavení formátů je velmi náročné a dá se to zvládnout jen buď vysokou spotřebou zdrojů počítače a složitostí prostředí programu, nebo omezením nabízených funkcí, nebo tolerancí občasných chyb.

Program TEX (čteme [*tech*]) a jeho nástavba LATEX (čteme [*latech*] nebo [*lejtech*]) jdou jinou cestou. Vlastně se nejedná o žádný editor, je to pouze sada překládacích programů, které ze zdrojového souboru s příponou TEX vyrobí cílový soubor s příponou DVI, PS nebo PDF. Zdrojový soubor můžeme vyrobit v prakticky jakémkoliv textovém editoru, který dokáže ukládat čistý text (třeba v Poznámkovém bloku), ale lepší je využít některý editor přímo k tomu určený, s tlačítky pro nejčastější akce.

Ve zdrojovém souboru vyznačujeme formátování a styly pomocí *příkazů*. Existuje například příkaz, který určitý blok písmen (slov) zformátuje jako kurzívu, nebo příkaz pro vysázení nadpisu určité úrovně. Jiné příkazy slouží k „provozním účelům“ – sdělujeme překládacímu programu, že se má použít česká znaková sada, jak se mají formátovat odstavce, jaký základní styl chceme pro dokument použít a kde přesně v souboru začíná a končí vlastní dokument.

Aby se nepopletly příkazy a samotný text, každý příkaz začíná znakem \ a jeho parametry (upřesňující informace) se dávají do složených {} (u povinných parametrů) nebo hranatých závorek [] (u nepovinných parametrů).

Práce v TEXu a LATEXu obnáší zhruba takový postup, jako když sedíme u počítače s někým jiným, koukáme mu přes rameno a diktujeme: *Napiš článek v češtině, kde nadpis bude „Postup výpočtu kvadratury kruhu“, autor budu já, datum tam dej dnešní (nevím, kterého je, zařiď to nějak), v abstraktu napiš větu „V článku se budeme věnovat různým postupům velmi zajímavého výpočtu kvadratury kruhu.“, v kapitole s názvem Úvod bude tento text: „Škoda že dosud tento problém nebyl“, teď vyznačeně „uspokojivě“, konec vyznačení, dále „vyřešen. Proto se nám zde otevírá poměrně široké pole působnosti“* atd.

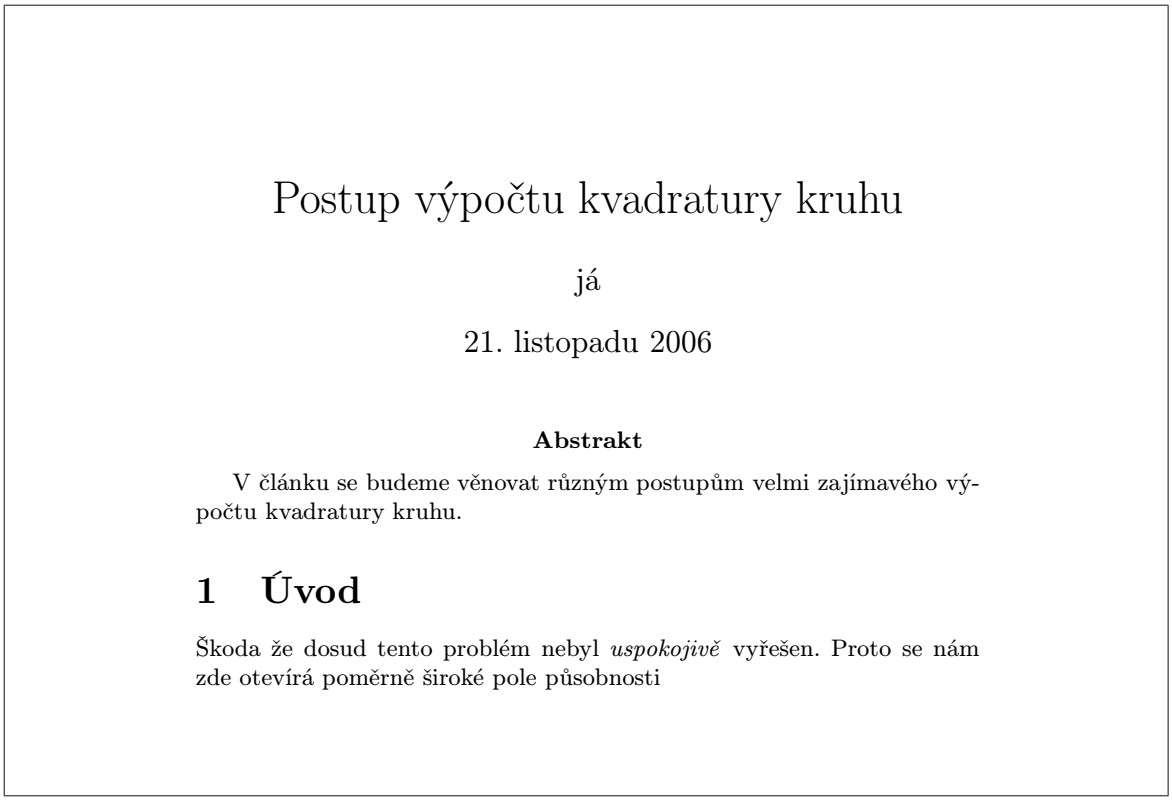
Výpis č. 2.1 [1]

```
\typdokumentu{článek}
\jazyk{čeština}

\název{Postup výpočtu kvadratury kruhu}
\autor{já}
\datum{\dnes}

\začátek_dokumentu

\tady_bude_titul

\abstrakt{V článku se budeme věnovat
různým postupům velmi zajímavého výpočtu
kvadratury kruhu.}

\kapitola{Úvod}

Škoda že dosud tento problém nebyl
\vyznač{uspokojivě} vyřešen. Proto se
nám zde otevírá poměrně široké pole
působnosti

\konec_dokumentu
```

```
\documentclass{article}
\usepackage{czech}

\title{Postup výpočtu kvadratury kruhu}
\author{já}
\date{\today}

\begin{document}

\maketitle

\abstract{V článku se budeme věnovat
různým postupům velmi zajímavého výpočtu
kvadratury kruhu.}

\section{Úvod}

Škoda že dosud tento problém nebyl
\emph{uspokojivě} vyřešen. Proto se
nám zde otevírá poměrně široké pole
působnosti

\end{document}
```

[1] Všimněte si, že u jednoduchých dokumentů jde vlastně jen o to – znát pár anglických slovíček.

Na výpisu 2.1 můžeme vidět způsob uvažování uživatele (v prvním sloupci) a vlastní zdrojový kód dokumentu, na obrázku 2.1 výsledný vygenerovaný dokument.



Postup výpočtu kvadratury kruhu

já

21. listopadu 2006

Abstrakt

V článku se budeme věnovat různým postupům velmi zajímavého výpočtu kvadratury kruhu.

1 Úvod

Škoda že dosud tento problém nebyl *uspokojivě* vyřešen. Proto se nám zde otevírá poměrně široké pole působnosti

Obrázek 2.1: Ukázka jednoduchého dokumentu

Jak vidíme, nemusíme vůbec myslet na typ, řez a velikost písma různých částí titulu, formát abstraktu, číslování kapitol (pak není problém kapitoly „přehazovat“), způsob formátování vyznačeného textu – vše je nastaveno podle typografických pravidel. Prakticky do čehokoliv je možné zasahovat, ale většinou to nebývá potřeba. Problém by snad mohl být se zapamatováním příkazů (to nás však nemusí trápit, pokud máme vhodný editor) a s tím, že u příkazů je potřeba dávat pozor na velká a malá písmena – když místo `\date` napíšeme `\Date`, je to chyba.

Používáme zde dva různé pojmy – TEX a LATEX. Jaký je mezi nimi rozdíl? Jde především o různé sady příkazů. LATEX je nástavbou TEXu, která má zjednodušit vytváření běžných typů dokumentů, a tak kromě původních příkazů TEXu máme k dispozici hodně dalších, které bychom jinak museli tvořit složitě pomocí sekvencí původních příkazů. V textu dále bude používán pouze LATEX.

TEX (a LATEX) není jenom jeden. Existují tzv. TEXovské distribuce, což je vždy sada samotného TEXu, pomocných nástrojů a především mnoha rozšiřujících *balíčků*[2].

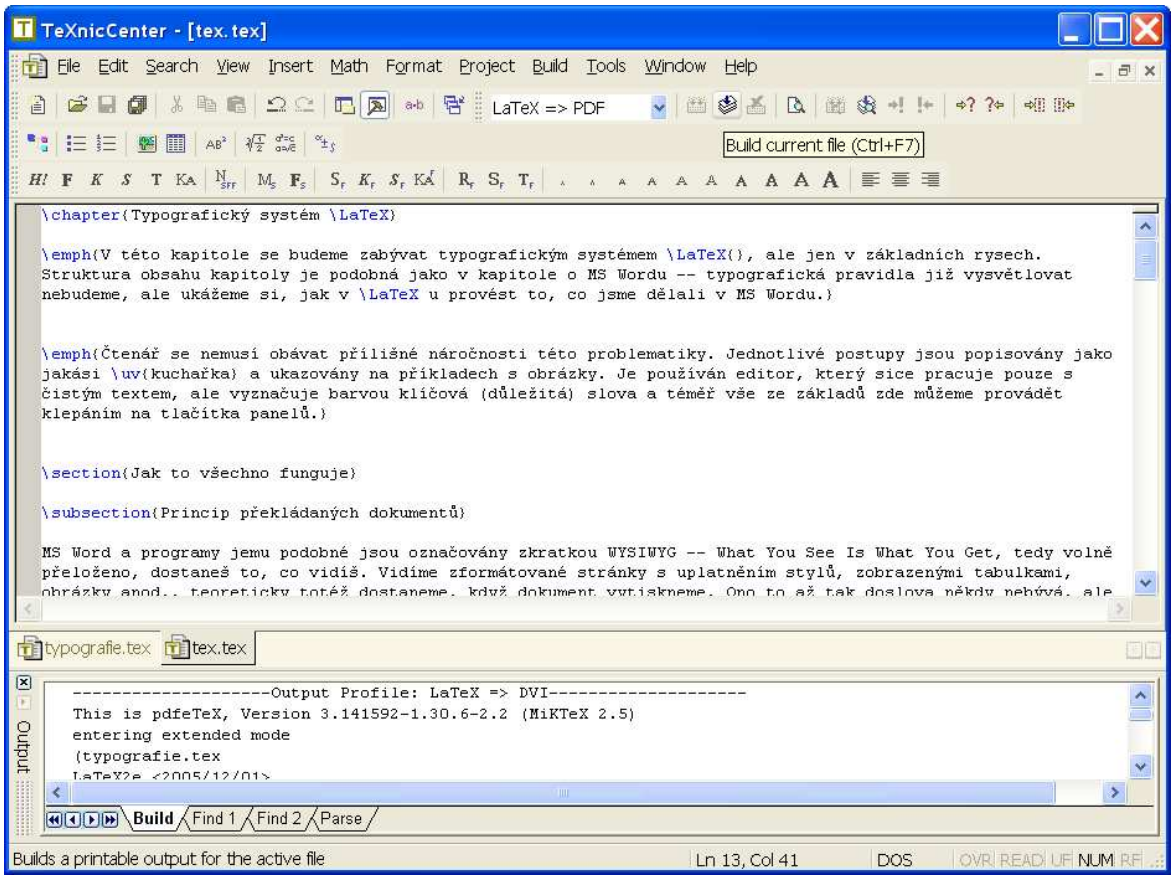
[2]Třeba balíček `czech` pro české prostředí, další balíčky s přednastavenými styly pro různé typy dokumentů, používání barev, fonty, vlastní definování záhlaví a zápatí stránky, speciální symboly pro různé vědní obory, atd.

U nás jsou pro Windows zřejmě nejoblíbenější distribuce MikTEX a TEXLive, na Unixových systémech je hodně používaný eTEX.

2.1.2 Jak vytvořit zdrojový soubor

Zdrojový soubor, jak je výše uvedeno, můžeme tvořit v téměř jakémkoliv textovém editoru. Ideální je použít takový editor, který má v sobě (především v menu a tlačítkách) zabudovánu podporu práce v TEXu a LATEXu.

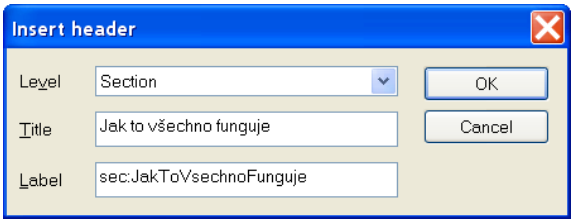
Takových programů je poměrně hodně – namátkou TEXnicCenter, TEXShell, TEXEdit, WinShell, WinEdt, ToolsCenter, EPM, Crimson Editor, PSPad, TEXMaker, MED, LyX, . . . Jeden z mnoha seznamů oblíbených editorů lze najít na [T30]. Každý si může vybrat ten, který mu nejvíc vyhovuje, většinou jde o volně šiřitelné programy (až na některé – například WinEdt je shareware, tedy oproti plné verzi má některá omezení), najdeme je většinou na internetu.



Obrázek 2.2: Prostředí programu TEXnicCenter

Na obrázku 2.2 vidíme prostředí jednoho z editorů, TEXnicCenter. Na horním okraji okna máme při ruce různé nástroje, které bychom mohli potřebovat (lze měnit podle vlastních potřeb), další jsou přístupné v menu. Dále máme k dispozici dialogová okna, do kterých můžeme vyplňovat údaje potřebné k automatickému

vygenerování některého příkazu. Ukázka jednoho takového dialogového okna pro automatické vložení příkazu generujícího nadpis je na obrázku 2.3 na str. 42.



Obrázek 2.3: Dialog pro vytvoření nadpisu a „záložky“, na kterou se lze odkazovat

2.2 Začínáme

Předpokládejme, že máme nainstalovánu některou distribuci TEXu a vhodný editor. Zde budeme používat MikTEX verze 2.5 a editor TEXnicCenter verze 7.01, ale to, co budeme dále probírat, je stejné téměř ve všech jiných používaných distribucích a v některých podobných editorech.

Samotná instalace není zrovna triviální (především nastavení češtiny), stručný postup instalace těchto verzí je na [P14].

Tak tedy máme spuštěn náš editor. Co teď? Vlastně nic, můžeme přímo začít psát náš text. Do nového souboru napíšeme třeba větu, kterou se obvykle testuje správné nastavení češtiny – když jsou všechna písmena správně „očárkovaná“ a „oháčkovaná“, je nastavení vpořádku (uvedeno např. v [T31]). Samotná věta moc smyslu nedává:

Příliš žluťoučký kůň pěl ďábelské ódy.

Ještě než pošleme tento soubor dále na zpracování, musíme provést ještě jednu věc – přidat příkazy, které LATEXu řeknou, o jaký typ dokumentu se jedná, že v souboru jsou znaky české abecedy a kde začíná a končí vlastní text dokumentu. Celý soubor bude vypadat tak jako ve výpisu 2.2.

Výpis č. 2.2

```
\documentclass{article}
\usepackage{czech}

\begin{document}
Příliš žluťoučký kůň pěl ďábelské ódy.
\end{document}
```

Část zdroje mezi příkazy `\begin{document}` a `\end{document}` se nazývá *tělo dokumentu*, část před příkazem `\begin{document}` se nazývá *hlavička dokumentu* neboli *preambule*. Do preambule se umísťují především informace o tom, jaké balíčky se mají načíst a jak má být co formátováno, tedy krátce řečeno, určujeme zde styl dokumentu.

Náš soubor je už „přeložitelný“. Uložíme ho s příponou TEX (to je přípona zdrojových souborů dokumentů v TEXu a LATEXu) a vybereme si formát a příponu pro výsledný dokument. Máme tyto možnosti:

PDF – tuto příponu už známe. PDF dokument zde můžeme vygenerovat bez jakýchkoliv dalších pomocných programů, narozdíl od MS Wordu. Soubor lze vytvořit přímo nebo přes následující formát PS[3]. Pro jeho prohlížení můžeme použít třeba volně šiřitelný program *Adobe Reader* (ke stáhnutí na [P11]).

PS – PostScript je formát příbuzný formátu PDF (dokonce má stejné tvůrce), jen je starší a soubory bývají většinou větší než u téhož dokumentu ve formátu PDF. Je vhodný zejména pro tisk, proto v tomto formátu máme dokumenty, které se především tisknou. Pro prohlížení existuje například program *GSView*.

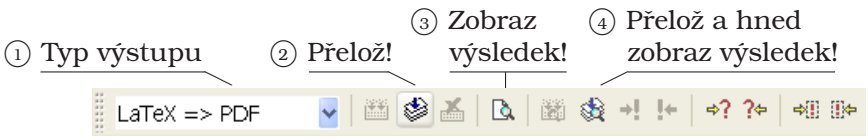
Oproti Adobe Readeru tento prohlížeč zobrazený soubor po načtení zavře a pravidelně ve velmi krátkých intervalech kontroluje, zda v souboru nedošlo ke změnám. Pro uživatele to znamená, že před dalším překladem mohou nechat dokument zobrazený (nezavírat) a navíc mají jistotu, že když se po následujícím překladu přepnou do GSView, vidí dokument v „čerstvém“ stavu, změny jsou aplikovány.

DVI – je to zkratka z *Device Independent*, tedy „nezávislý na zařízení“. Do tohoto formátu se soubor překládá nejrychleji, je to původní cílový formát TEXu. Obvyklým prohlížečem je *Yap* (*Yet Another Preview*, je součástí distribuce MikTEX).

Formát přejímá výhody formátu PS (před dalším překladem není třeba výsledný soubor zavírat a jsou hlídány změny), navíc má výhodu rychlosti překladu. Nevýhodou je menší rozšířenost tohoto formátu, když soubor s příponou DVI zveřejníme na internetu nebo někomu pošleme, není moc velká pravděpodobnost, že adresát bude vědět, co s ním.

Kdy který formát použít? Když se chceme na výsledek podívat během práce na dokumentu, je nejlepší přeložit soubor do DVI a podívat se na dokument v Yapu. Je to nejrychlejší a s použitím klávesové zkratky Alt + Tab máme téměř „WYSIWYG“ náhled. Hotový dokument pak zveřejňujeme v PS nebo PDF (nebo v obou formátech).

V programu TEXnicCenter používáme pro cokoliv souvisejícího s překladem panel nástrojů zobrazený na obrázku 2.4.



Obrázek 2.4: Panel nástrojů s nástroji pro překlad dokumentu

Takže naším úkolem je nyní vybrat si typ výstupu v ① (v našem případě překlad do DVI) a použít tlačítko ② nebo ④.

Prohlížeč Yap má oproti jiným ještě jednu důležitou výhodu – když v něm poklepeme myší na některou část dokumentu, jsme automaticky přeneseni na dané místo ve zdroji dokumentu. U tak krátkého dokumentu, jaký jsme vytvořili, to nemá moc velký význam, ale u dokumentu zabírajícího několik desítek stránek případně

[3] Přímé vytvoření umožňuje lépe pracovat se stránkou a jinak obcházet omezení daná formátem PS, vytvoření přes PS obvykle vytvoří menší soubor.

se zdrojem rozděleným do více souborů se to rozhodně hodí. Pokud tato vlastnost nepracuje jak má, můžeme ji nastavit v menu Yapu View → Options, záložka Inverse DVI Search (obvykle stačí jen klepnout myší na již vložený záznam, nemusíme vkládat nový).

2.3 Délkové jednotky

V kapitole jsme se na straně I seznámili s pojmem *kuželka* a určili *stupeň písma* jako výšku kuželky v typografických bodech. Z tohoto pojmu teď odvodíme další pojmy:

čtverčík (značíme `em`) je míra (tj. délka) odpovídající výšce kuželky (stupni), je to přibližně výška písmene M (od toho také zkratka),

půlčtverčík (značíme `ex`) je polovina čtverčíku, odpovídá přibližně výšce písmene x.

Pro délky můžeme využít přímo hodnoty `em` a `ex` (například při nastavování vlastností odstavce – tyto délky se mění se změnou velikosti písma a odstavec pak vypadá „stejně“), ale používají se i další jednotky – `pt` (anglosaský typografický bod), `in` (palec), `cm`, `mm`, ... Mezi číslo a jednotku se nepíše mezera.

Například stupeň písma bývá `10pt`, `11pt`, `12pt`, odsazení prvního řádku odstavce `2em`, mezi dvěma odstavci, kde má čtenář na papíru něco doplnit, je třeba mezera `4.5cm` nebo `.8em` (před tečkou nemusí být žádná číslice – dosadí se 0).

2.4 Písmo

Ještě než se budeme věnovat možnostem nastavení písma, musíme si ujasnit jednu důležitou věc – používání lomených závorek. Lomené závorky `{}` určují *skupinu*, skupiny můžeme libovolně přidávat a vnořovat. Kdekoliv máme možnost stanovit, že určitá vlastnost včetně vlastností písma bude platit od místa, kde ji určíme, až do nejbližšího konce některé skupiny.

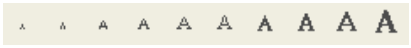
Skupinu určují také některé příkazy, především `\begin` a `\end` (oblast ohraničená těmito dvěma příkazy se nazývá *prostředí*) – například „hlavní skupinou“ je tělo programu ohraničené příkazy `\begin{document}` a `\end{document}`.

2.4.1 Stupeň písma

Stupeň písma můžeme nastavit pro celý dokument hned na začátku, když TEXu sdělujeme, jaký typ dokumentu chceme vyrobit:

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

Stupeň písma zde zadáváme v nepovinném parametru (je v hranatých závorkách), a to v typografických bodech (zkratka `pt`). Kolem závorek příkazu nepíšeme mezeru, příkaz musí být „kompaktní“.



Obrázek 2.5: Panel nástrojů pro změnu stupně písma

Příkaz	*Velikost*
`\tiny`	velmi velmi malé písmo
`\scriptsize`	velmi malé písmo
`\footnotesize`	malé písmo
`\small`	menší písmo
`\normalsize`	normální velikost písma
`\large`	větší písmo
`\Large`	ještě větší písmo
`\LARGE`	velké písmo
`\huge`	dost velké písmo
`\Huge`	velmi velké písmo

Tabulka 2.1: Příkazy pro změnu stupně písma

Při psaní občas také chceme změnit stupeň písma. To provádíme buď na panelu podle obrázku 2.5 nebo příkazy podle tabulky 2.1 na straně 45.

Tyto příkazy nemají žádný parametr, jejich platnost je od místa, kde je napíšeme, do nejbližšího konce skupiny. Odvíjejí se od základního stupně písma, který jsme mohli určit v parametru příkazu `\documentclass` – tomu stupni odpovídá příkaz `\normalsize`.

Základní stupeň písma dále můžeme (včetně řádkování) určit přímo příkazem `\fontsize`. Informace o tomto příkazu získáme v nápovědě nebo v [T31].

Příklad 2.1 Na následující ukázce vidíme, jak na čtenáře může působit častá změna stupně písma.

Výpis č. 2.3

```
\small Pokud chceme \normalsize vytvořit
{\Large esteticky zajímavý dokument}, hodně
\footnotesize často měníme velikosti písma.
\normalsize Pro čtenáře to je moc dobrý
způsob, jak si nacvičit {\LARGE ničení}
dokumentu na jakýkoliv způsob.
```

Pokud chceme vytvořit esteticky zajímavý dokument, hodně často měníme velikosti písma. Pro čtenáře to je moc dobrý způsob, jak si nacvičit ničení dokumentu na jakýkoliv způsob.

2.4.2 Nastavení základních vlastností písma

Řez (*shape*), tmavost (*series*) a rodinu (*family*) písma nastavujeme jednoduše – buď použijeme tlačítka na hlavním panelu (viz obrázek 2.6) nebo přímo příkazy.

Obrázek 2.6: Panel nástrojů pro volbu řezu, tmavosti a rodiny písma

Často nám úplně stačí příkaz `\emph{vyznačený text}` pro základní vyznačení (je přednastaven na kurzívu, i když to lze změnit), kterého jsme si mohli všimnout už ve výpisu 2.1 na straně 39 – použijeme buď přímo tento příkaz nebo tlačítko H! (podle obrázku 2.6 tlačítko ⑤). Příkaz lze také vnořovat, pak dostaneme text vyznačený ve vyznačeném textu, což představuje normální řez.

V tabulce 2.2 jsou vypsány různé příkazy nastavení řezů, tmavosti a rodiny písma, a to nejdřív klasickým příkazem, jehož parametrem je text pro formátování, a pak s použitím skupiny – příkaz nemusí být na začátku skupiny, ale kdekoliv v ní, pak platí od místa uvedení příkazu do konce skupiny – nejbližšího znaku }.

Řez	*Příkaz*	*Tlačítko*
kurzíva (ithalic)	`\textit{kurzíva}`	K ⑦
	`{\itshape kurzíva}`	K_r ⑮
skloněné písmo (slanted)	`\textsl{skloněné písmo}`	S ⑧
	`{\slshape skloněné písmo}`	S_r ⑯
KAPITÁLKY (small caps)	`\textsc{kapitálky}`	KA ⑩
	`{\scshape kapitálky}`	KA^r ⑰
vzpřímené (vertical)	`\textup{vzpřímené}`	
	`{\upshape vzpřímené}`	S_r ⑭

Tmavost	*Příkaz*	*Tlačítko*
polotučné písmo (medium)	`\textmd{polotučné písmo}`	
	`{\mdseries popotučné písmo}`	M_s ⑫
tučné písmo (bold)	`\textbf{tučné písmo}`	F ⑥
	`{\bfseries tučné písmo}`	F_s ⑬

Rodina	*Příkaz*	*Tlačítko*
patkové (serif, roman)	`{\rmfamily patkové}`	R_r ⑱
bezpatkové (sans-serif)	`{\sffamily bezpatkové}`	S_r ⑲
`neproporcionální` (typewriter)	`\texttt{neproporcionální}`	T ⑨
	`{\ttfamily neproporcionální}`	T_r ⑳

Tabulka 2.2: Příkazy pro změnu řezu, tmavosti a rodiny písma

Jak vidíme, některé příkazy jakoby neměly na písmo žádný vliv. To je proto, že například vzpřímené patkové písmo je výchozí, a dále příkaz pro polotučné písmo ve většině typů písem odpovídá normální tmavosti.

Pokud chceme zrušit všechny nastavené řezy, použijeme příkaz `\upshape` (tlačítko ⑭ – S_r), když chceme zrušit nejen nastavení řezu, ale i tmavost nebo neproporcionalitu, použijeme příkaz `\normalfont` (tlačítko ⑪ – N_{sff}).

LaTeX má také speciální způsob formátování matematických výrazů (typograficky správný). Zde se běžná velká i malá písmena abecedy sázejí skloněná nebo kurzívou a symboly, které nejsou písmeny (včetně čísel), vzpřímeným písmem, mezery mezi symboly jsou jiné než v běžném textu. Pro řízení formátu znaků zde máme k dispozici příkazy `\mathrm`, `\mathsf`, `\mathbf`, `\mathcal`, `\mathnormal`, atd.

Příklad 2.2 Ukážeme si způsob použití výše uvedených příkazů a možnosti jejich kombinace.

Výpis č. 2.4

```
\textsc{Různé} způsoby \textbf{vyznačení} je
dobré znát, používáme jak \textit{tlačítka},
tak i přímo \textit{příkazy}.

Pro většinu vyznačení existuje možnost
\textbf{dát vyznačovaný text do \itshape
parametru příkazu} a druhá možnost
{\bfseries s použitím \itshape skupiny}.

\emph{Musíme dávat pozor, kde je vlastně
\emph{konec} té skupiny, ve které chceme
mít dané vyznačení!}

Příkazy \texttt{můžeme \bfseries
kombinovat}, ale {\sffamily ne vždy výsledek
vypadá \itshape pěkně}.

{\bfseries\slshape Pozor na mezery,
\normalfont zvláště u příkazů využívajících
skupiny!}
```

RŮZNÉ způsoby **vyznačení** je dobré znát, používáme jak *tlačítka*, tak i přímo *příkazy*.

Pro většinu vyznačení existuje možnost dát **vyznačovaný text do *parametru příkazu*** a druhá možnost **s použitím *skupiny***.

Musíme dávat pozor, kde je vlastně konec *té skupiny, ve které chceme mít dané vyznačení!*

Příkazy `můžeme` **`kombinovat`**, ale ne vždy výsledek vypadá *pěkně*.

Pozor na mezery, zvláště u příkazů využívajících skupiny!

Za příkaz využívající hranice skupiny vždy dáváme buď mezeru nebo některé interpunkční znaménko (tečka, čárka, pomlčka, apod.), abychom tento příkaz oddělili od následujícího slova.

Před takový příkaz bychom měli dát mezeru, protože mezera (i více mezer) za příkazem se bere jako oddělovač příkazu a textu a tedy se do dokumentu nezahrnuje – tedy například `počet\itshape ` žáků se vysází jako počet*žáků*.

2.4.3 Fonty

Jak víme z předchozího textu, můžeme žádat jakékoliv změny mezi proporcionálním, patkovým a bezpatkovým písmem bez nutnosti „ruční“ změny fontu. Někdy ale potřebujeme nastavovat fonty přímo, tedy určovat typ písma dokumentu.

Narozdíl od programů typu MS Word, které využívají fonty instalované v operačním systému, nemáme v TeXu takový výběr, používáme pouze fonty dodané přímo s TeXem a těch roznodně není tolik. Je to především z toho důvodu, že systém obvykle nebývá používán pro sazbu reklamních tiskovin, ale spíše odbornějších článků a knih.

Nevýhodou je tedy menší výběr, výhodou ale snadnější přenositelnost – fonty se načítají přímo do dokumentu, proto se nemůže stát, že když je dokument otevřen na jiném počítači, fonty se nezobrazí správně.

Z typografických důvodů není moc vhodné měnit fonty v různých částech dokumentu, je zvykem mít nastaven jeden a tentýž font od začátku na celý dokument. Zde si ukážeme jednoduchý způsob nastavení fontu na začátku dokumentu.

Fonty se nacházejí v balíčcích, stejně jako čeština nebo různá nastavení stylů, proto je budeme načítat stejně jako ve výpisech 2.1 a 2.2 příkazem `\usepackage`. Standardně je nastaven font *Computer Modern*. Můžeme vyzkoušet další fonty:

- palatino,
- bookman,
- helvetica,
- optima,
- times,
- garamond,
- utopia,
- avantgar (pismo AvantGarde),
- zapfchan,
- ...

Například když si vybereme první možnost, font *palatino*, napíšeme do preambule někam hned za `\usepackage{czech}` příkaz `\usepackage{palatino}`, tak jak vidíme na výpisu číslo 2.5.

Každý z těchto fontů má trochu jiné vlastnosti a možnosti použití, proto je dobré alespoň několik vyzkoušet, než si definitivně vybereme. Některé fonty nejsou definovány pro některé řezy nebo tučnost a při překladu je daná vlastnost buď pozměněna na základní, nebo nahrazena z jiného fontu.

Výpis č. 2.5

```
\documentclass{article}
\usepackage{czech}
\usepackage{palatino}

\begin{document}
Toto je ukázka dokumentu, kde používáme font
\emph{palatino}.
Tento font má zajímavější kresbu než výchozí
\emph{Computer Modern}, je výraznější.

Předtím jsme byli pořád ve stejném odstavci,
i když jsme po první větě zařádkovali,
ale teď po druhé větě jsme vložili prázdný
řádek, proto jsme vytvořili další odstavec.
\end{document}
```

Jiný způsob nastavení fontů spolu s dalšími vlastnostmi je pomocí příkazů `\usefont`, `\fontencoding`, `\fontfamily`, `\fontseries`, `\fontshape`, `\fontsize`, těmito postupy se však zde nebudeme zabývat – jsou k nalezení v nápovědě nebo třeba v [T31, str. 153].

2.4.4 Zvláštní znaky a sekvence znaků

Komentáře jsou části textu, které se ve výsledném dokumentu neobjeví. Používáme je pro vlastní okomentování textu, například informaci o tom, že na určitém místě ještě chceme něco doplnit či změnit, opticky oddělujeme různé části textu nebo potřebujeme okomentovat příkaz, o kterém si myslíme, že v budoucnu budeme mít problémy s jeho pochopením. Stačí napsat znak % (procento) a vše za ním až do konce řádku bude bráno jako komentář.

Spojovník píšeme jednoduše klávesou s tímto znakem: `bude-li`, `česko-anglický slovník`.

Pomlčka se zadává jako dva spojovníky za sebou bez mezery, tedy píšeme `10--15 minut`, `napsala -- pro upřesnění`, `zápas Sparta--Slavia`. Dlouhou pomlčku pro anglické texty píšeme třemi spojovníky `---` (při obklopení mezerami).

Nezlomitelná mezera taková, které je dovoleno se případně rozšiřovat, se nahrazuje vlnovkou ~, například `u~domu`, `s~mámou`.

Pevná mezera (musí zůstat malá, při posouvání slov pro rovný pravý okraj textu se nesmí rozšiřovat) se zadává příkazem `\,`, například `12\,450\,832`, `10\,kg`. Často se místo ní používá nezlomitelná mezera (~).

Vybarvený obdélník ▬ se sází příkazem `\rule{šířka}{výška}` (zde jsou použity parametry `\rule{1em}{1ex}`).

Pokud zadáme některý rozměr nulový, i tak se „zabere“ místo, třebaže obdélník neuvidíme. Toho se často využívá při „nenápadném“ odsazení objektu zleva (`\rule{odsazení}{0pt}`) nebo shora (`\rule{0pt}{odsazení}`). V nepovinném parametru můžeme zadat také posun nahoru nebo dolů, takto se dá například vytvořit mezera pod textem (například `\rule[-2em]{0pt}{1pt}`).

Dlouhé horizontální mezery se zadávají příkazy `\quad` (čtverčíková) a `\qquad` (dvoučtverčíková).

Vertikální mezery jsou `\smallskip`, `\medskip` a `\bigskip` pro malou, střední a velkou mezeru mezi odstavci (například místo, kde se má něco doplnit „ručně“), s udáním délky `\vspace{délka}` – například `\vspace{2em}`.

Pružné mezery jsou `\vfill` a `\hfill`. Horizontální mezera se může použít například pro zarovnání části textu k pravému okraji (třeba číslo stránky v obsahu). Někdy je potřeba před a za tento příkaz vložit příkaz `\rule` s nulovou šířkou: `\rule{0pt}{1pt}\hfill\rule{0pt}{1pt}`.

Pružné čáry se sázejí příkazy `\hrulefill` (plná čára) a `\dotfill` (tečkovaná).

Trojtečku vyrobí příkaz `\dots`, za tímto příkazem píšeme buď některé interpunkční znaménko (třeba čárku, ale určitě ne další tečku!) a nebo, pokud má následovat mezera, musíme napsat prázdnou skupinu, aby mezera „nezmizela“, tedy píšeme `\dots{}`.

Uvozovky píšeme příkazem `\uv{text v uvozovkách}` (výsledkem je „text v uvozovkách“), úhlové jednoduše dvojicemi znaků `>>text v uvozovkách<<` (výsledkem je »text v uvozovkách«).

Procento sice najdeme na klávesnici, ale musíme použít příkaz `\%`, protože samotný znak procenta je vyhrazen pro komentáře. Píšeme `10\,\%` nebo `10~\%`.

TEX, LATEX, LATEX 2ε vysázíme příkazy `\TeX`, `\LaTeX`, `\LaTeXe` (to je označení zatím poslední verze systému LATEX, čteme v češtině [*latech dvě é*]).

Stejně jako u jiných příkazů, pokud má následovat mezera, musíme za příkaz napsat prázdnou skupinu – `\TeX{}`. V případě skloňování a ohýbání slov toho využijeme – například slovo TEXový vysázíme příkazem `\TeX ový`.

Slitky a podřezávání (viz strana I) TEX dělá automaticky. Pokud se nám zrovna toto chování nehodí, vložíme mezi dva znaky, které nechceme „spojit“, prázdnou skupinu (bez mezery!) – například místo `fi` bude `f{}i` a místo fi získáme fi.

Promile, stupeň a další symboly píšeme příkazy, jejichž seznamy najdeme i na internetu, například ve zdrojích [T33] nebo [T34]. Někdy je třeba v preambuli dokumentu před příkazem `\begin{document}`) uvést balíček, ve kterém jsou tyto příkazy určeny, například když je v preambuli dokumentu `\usepackage{textcomp}`, můžeme používat příkazy `\texttildelow` (~), `\textperthousand` (‰), `\textonehalf` (½), nebo třeba `\textmusicalnote` atd., v balíčku `gensymb` najdeme mimo jiné i příkazy `\celsius` (°C) nebo `\degree` (°).

V některých balíčcích jsou i symboly „ozdobné“ či „zábavné“, například v balíčku `marvosym` je `\Dontwash` (🜨) nebo podobenka autora balíčku `\MartinVogel` (🕭).

Spousta příkazů pro symboly je však dostupná bez dalších přídavných balíčků, například `\$` ($), `\#` (#), `\{` ({), `\&` (&), `\copyright` (©), `\texttrademark` (™), `$\ast$` (∗), `$\times$` (×), `$\alpha$` (α), atd. Mnohé najdeme v menu a na panelech editoru.

Některé zde uvedené příkazy jsou obklopeny znaky $ (například `$\ast$` nebo znaky řecké abecedy). Je to z toho důvodu, že jde o symboly určené pro matematickou sazbu (v rovnicích apod.), a znaky $ se obklopuje právě prostředí pro matematickou sazbu. To nám však nebrání tyto symboly používat i v běžném textu, stačí příkaz obklopit znaky dolaru.

Příklad 2.3 Všimněte si použití obdélníků na konci výpisu 2.6.

Výpis č. 2.6

```
Budeme-li v~\LaTeX u potřebovat komentář,
použijeme znak \% -- nic za ním nebude
v~dokumentu vidět, a~to ani ve dne
(6--19~hodin). % fakt!

Pro násobnost používáme znak $\times$,
například $3\times$, a~to v~matematickém
prostředí ohraničeném třeba symboly \$.
Rovnice vložená do textu: $x^2-y+25=250$.
\hfill\rule{1ex}{1ex}

Telefonní číslo můžeme zvýraznit symbolem
\Telefon{} z balíčku \texttt{marvosym}.
Tento odstavec má už 13~řádků, je třeba ho
ukončit \Rightscissors.
\rule{.5ex}{1ex}
\rule{2pt}{0pt}  % obdélník s nulovou výškou
\rule[-1ex]{.5ex}{1ex}  % snížený o 1ex
```

Budeme-li v LATEXu potřebovat komentář, použijeme znak % – nic za ním nebude v dokumentu vidět, a to ani ve dne (6–19 hodin).

Pro násobnost používáme znak ×, například $3\times$, a to v matematickém prostředí ohraničeném třeba symboly $. Rovnice vložená do textu: $x^2 - y + 25 = 250$. ■

Telefonní číslo můžeme zvýraznit symbolem ☎ z balíčku `marvosym`. Tento odstavec má už 13 řádků, je třeba ho ukončit ✂. ▮

2.4.5 Diakritická znaménka

Diakritika obvykle nečiní problémy, lze používat i diakritická znaménka, která nemáme dostupná na klávesnici. Například `\^{a}` nám vysází â, `\c{o}` dává ǫ, příkaz `\ae` je pro æ. Podrobnější seznam je například v [T31, str. 40] nebo v [T38, str.17].

Abychom nemuseli pro nezlomitelnou mezeru tak často psát vlnku (a tedy přepínat mezi českou a anglickou klávesnicí), existuje program, který to udělá za nás. Jmenuje se *vlna* nebo *vlna32*[4]. Program vlna32 (který budeme spíše volit ve Windows) se používá takto:

```
vlna32.exe soubor.tex
```

kde `soubor.tex` je zpracovávaný soubor. Dále je nutné přidat cestu k programu i zpracovávanému souboru.

Standardně bohužel program nezpracovává úplně vše, co by měl, ale máme možnost určovat, za která jednopísmenková slova bude vlnka přidána:

```
vlna32.exe -v KkSsVvZzOoUuAaIi soubor.tex
```

Tento způsob používání se může zdát poněkud pracný, ale pokud pracujeme v editoru TEXnicCenter nebo jiném překládájícím v profilech, dá se trochu zautomatizovat. Následující postup platí pro TEXnicCenter, v jiných editorech bude obdobný:

- v menu editoru najdeme okno Profiles (Build → Define output profiles),
- v okně vlevo dole klepneme na tlačítko Add,
- napíšeme název profilu (třeba *vlna*) a potvrdíme,
- v pravé polovině okna Profiles zaškrtneme políčka Do not use BibTeX... a Do not use MakeIndex... (odtud se nic spouštět nebude) a přepneme se na druhou záložku (Postprocessor),
- tam přidáme nový program (vlevo nahoře tlačítko se čtverečkem), napíšeme jeho název (je jedno jaký, třeba `vlna`) a do řádků níže zadáme:
  - doplníme nebo najdeme (tlačítko … vedle pole *Path*) adresu k programu `vlna32.exe`,
  - doplníme argumenty příkazu (Command line arguments)

    ```
    -v KkSsVvZzOoUuAaIi "%pm"
    ```

    (to znamená, že že se bude vlnka doplňovat za uvedená samotná písmena a `%pm` se nahradí názvem toho souboru, se kterým právě pracujeme).

Pak stačí vždy jen zvolit tento profil v rozbalovacím seznamu na hlavním panelu a klepnout na tlačítko pro překlad.

[4] Program si můžeme stáhnout na `ftp://math.feld.cvut.cz/pub/olsak/vlna/`. Potřebujeme soubor `vlna.exe` nebo `vlna32.exe` a dále soubor `vlna.txt`, kde je popsáno, jak se tento soubor používá.

2.5 Odstavcové styly

2.5.1 Prostředí

Prostředí je úsek textu, který má být určitým způsobem zformátován. Až na vyjímky vyznačujeme začátek prostředí příkazem `\begin{název prostředí}` a konec příkazem `\end{název prostředí}`. Každé prostředí je zároveň skupinou (viz kap. 2.4, str. 44), tedy příkazy s platností do konce skupiny (třeba pro změnu stupně písma) končí svou platnost nejen u nejbližšího výskytu `}`, ale také `\end{...}`.

V LaTeXu se běžně používá mnoho různých prostředí – například `center` pro zarovnání úseku textu na střed, `quotation` pro sazbu citátů, `itemize` jako výčtové prostředí s odrážkami, máme také několik matematických a tabulkových prostředí i prostředí, ve kterých se příkazy nevyhodnocují a považují se za běžný text. Některými z nich se budeme zabývat v následujícím textu, informace o dalších najdeme například v [T31], [T36], [T37] a dalších.

2.5.2 Formát odstavce

Parametry odstavce nastavujeme (většinou v preambuli – hlavičce – dokumentu) příkazy podle tabulky 2.3[5]. Pro vertikální odsazení odstavců a odstavcovou zarážku používáme násobky délek `ex` a `em`. Jsou odvozeny z rozměrů kuželky momentálního stupně písma, proto po změně stupně písma nebo fontu se optické vlastnosti odstavce příliš nezmění.

Ukázka příkazu	*Význam*
`\setlength{\parskip}{1ex}`[6]	nastaví vertikální odsazení odstavců na půl délky `ex`
`\setlength{\parindent}{2em}`	nastaví odsazení prvního řádku odstavce (zarážky) na 2 `em`
`\def\baselinestretch{1.5}\normalsize`	nastaví řádkování na 1,5 násobek běžného řádkování

Tabulka 2.3: Nastavení parametrů odstavce

Odstavce oddělujeme prázdným řádkem nebo příkazem `\par`, první možnost je samozřejmě jednodušší. Dále máme možnost přerušit řádek kdekoliv uprostřed odstavce příkazem `\\` nebo `\newline` tak, že na novém řádku nebude použito žádné odsazení (prostě se jen přesuneme na nový řádek). Zároveň můžeme i vynechat místo, velikost mezery udáváme v nepovinném parametru – například `\\[2em]`. Pro ukončení stránky použijeme příkaz `\newpage`.

Běžný text je zarovnán do bloku, tedy levý i pravý okraj jsou rovné. To máme možnost změnit a nastavit zarovnání na střed nebo na praporek – praporek vlající

[5] V prvních dvou případech jde o práci s tzv. *délkovými registry* (délkami), jakýmisi zástupnými názvy pro často používané délky. V registru `\parskip` je uložena vzdálenost mezi odstavci, v registru `\parindent` velikost odstavcové zarážky.

[6] Zde je vhodné dát určitou rezervu, aby TEX při vkládání konce stránky měl možnost mírně „pružně“ upravit vzdálenosti mezi odstavci. Pokud to neuděláme, TEX si ji stanoví sám tak, že se nám to nemusí líbit. Příkaz může vypadat například takto: `\setlength{\parskip}{1ex plus .3ex minus .2ex}`.

vpravo („zubatý“ text vpravo, rovný vlevo) je zarovnání vlevo, praporek vlající vlevo je zarovnání vpravo.

Zarovnání	*Příkaz*
na střed	`\begin{center} ... \end{center}` `{ ... \centering ... }`
vlevo (praporek doprava)	`\begin{flushleft} ... \end{flushleft}` `{ ... \raggedright ... }`
vpravo (praporek doleva)	`\begin{flushright} ... \end{flushright}` `{ ... \raggedleft ... }`

Tabulka 2.4: Možnosti zarovnání textu

V tabulce 2.4 máme u každé možnosti dva způsoby, jak určitého zarovnání dosáhnout. První je pomocí prostředí, druhá příkazem platným do konce skupiny (ve skutečnosti platí už od začátku odstavce, ve kterém se nachází). Většinou je lepší použít prostředí, druhý způsob je vhodný třeba při umísťování obrázku nebo tabulky. Když nechceme psát příkazy, můžeme využít menu nebo hlavní panel editoru, kde najdeme tlačítka podobná těm z Wordu.

Výpis č. 2.7

```
Zatím ještě máme odstavce zarovnány do
bloku, ale to brzy změníme hned po\\
ukončení řádku uvnitř
odstavce.

\begin{flushright}
Toto je ukázka zarovnání textu k~pravému
okraji. Platí až do konce prostředí,
jestliže zarovnání nezměníme některým
příkazem.

Tento\centering text je centrován, a~to již
od začátku odstavce.
\end{flushright}

Vracíme se k~zarovnání do bloku, předchozí
prostředí už skončilo.
```

Zatím ještě máme odstavce zarovnány do bloku, ale to brzy změníme, hned po
ukončení řádku uvnitř odstavce.

Toto je ukázka zarovnání textu k pravému okraji. Platí až do konce prostředí, pokud zarovnání nezměníme některým příkazem.

Tento text je centrován, a to již od začátku odstavce.

Vracíme se k zarovnání do bloku, předchozí prostředí už skončilo.

Pokud je určeno nenulové odsazení prvního řádku (tzv. *odstavcová zarážka*), můžeme používání této zarážky řídit příkazy `\noindent` (pro tento odstavec se nepoužije) a `\indent` (pro tento odstavec se použije). Druhý příkaz je užitečný například tehdy, když se zarážka používá pro všechny odstavce kromě prvního v kapitole, a my vyjímečně chceme odsadit i ten první. Oba příkazy píšeme těsně před začátek odstavce, pro který mají platit:

```
\noindent
text odstavce, který nemá mít odsazený první řádek
```

Odstavce mohou být *pojmenované*. Takto lze nenásilně bez nadpisů členit příliš dlouhou kapitolu, ve které se určité odstavce týkají vždy určitého tématu. Název

odstavce je formátován tučně stejně jako názvy kapitol, to ale můžeme změnit použitím příkazu `\normalfont` a příkazu pro vybraný formát.

Výpis č. 2.8

```
\paragraph{Pojmenovaný odstavec.}
Pojmenované odstavce používáme pro
podrobnější členění kapitoly, kde
odstavce lze pojmenovat jejich
tématem. Všimněte si vyznačení
pojmenování a~odsazení textu.

Někdy odstavec nelze pojmenovat -- pak
ho necháme tak, jak je. Obvykle se
počítá s~tím, že tematicky patří
k~předchozímu odstavci.

\paragraph{Nadpisy.} Nadpisy obvykle
členíme do úrovní, ale maximálně čtyř,
ideálně tří. Pro podrobnější členění
můžeme použít pojmenované odstavce.
```

Pojmenovaný odstavec. Pojmenované odstavce používáme pro podrobnější členění kapitoly, kde odstavce lze pojmenovat jejich tématem. Všimněte si vyznačení pojmenování a odsazení textu.

Někdy odstavec nelze pojmenovat – pak ho necháme tak, jak je. Obvykle se počítá s tím, že tematicky patří k předchozímu odstavci.

Nadpisy. Nadpisy obvykle členíme do úrovní, ale maximálně čtyř, ideálně tří. Pro podrobnější členění můžeme použít pojmenované odstavce.

2.5.3 Seznamy

V LaTeXu používáme tři druhy výčtových seznamů – výčtové (*itemize*), číslované (*enumeration*) a s popisky (s pojmenovanými položkami, *description*). Všechny tři druhy jsou tvořeny prostředím a položky začínají příkazem `\item`. U všech tří druhů můžeme v nepovinném parametru položky určit „popisek“, který však využijeme obvykle pouze u seznamu s popisky (to budou ty popisky), jinak tímto způsobem lze například změnit symbol pro odrážku.

Strukturu seznamu vytvoříme buď ručně podle příkladu 2.4 nebo pomocí tlačítek na hlavním panelu či položkami v menu (Insert → Enumerations), položky seznamu vložíme tamtéž (... Entry). V jednom prvku seznamu může být více odstavců.

V příkladu 2.4 je ve výpisu 2.9 ukázán způsob změny symbolu pro odrážku (v tomto smyslu však moc nepoužíváme), ve výpisu 2.10 vidíme možnost odkazování se na jednotlivé položky číslovaného seznamu (je to použitelné i mimo seznam), ve výpisu 2.11 je pozměněn formát posledního popisku.

Příklad 2.4 Ve třech výpisech vidíme ukázku použití základních druhů seznamů.

Výpis č. 2.9

```
Výčtový seznam:
\begin{itemize}
\item první odrážka,
\item druhá odrážka,
\item[\CheckedBox] třetí odrážka.
\end{itemize}
```

Výčtový seznam:

- první odrážka,
- druhá odrážka,

☑ třetí odrážka.

Výpis č. 2.10

```
Vztahy mezi zvířaty,
odkaz na \ref{seznam:kocka}:
\begin{enumerate}
\item Pes.
\item\label{seznam:kocka} Kočka.
\item Myš utíká před tím, co je v~položce
  \ref{seznam:kocka}.
\end{enumerate}
```

Vztahy mezi zvířaty, odkaz na **2**:

1. Pes.
2. Kočka.
3. Myš utíká před tím, co je v položce **2**.

Výpis č. 2.11

```
Seznam s~popisky:
\begin{description}
\item[Spojovník] píšeme jednoduše klávesou
  s~tímto znakem.
\item[Pomlčka] se zadává jako dva
  spojovníky.
\item[\normalfont\itshape Nezlomitelná
  mezera] se nahrazuje vlnovkou.
\end{description}
```

Seznam s popisky:

Spojovník píšeme jednoduše klávesou s tímto znakem.

Pomlčka se zadává jako dva spojovníky.

Nezlomitelná mezera se nahrazuje vlnovkou.

Výpis č. 2.12

```
Víceúrovňové číslování:
\begin{enumerate}
\item Toto je první úroveň, první odstavec.

Tady je druhý odstavec.
\item Další položka první úrovně.
  \begin{enumerate}
  \item Toto je druhá úroveň.
  \item Další položka.
  \end{enumerate}
\end{enumerate}
```

Víceúrovňové číslování:

1. Toto je první úroveň, první odstavec.

   Tady je druhý odstavec.
2. Další položka první úrovně.
   (a) Toto je druhá úroveň.
   (b) Další položka.

Výpis 2.12 je ukázkou možnosti vnořování výčtových prostředí, čímž vzniká třeba víceúrovňové číslování. Seznamy můžeme libovolně kombinovat včetně různých typů, LaTeX se vždy sám postará o vhodné odlišení úrovní. Jen si musíme dávat pozor na „překřížení“ začátků a konců různých prostředí.

Vlastnosti prostředí pro seznamy se určují obvykle hned za úvodním příkazem prostředí. Například vertikální mezery mezi položkami seznamu změníme takto:

```
\begin{itemize}\setlength{\itemsep}{0pt} \item ...
```

U seznamů se dále mohou hodit délkové registry `\partopsep`, `\labelsep`, `\itemindent`, `\leftmargini`, `\leftmarginii`, `\leftmarginiii`, `\leftmarginiv`.

2.6 Dokumenty

2.6.1 Třída dokumentu

Typ (třídu) dokumentu určujeme v příkazu `\documentclass`. Standardní třídy jsou:

article – podle názvu by se mohlo zdát, že je použitelná pouze pro články, ale je to nejpoužívanější třída pro běžné texty do velikosti několika desítek stran. Výchozí nastavení je jednostranný dokument.

report – třída vhodná pro dokumenty v délce několika desítek až stovek stran, výchozí nastavení je jednostranný dokument.

book – třída vhodná pro rozsáhlé dokumenty. Oproti třídě `report` je nastaven oboustranný dokument a jinak zformátováno záhlaví a zápatí.

letter – třída pro psaní dopisů.

Nejčastěji budeme používat třídu `article`.

Zároveň s určením třídy obvykle nastavujeme některé vlastnosti, které se nám ve výchozím nastavení nelíbí. Tyto vlastnosti se uvádějí do nepovinného parametru. Například když chceme použít třídu `article`, ale nastavit rozměr stránky A4, stupeň písma 12 a oboustranný dokument, uděláme to takto:

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

Nikde neděláme mezery, vlastnosti mohou být v jakémkoliv pořadí.

Kromě standardních tříd existují ještě další třídy, obvykle vytvořené lidmi k určitému konkrétnímu účelu. Hodně tříd je pro vytváření prezentací (například `beamer`, `prosper`, `ifmslide`, `texpower`, `elpres`), zasílání článků do odborných časopisů (`paper`, `llncs`, `elsevier`, `acmtrans2e`, `amsart`, `nature`), diplomové a disertační práce (`scrreprt`, `scrartcl`, `adfathesis`, `classicthesis`, `ua-thesis`), referenční manuály (`refart`, `refrep`), odborné knihy a beletrii (`scrreprt`, `scrbook`, `hitec`, `memoir`, `octavo`, `amsbook`), životopisy (`cv`, `moderncv`), letáky (`leaflet`), laboratorní záznamy (`labbook`), postery (plakáty většinou v rozměru A0, používají se například na konferencích pro grafickou prezentaci výsledků výzkumu, máme možnost určovat pozici jednotlivých objektů či textu na stránce) (`a0poster`) atd.

Seznamy z předchozího odstavce rozhodně nejsou kompletní. Počet těchto tříd je různý v různých distribucích TEXu, je jich však více, než běžný uživatel tuší (běžnému uživateli ostatně stačí standardní třídy). Později, v kapitole 2.6.2, si ukážeme, jak o nich najít informace (tyto třídy se liší nejen nastavením stylů, tedy „vizuálně“, ale někdy i způsobem použití – definovanými příkazy) a jak je využít v dokumentech.

2.6.2 Styly

Podobně jako jiné systémy, i LATEX umožňuje ovlivňovat vzhled dokumentu úpravou stylů. Pro jednodušší dokumenty dostačují předdefinované styly (pro odstavce, nadpisy apod.), tyto styly lze ovlivňovat následovně:

1. *Výběrem třídy dokumentu.* Každý dokument má přiřazenu právě jednu třídu, která stanoví základní vzhled a strukturu a případně přidává některé specifické

příkazy, prostředí a objekty k té třídě, ze které je odvozena. S některými třídami jsme se seznámili v kapitole 2.6.1, tedy už víme, že třída se načítá příkazem

```
\documentclass[nepovinné argumenty]{název třídy}
```

Například `\documentclass[a4paper,11pt,twoside]{report}` znamená, že pro dokument chceme použít třídu `report`, budeme tisknout na papír formátu A4, velikost písma je 11 bodů a jde o oboustranný dokument.

2. *Načtením balíčku.* V balíčku (package) je nadefinován nebo pozměněn jen jeden nebo více stylů nebo je vytvořen nový příkaz či prostředí pro některý konkrétní účel, obvykle lze balíček používat zároveň s jinými balíčky nebo kteroukoliv třídou. Už dříve jsme používali příkaz pro načtení balíčku

   ```
   \usepackage[nepovinné argumenty]{název balíčku}
   ```

   Nepovinné argumenty jsme u balíčků ještě nepoužívali, seznámíme se s nimi například u balíčku `hyperref` nebo `color`.

   V jednom příkazu `\usepackage` můžeme načíst více balíčků, jak uvidíme v příkladu 2.5.

3. *Změna nebo vytvoření stylu přímo v dokumentu.* V dokumentu, obvykle v jeho hlavičce (preambuli), si můžeme vytvořit vlastní příkazy nebo prostředí nebo pozměnit existující. Ve většině případů používáme příkazy `\def`, `\newcommand`, `\renewcommand`, `\newenvironment`, `\renewenvironment` nebo pro nastavení délek `\setlength`. Pro číslované prostředí je výhodné použít třeba příkaz `\newtheorem` v kombinaci s jiným příkazem. Ukázky použití těchto příkazů jsou ve výpisu 2.14 v příkladu 2.5.

4. *Jiné.* Některé styly se načítají trochu jiným způsobem. Jde například o různé typy a řezy písma. My jsme si v kapitole 2.4.3 o fontech ukazovali změnu typu písma načtením balíčku, ale obvyklejší (a o něco složitější) cesta je pomocí dalších příkazů (viz [T31]). Dále některé zvláštní řezy písma se načítají trochu netradičně, například řez *tapír* se používá takto:

Výpis č. 2.13

```
% inicializace třeba v preambuli dokumentu:
\font\tapir tap

% používání kdekoliv v dokumentu:
{\tapir Tento text je psán řezem tapír.\par Bohužel
v~něm nefungují kombinace s~dalšími řezy.}
```

Tento text je psán řezem tapír.

Bohužel v něm nefungují kombinace s dalšími řezy.

Příklad 2.5 Ve výpisu máme ukázku jednoduché preambule dokumentu typu článek (`article`) s fontem bookman a také vidíme způsob použití výše uvedených příkazů pro definování nových příkazů či stylů. Přímo z příkladů lze odvodit způsob používání jednotlivých příkazů včetně potřebných argumentů.

Výpis č. 2.14

```
\documentclass[a4paper,12pt]{article}
\usepackage{czech,a4wide,comment}   % nikde v parametrech nesmí být mezera!!!
    % balíček a4wide zajistí obvyklé rozměry okrajů stránky pro formát A4,
    % balíček comment umožní používat prostředí comment pro víceřádkové komentáře
    %   \begin{comment} ... \end{comment}
\usepackage{bookman}

\usepackage{textcomp,gensymb,marvosym,wasysym}   % balíčky s různými symboly

% příkaz \zahlavitabulky se použije pro styl záhlaví tabulky, na tomto místě
% (v preambuli dokumentu) to lze jednoduše změnit pro celý dokument:
\def\zahlavitabulky{\bfseries}

% vytváříme vlastní příkaz s jedním parametrem:
\newcommand{\kolecko[1]{\textcircled{\tiny #1}}
% používá se například takto: \kolecko{10} - vytvoří číslo 10 v kolečku

% vytvoříme jednoduché prostředí mírně pozměňující původní prostředí itemize,
% pozměníme mezery mezi položkami seznamu (všimněte si strukturování příkazu
% pomocí komentářů):
\newenvironment{mitemize}%
  {\begin{itemize}\setlength{\itemsep}{0pt}}%
  {\end{itemize}}
% používá se: \begin{mitemize} ... \end{mitemize}

% následující prostředí slouží k vhodnému formátování poznámek - na začátku
% slovo Poznámka následované mezerou a na konci podtržení celého odstavce:
\newenvironment{poznamka}%
  {\textbf{Poznámka:\quad}}%
  {\newline\rule{0pt}{1pt}\hrulefill\rule{0pt}{1pt}}

% potřebujeme číslované prostředí priklad, číslování bude záviset na číslu
% kapitoly,využijeme kombinaci newtheorem a newenvironment:
\newtheorem{thpriklad}{Příklad}[chapter]
\newenvironment{priklad}%
  {\begin{thpriklad}\rule{1pt}{0pt} \normalfont}%
  {\end{thpriklad}}
% používá se: \begin{priklad} ... \end{priklad}
```

U příkazů `\def` a `\newcommand` jsou argumenty celkem jasné – je potřeba pouze zadat, co se stane, když je vytvářený příkaz použit, v nepovinném parametru můžeme určit případné parametry příkazu (pro `\def` viz nápověda).

U příkazu `\newenvironment`, který vytvoří nové prostředí, zadáváme po jeho názvu nejdřív, co se má provést při `\begin{nové prostředí}`, tedy na jeho začátku, a v dalším argumentu určíme, co se má provést při `\end{nové prostředí}`. U prostředí definovaného příkazem `\newtheorem` obvykle pouze určíme jeho název a vazbu číslování (máme více možností, než je zde naznačeno, podrobnosti najdeme v nápovědě).

Častou a nenápadnou chybou je „chaos“ v závorkách. Když nám program hlásí při překladu chybu související se závorkami nebo s hranicemi prostředí, nezbývá nic jiného než počítat levé a pravé závorky a kontrolovat jejich umístění. Zde se vyplácí psát přehledně jak je to ukázáno v předchozím příkladu.

Třídy jsou definovány v souborech s příponou `.cls`, balíčky v souborech s příponou `.sty`. Najdeme je obvykle v adresáři, kde je TEX nainstalován, a to v podadresáři `...\tex\latex\název třídy či balíčku`. Dokumentace k těmto souborům bývá pak v adresáři `...\doc\latex\název třídy či balíčku`, většinou v `.dvi` nebo `.pdf` souborech, někdy doprovázená soubory s příklady.

Na internetu (na [P19]) lze někdy sehnat novou třídu nebo balíček, případně novou verzi. Při instalaci postupujeme podle pokynů v souboru dodávaném s třídou nebo balíčkem, obvykle jde jen o to v adresáři `...\tex\latex` vytvořit vhodně pojmenovaný adresář, do něho zkopírovat několik souborů a pak obnovit databázi.

O jakou databázi vlastně jde? TEX se skládá ze spousty velmi malých souborů. Aby ho časté přistupování k těmto souborům zbytečně nezdržovalo (především jejich hledání na disku), udržuje si jejich databázi, kde má uloženy veškeré potřebné informace, které mohou urychlit přístup k používaným souborům. To značně zrychluje práci, ale vedlejším efektem je, že když provedeme změny ve struktuře těchto souborů (včetně přidání nového balíčku), musíme o tom TEXu „dát vědět“. U MiKTEXu ve Windows se to dělá v menu START pomocí START → MiKTeX → MiKTeX Options, tlačítko Refresh Now.

Úlohy:

1. Zjistěte, ve kterém adresáři (složce) je na vašem počítači nainstalován TEX, a kde jsou soubory s třídami, balíčky a jejich dokumentací. Nezapomeňte, že samotný operační systém se svou funkcí hledání vám může ulehčit práci.
2. Srovnejte dokumentace k třídám `jasthesis`, `adfathesis`, `scrreprt` (je v adresáři `koma-script`, v souboru `scrguien.pdf`) a `octavo`. Všechny tyto třídy vyzkoušejte na jednoduchém dokumentu s nadpisy a textem.
3. Najděte dokumentaci k těmto balíčkům:
   (a) balíček `clock`, zjistěte, jak se vkládá obrázek hodin a jak se ovládá jeho vzhled,
   (b) balíčky pro úpravu nadpisů – `fncychap`, `sectsty`, `quotchap`,
   (c) balíček pro tvarování odstavců – `shapepar`.
4. Zjistěte, k čemu slouží balíčky `parallel` a `qsm`.
5. Najděte dokumenteci balíčků pro prezentace `beamer` a `prosper`, u obou se podívejte na příklady (u `prospera` bude nutné provést překlad `LaTeX=>PS=>PDF`).

2.6.3 Nadpisy

Na tom, pro jakou třídu dokumentu jsme se rozhodli, závisí nejen vzhled výsledného dokumentu a výchozí nastavení vlastností (jednostranný, oboustranný apod.), ale v některých případech i příkazy, které máme k dispozici. Týká se to především nadpisů.

Ve výpisu 2.15 je způsob vkládání nadpisů prvních tří úrovní, a to v prvním sloupci pro třídu `article`, v druhém pro třídy `report` a `book`. Jak vidíme, `article` nepoužívá vůbec příkaz `\chapter`. Tento příkaz totiž generuje graficky řešené, „zdobnější“ a nápadnější nadpisy první úrovně, kde obvykle bývá nadpis samotný na jiném řádku než číslo kapitoly, zatímco příkazy obsahující slovo section generují jednořádkové nadpisy s číslem a textem nadpisu.

Výpis č. 2.15

```
\section{Nadpis první úrovně}
Text první kapitoly článku.
\subsection{Nadpis druhé úrovně}
Text podkapitoly.
\subsubsection{Nadpis třetí úrovně}
atd.
```

```
\chapter{Nadpis první úrovně}
Text první kapitoly dokumentu.
\section{Nadpis druhé úrovně}
Text podkapitoly.
\subsection{Nadpis třetí úrovně}
atd.
```

Někdy je třeba vložit *nečíslovaný nadpis*, ať už kterékoliv úrovně (třeba pro úvodní kapitolu). To provedeme mírnou změnou příkazu pro vložení nadpisu, a to přidáním hvězdičky. Pro různé úrovně tedy použijeme příkazy:

```
\chapter*{Nadpis první úrovně v report a book}
\section*{Nadpis druhé úrovně v report a book nebo první v article}
```

atd. pro další úrovně. Nečíslované nadpisy se však nezahrnou do obsahu, ale i to lze řešit přidáním příkazu pro přidání položky obsahu – podle příkladu 2.6.

Příklad 2.6 Úvodní kapitola má být nečíslovaná, ale chceme ji v obsahu dokumentu. Vlevo je řešení pro třídu `article`, vpravo pro `report` a `book`:

Výpis č. 2.16

```
\section*{Úvod}
\addcontentsline{toc}{section}{Úvod}
text kapitoly článku
```

```
\chapter*{Úvod}
\addcontentsline{toc}{chapter}{Úvod}
text kapitoly rozsáhlejšího dokumentu
```

V parametrech příkazu zkratka `toc` znamená *Table of Contents*, tedy obsah. Když použijeme `lof`, vkládáme do seznamu obrázků (*List of Figures*), když použijeme `lot`, vkládáme do seznamu tabulek (*List of Tables*). Dalšími možnostmi ovlivňování dokumentových seznamů se budeme zabývat v kapitole 2.12.

Dokument může být členěn také na větší díly než kapitoly. Když chceme shrnout více kapitol do většího celku, části, pak pro jednotlivé části použijeme „nadnadpis“ `\part{Název části}`.

Pro oddělení příloh od běžných kapitol použijeme příkaz `\appendix`. Od tohoto příkazu dále budou nadpisy nejvyšší úrovně číslovány velkými písmeny od „A“.

Ve třídě book jsou k dispozici příkazy, které rozčlení dokument na části lišící se především číslováním stránek – \frontmatter, \mainmatter a \backmatter. Způsob jejich použití je naznačen ve výpisu 2.17.

Výpis č. 2.17

```
...
\begin{document}
\frontmatter    % přední část dokumentu
... titulní strany, předmluva, obsah, seznam tabulek, apod.
\mainmatter    % hlavní část dokumentu
... běžné kapitoly dokumentu, číslované čísly
\appendix
... přílohy číslované písmeny, také patří do hlavní části dokumentu
\backmatter    % koncová část dokumentu
... seznam literatury, rejstřík apod.
```

2.6.4 Titul a abstrakt

Už ve výpisu 2.1 na straně 39 jsme si mohli všimnout, že článek nemá jen název, ale je uveden i autor, datum a abstrakt.

Název dokumentu, seznam autorů a datum jsou součástí titulu, některé typy dokumentů do titulu řadí i jiné informace. Zatímco v případě článku (třída article) je titul jen jednou ze součástí první strany, třídy report a book vytvářejí automaticky celou titulní stranu. Většinou používáme následující sekvenci příkazů:

Výpis č. 2.18

```
\title{Název dokumentu}
\author{Autor dokumentu}    % Více autorů oddělíme příkazem \and
\date{\today}    % Když příkaz vynecháme, automaticky se doplní skutečné datum,
                 % pokud příkaz použijeme, ale prázdný, datum nebude součástí
                 % titulu vůbec

\maketitle
```

V editoru TEXnicCenter nemusíme tyto příkazy vypisovat, můžeme použít menu Insert → Document Title, kde Title Properties vloží příkazy pro název, autora a datum, Title pak příkaz \maketitle.

Volbu Title Page využijeme, pokud chceme i ve třídě article mít celou titulní stranu. Je to prostředí, které má obsahovat příkazy vložené volbou Title Properties a nahrazuje příkaz \maketitle.

Do parametrů uvedených příkazů nemusíme vkládat pouze text, můžeme použít příkazy pro jinou velikost písma, vkládat více řádků, použít různé řezy písma či dokonce barvy a obrázky (u těch je trochu problém s umístěním).

V menu Insert → Document Title najdeme i volbu Abstract k vložení *abstraktu* (nebo napíšeme \begin{abstract} ...\end{abstract}). Ten následuje za titulem.

Když píšeme článek nebo dokument s použitím jiné než standardní třídy, obvykle s touto třídou máme k dispozici jakýsi „návod k použití“, což je dokument psaný s použitím této třídy a popisuje její možnosti a způsob, jak je využívat. V kapitole 2.6.2 je o této problematice více.

2.6.5 Poznámky pod čarou a na okraj

Občas potřebujeme vložit do textu *poznámku pod čarou*. Ta se vkládá příkazem

```
\footnote{Text poznámky pod čarou.}
```

Před tímto příkazem obvykle nebývá mezera (znak poznámky pod čarou musí přilé-hat k tomu, co komentujeme), v poznámce píšeme „celými větami“, tedy ještě před ukončující závorkou bývá tečka či jiné interpunkční znaménko, a to i v případě, že je i za závorkou.

Poznámka na okraj.

Poznámka na okraj se vytvoří příkazem

```
\marginpar[text pro levý okraj]{text pro (pravý) okraj}
```

Text se vloží jako poznámka k pravému okraji textu u jednostranného dokumentu, u oboustranného vždy na vnější stranu. Pokud zadáme i nepovinný parametr, tak v případě, že u oboustranného dokumentu vyjde poznámka vlevo, použije se text (případně s příkazy) z nepovinného parametru.

Poznámka je vždy zarovnána ke svému levému okraji, ale nic nám nebrání použít příkaz nebo prostředí pro zarovnání doprava na levé stránce, jak je tomu i ve výpisu 2.19 (poznámka vytvořená druhým odstavcem výpisu je u předchozího textu), případně různé formátovací příkazy.

Výpis č. 2.19

```
Poznámky pod čarou můžeme vložit téměř kdekoliv, například tady\footnote{Toto je
poznámka pod čarou.}, místo je označeno obvykle číslem. Poznámky na okraj
\marginpar[\raggedleft Toto je poznámka na okraj.]{Toto je poznámka na okraj.}
se objeví na okraji v~linii umístění tohoto příkazu.

V poznámce na okraj nemusí být jenom text, ale také obrázek nebo text různě
upravený, například\marginpar{\raisebox{-1em}{\fbox{\huge\bfseries ?}}}.
```

Poznámku na okraj nelze přímo použít uvnitř některých prostředí, například v tabulkách nebo v prostředí `minipage`. Existuje však možnost, jak to obejít (je použita například v tabulce 2.3 na straně 52) – na místě, kde chceme poznámku vložit, napíšeme místo příkazu `\footnote` příkaz

```
\footnotemark
```

a hned za koncem daného prostředí (třeba tabulky nebo `minipage`) použijeme příkaz

```
\footnotetext{Text poznámky pod čarou.}
```

2.6.6 Záhlaví a zápatí

Stránky dokumentu jsou obvykle číslovány, a pokud je vytvořena titulní strana, automaticky není číslována. Způsob číslování můžeme ovlivnit různými způsoby. Jeden z nich jsme si už řekli – ve třídě `book` pomocí `\frontmatter`, `\mainmatter` a `\backmatter`.

Styl číslování úzce souvisí s definováním záhlaví a zápatí. Ve třídách `article` a `report` je číslo stránky vždy v zápatí uprostřed, záhlaví je prázdné, ve třídě `book` je v záhlaví číslo stránky a název kapitoly a na stránkách, kde začíná hlavní kapitola, je záhlaví prázdné a číslo stránky je v zápatí uprostřed.

Styl stránky včetně umístění číslování ovlivníme příkazy

```
\pagestyle{styl stránky}
\thispagestyle{styl stránky}
```

kde styl stránky může standardně být

- `empty` – záhlaví a zápatí jsou prázdné, stránky nejsou číslovány,
- `plain` – záhlaví je prázdné a v zápatí uprostřed je číslo stránky – výchozí pro třídy `article` a `report`,
- `headings` – platí to, co bylo popsáno pro třídu `book`,
- `myheadings` – můžeme do určité míry určovat obsah záhlaví a zápatí.

První příkaz bývá často v preambuli, ale může být kdekoliv v dokumentu, platí od stránky, na které je uveden. Druhý příkaz platí pouze pro stránku, na které je uveden. Tyto příkazy nemají vliv na hodnotu čísla stránky, jen na to, zda a kde je.

Pokud je nastaven styl `myheadings`, máme k dispozici příkazy pro změnu obsahu záhlaví:

```
\markright{pravá hlavička}
\markboth{levá hlavička}{pravá hlavička}
```

První příkaz se používá pro jednostranné dokumenty, druhý pro oboustranné. Do parametru můžeme napsat jakýkoliv text (třeba název dokumentu).

U těchto příkazů je nutno si dát pozor na začátek jejich platnosti. Záhlaví na *levé* stránce je vygenerováno podle *posledního použití* příkazu `\markboth` před začátkem této stránky, zatímco záhlaví na *pravé* stránce podle *prvního použití* `\markboth` nebo `\markright` na této stránce.

Další příkaz, tentokrát ovlivňující i hodnotu čísla, je

```
\pagenumbering{typ číslic}
```

Typ číslic může být `arabic` (běžné arabské číslice), `roman` (malé římské číslice) nebo `Roman` (velké římské číslice).

Tento příkaz, jak je z parametru patrné, mění typ číslic pro číslo stránky, platí od strany, na které je příkaz uveden. Pokaždé je číslo stránky vynulováno, tedy na straně, kde je příkaz uveden, je vždy číslo 1, i nebo I (ne 0, protože číslo je vždy před vložením na stránku zvýšeno o 1).

Mnohem lepší možnosti ovlivnění tvaru záhlaví a zápatí máme s balíčkem `fancyhdr`. Používá se stejně jako předdefinované styly stran, tedy po načtení balíčku příkazem `\usepackage` použijeme příkaz `\pagestyle{fancy}` a dále definujeme záhlaví a zápatí příkazy z tabulky 2.5.

Parametr `pozice` u posledních dvou příkazů tabulky 2.5 se skládá ze dvou písmen. První určuje, zda se obsah má vložit vlevo (L), doprostřed (C) nebo vpravo

Jednostranný dokument:	
`\lhead{obsah}`, `\chead{obsah}`, `\rhead{obsah}`	levá, střední, pravá část záhlaví
`\lfoot{obsah}`, `\cfoot{obsah}`, `\rfoot{obsah}`	levá, střední, pravá část zápatí

Oboustranný dokument:	
`\fancyhead[pozice]{obsah}`	obsah záhlaví na dané pozici
`\fancyfoot[pozice]{obsah}`	obsah zápatí na dané pozici

Tabulka 2.5: Příkazy balíčku `fancyhdr`

(R) na dané stránce, druhý udává, jestli se má příkaz použít pro liché (E – even) nebo sudé (O – odd) stránky. Když chceme určit více různých dvojic, oddělíme je čárkou. Tedy pokud chceme mít na lichých stránkách v záhlaví vlevo název dokumentu a v zápatí na sudých stránkách vlevo a lichých vpravo datum, použijeme příkazy

```
\fancyhead[LE]{Název našeho dokumentu}
\fancyfoot[LO,RE]{\today}
```

V parametrech příkazů můžeme také použít příkaz, který vygeneruje název hlavní kapitoly (napíšeme `\leftmark`), kapitoly druhé úrovně (`\rightmark`) nebo číslo stránky (`\thepage`). Dále lze nastavit čáry oddělující záhlaví nebo zápatí od textové části stránky (jejich výšku):

```
\renewcommand{\headrulewidth}{tloušťka}
\renewcommand{\footrulewidth}{tloušťka}
```

Příklad 2.7 Ukážeme si vytvoření jednoduchého záhlaví a zápatí v jednostranném (vlevo) a oboustranném (vpravo) dokumentu pomocí balíčku `fancyhdr`.

Výpis č. 2.20

```
...
\pagestyle{empty}
... stránky bez číslování
\pagestyle{fancy}
\pagenumbering{Roman}
\cfoot{}
\rhead{\thepage}
\lhead{}
\renewcommand{\headrulewidth}{0pt}
... obsah; římské číslování stránek
\lhead{\leftmark}
\renewcommand{\headrulewidth}{.8pt}
\pagenumbering{arabic}
... stránky s arabským číslováním
```

```
...
\pagestyle{empty}
... stránky bez číslování
\pagestyle{fancy}
\pagenumbering{Roman}
\fancyfoot[CE,CO]{}
\fancyhead[LO,RE]{\thepage}
\fancyhead[LE,RO]{}
\renewcommand{\headrulewidth}{0pt}
... obsah; římské číslování stránek
\fancyhead[LE]{\leftmark}
\fancyhead[RO]{\rightmark}
\renewcommand{\headrulewidth}{.8pt}
\pagenumbering{arabic}
... stránky s arabským číslováním
```

2.6.7 Dlouhé dokumenty

Samotnému TEXu, LATEXu a ani programům zobrazujícím DVI, PS a PDF soubory obvykle dlouhé dokumenty nečiní žádné problémy. Přesto však především pro přehlednost, snadnější orientaci a také pro zjednodušení práce s editorem je vhodné velmi dlouhý zdrojový soubor rozdělit do více souborů.

Všechny soubory, které takto vytvoříme, budou mít příponu TEX a měly by být ve stejném adresáři (složce). Jeden ze souborů bude *hlavní*. V něm je dána struktura celého dokumentu včetně preambule, titulních stran, obsahu apod., tedy styl dokumentu určujeme právě zde. Tento soubor pojmenujeme tak, jak chceme mít pojmenovaný i výsledný dokument (kromě přípony, samozřejmě).

Jednotlivé části (třeba kapitoly) uložíme do vhodně pojmenovaných souborů s příponou TEX a vložíme do hlavního dokumentu pomocí některého z těchto příkazů:

```
\input{soubor}
\include{soubor}
```

V obou případech píšeme název souboru *bez přípony*. První z těchto dvou příkazů vlastně jen provede při každém překladu „vkopírování“ vkládaného souboru do toho, kde je příkaz uveden, druhý před a po tomto přenesení navíc vloží stránkový zlom (tedy se přejde na novou stránku). Při vkládání celých kapitol, které mají začínat na nové stránce, proto používáme příkaz `\include`, jindy příkaz `\input`. Když potřebujeme stránkový zlom před souborem, ale ne za ním, vložíme soubor příkazem `\input` a před ním použijeme třeba příkaz `\newpage`.

Příklad 2.8 Ve výpisu 2.21 vidíme možný tvar hlavního souboru `chaloupka.tex`. Vkládané soubory nemusí mít žádnou zvláštní strukturu, obvykle začínají příkazem `\chapter`.

Výpis č. 2.21

```
\documentclass[11pt,a4paper]{report}
\usepackage{czech,a4wide}

\title{Perníková chaloupka\\[1em] Možný postup konstrukce}
\author{Jan Novák}
\date{\itshape Poslední aktualizace: \today}

\begin{document}
\maketitle
\tableofcontents   % vložení obsahu

\include{uvod}   % vložen soubor uvod.tex, nečíslovaná kapitola Úvod
\include{material}   % vložen soubor material.tex s kapitolou Použitý materiál
\newpage\input{konstrukce1}
  % soubor konstrukce1.tex s první částí kapitoly Postup konstrukce
\input{konstrukce2}   % soubor konstrukce2.tex, druhá část kapitoly
\include{statika}   % vložen soubor statika.tex, kapitola Statika objektu
\include{limitypouziti}   % limitypouziti.tex, kapitola Limity použití konstrukce

\appendix   % následují přílohy:
\include{prilohaopohadce}   % prilohaopohadce.tex, první příloha
\include{konstrukcninakresy}   % konstrukcninakresy.tex, druhá příloha
```

```
\include{zaver}   % vložen soubor zaver.tex, nečíslovaná kapitola Závěr
\include{literatura}
  % literatura.tex, nečíslovaná kapitola se seznamem literatury
\end{document}
```

Překládáme vždy jen hlavní dokument, tedy například v editoru TEXnicCenter bychom se měli před každým překladem přesunout ze souboru, se kterým pracujeme (a také ho uložit), do hlavního souboru a pak teprve spustit překlad. Tento postup nám zjednodušuje možnost práce s *projekty*, kterou nabízí mnoho editorů určených pro práci s TEXem a LATEXem. Bohužel v každém editoru se s projekty pracuje trochu jinak, včetně přípon pomocných souborů projektu.

TEXnicCenter ukládá informace o projektu do pomocných souborů pojmenovaných stejně jako hlavní soubor, a to s příponou `TCP` (nastavení editoru, jako je verze, jazyk, naposledy použitý profil pro překlad, název hlavního souboru apod.) a `TPS` (které soubory jsou právě otevřené–rozpracované, velikost a pozice oken těchto souborů, pozice kurzoru v souboru apod.). Oba soubory jsou textové, tedy je můžeme otevřít třeba v Poznámkovém bloku.

Do těchto souborů se ukládají data při každém uzavření projektu. Projekt otevřeme jednoduše tak, že poklepeme na soubor `TCP` a získáme stejné pracovní prostředí jako při posledním ukončení práce.

Překlad projektu se provádí jiným tlačítkem a klávesovou zkratkou než v případě jednoho dokumentu – používáme tlačítko nebo klávesovou zkratku F7. Máme také k dispozici strukturu objektu, seznam souborů a seznam objektů v okně Navigator, které si zobrazíme přes menu View → Navigator Bar.

Nový projekt vytvoříme (v editoru TEXnicCenter) přes menu File → New project a existující otevřeme přes File → Open project nebo v případě, že jsme s ním na tomto počítači už pracovali, rychlejší cestou přes File → Recent Projects. Může se také hodit možnost rychle bez hledání otevřít hlavní soubor projektu, pokud tak editor neučinil, a to přes Project → Open main file.

Projekt lze vytvořit i dodatečně, stačí v menu editoru zvolit Project → Create with active file as main file.

2.7 Vícesloupcová sazba

Někdy se může hodit možnost sazby do více sloupců, zúžení odstavce nebo dokonce naskládání několika takových odstavců vedle sebe (jeden z postupů je využit i v tomto dokumentu pro umístění výpisu zdrojového textu a výsledku vedle sebe).

Pro vícesloupcovou sazbu nebo skládání bloků textu třeba i vedle sebe jsou v TEXu tyto možnosti:

1. Při určení třídy dokumentu použijeme také parametr `twocolumn`, například:

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

Pak se sice titul dokumentu vysází „jednosloupcově“, ale text dokumentu bude ve dvou sloupcích. Tento způsob není použitelný, pokud chceme v dokumentu měnit dvousloupcovou a jednosloupcovou sazbu nebo používat více než dva sloupce.

2. Použijeme balíček `multicol`, ve kterém je definováno prostředí `multicols`, tedy do hlavičky dokumentu umístíme příkaz pro načtení balíčku

   ```
   \usepackage{multicol}
   ```

   a text, který chceme zformátovat do více sloupců, uzavřeme do prostředí:

   ```
   \begin{multicols}{počet sloupců} text \end{multicols}
   ```

   Například pro tři sloupce:

   ```
   \begin{multicols}{3}
   text, který chceme mít ve více sloupcích
   \end{multicols}
   ```

   Všechny sloupce budou míst stejnou šířku.

3. Text nebo jiné objekty umístíme do příkazu `parbox`, který se používá následovně:

   ```
   \parbox[zarovnání od předchozího]{šířka}{text nebo něco jiného}
   ```

   Nepovinných parametrů je ve skutečnosti více než jeden (dokonce i výška parboxu), podrobnosti viz nápověda nebo [T31]. V nepovinném parametru, který zde máme uveden, můžeme zadat některé z písmen `t`, `b`, `c` (`top`, `bottom`, `center`) pro vertikální pozici celého boxu vzhledem k okolnímu textu.

4. Podobně se používá prostředí `minipage`:

   ```
   \begin{minipage}[zarovnání od předchozího]{šířka}
   text nebo něco jiného
   \end{minipage}
   ```

Šířku můžeme udat v běžných jednotkách, ale protože se obvykle vztahuje k šířce stránky (tu vrací příkaz `\textwidth`), často se používá údaj, který se odvíjí právě od šířky stránky. Například pokud chceme zadat o něco méně než polovinu stránky, napíšeme jako šířku údaj `.49\textwidth` a automaticky se šířka `parboxu` nastaví na 0,49-násobek šířky stránky[7].

Při skládání `parboxů` a prostředí `minipage` je můžeme sázet přímo vedle sebe nebo oddělovat jinými objekty (třeba příkazem `\rule` s některým parametrem nulovým, příkazem `\hfill` nebo jiným parboxem).

Příklad 2.9 Ukážeme si použití prostředí `multicols`, příkazu `parbox` a prostředí `minipage`. `parbox` a `minipage` ohraničíme rámečkem (příkaz `\fbox`), abychom viděli skutečnou zabranou šířku. Použijeme také příkaz `\rule`, který vykreslí obdélník

[7] Všimněte si, že mezi reálným číslem .49 (tedy 0,49) a příkazem `\textwidth` se sice symbol pro násobení nepíše, ale vyhodnocuje se zde.

(v prvním případě o nulové výšce a pak o výšce dva body), jeho účelem je oddálit od sebe předchozí a následující objekty o zadanou „šířku“. Druhý má výšku nenulovou, abychom viděli skutečnou pozici obdélníku.

Výpis č. 2.22

```
\begin{multicols}{2}
Toto je text, který bude zarovnán do dvou sloupců, protože v~parametru prostředí
\texttt{multicols} je číslo 2. Kdyby tam bylo třeba číslo 4, tak by byl text
zarovnaný do čtyř sloupců. V~tomto prostředí mohou být trochu problémy třeba
s~popisky obrázků nebo poznámkami pod čarou.
\end{multicols}

\noindent
\fbox{    % všimněte si odsazení dalšího textu zleva - pro přehlednost
  \parbox{.45\textwidth}{
  Toto je obsah prvního boxu o šířce 0,45-násobku šířky textu na stránce.
  }
}
\rule{1ex}{0pt}    % tento obdélník nebude vidět
\fbox{
  \parbox{.5\textwidth}{
  Toto je obsah druhého boxu širokého polovinu šířky textu na stránce.
  }
}

\noindent
\fbox{
  \begin{minipage}{.45\textwidth}
  Toto je obsah prvního boxu o šířce 0,45-násobku šířky textu na stránce.
  \end{minipage}
}
\rule{1ex}{2pt}    % tento obdélník bude vidět
\fbox{
  \begin{minipage}{.5\textwidth}
  Toto je obsah druhého boxu širokého polovinu šířky textu na stránce.
  \end{minipage}
}
```

Zde vidíme výsledek předchozího kódu:

Toto je text, který bude zarovnán do dvou sloupců, protože v parametru prostředí `multicols` je číslo 2. Kdyby tam bylo třeba číslo 4, tak by byl text zarovnaný do čtyř sloupců. V tomto prostředí mohou být trochu problémy třeba s popisky obrázků nebo poznámkami pod čarou.

Toto je obsah prvního boxu o šířce 0,45-násobku šířky textu na stránce.

Toto je obsah druhého boxu širokého polovinu šířky textu na stránce.

Toto je obsah prvního boxu o šířce 0,45-násobku šířky textu na stránce.

Toto je obsah druhého boxu širokého polovinu šířky textu na stránce.

2.8 Křížové odkazy

Křížové odkazy se dělají velice jednoduše a rychle. Zde si ukážeme, jak se vytvářejí odkazy na kapitolu nebo konkrétní místo v dokumentu (číslo strany), a také na číslované prostředí, které jsme vytvořili pomocí příkazu `\newtheorem`.

Odkaz na položku číslovaného seznamu jsme viděli v kapitole 2.4 na straně 55, odkazy na obrázek a tabulku probereme v kapitole 2.10.5.

Při práci s odkazy se může objevit jedno nebo i více varovných hlášení (`LaTeX Warning: Reference 'nejakyodkaz' on page xx undefined ...`). Toho si nemusíme všímat, protože když si označíme místo, na které chceme odkazovat, je zaregistrováno až při překladu samotném, ale při adresování odkazu LaTeX vychází z informací získaných při předchozím překladu. Tedy při objevení varovných hlášení vztahujících se k odkazům prostě *znovu přeložíme*. Pokud se varovná hlášení objevují dále, je v názvu odkazu překlep.

Místo v dokumentu (záložku) pro jakýkoliv typ odkazu označíme příkazem

```
\label{nazev}
```

kde `nazev` je název označeného místa, který budeme v různých odkazech používat. Nepoužíváme v něm českou diakritiku (tedy píšeme „cesky“), ani mezery, tečky či většinu dalších oddělovačů. Naopak velmi často se v názvu objevuje dvojtečka jako užitečný oddělovač zkratky pro typ odkazu a zbytku názvu. Například

```
\label{kap:obsluhaprogramu},      \label{tab:obsluhaprogramu},
\label{obr:obsluhaprogramu},      \label{popis:obsluhaprogramu}
```

jsou různá místa v dokumentu (začátek kapitoly, obrázek, tabulka, popis), u kterých nám fantazie či spoléhání na vlastní paměť nestačily na různé názvy, ale pro přehlednost jsou odlišena jakousi předponou. Není to nutné, ale je to praktické a přehledné. Netřeba připomínat, že každý název musí být jedinečný – kdyby dvě místa byla pojmenovaná stejně, vznikl by jistý chaos při rozhodování, na které z nich se vlastně daným názvem odkázat.

Pokud chceme získat *číslo kapitoly*, ve které se nachází určité pojmenované místo, použijeme příkaz

```
\ref{nazev}
```

kde `nazev` je název pojmenovaného místa v příkazu `\label`. Pojmenované místo se může nacházet kdekoliv v kapitole, jen ne uvnitř prostředí, kde jsou čítače, na které se dá také odkazovat (třeba číselný seznam nebo prostředí označovaná čísly, jako je třeba v tomto dokumentu prostředí pro příklady nebo výpisy kódu).

Číslo stránky, na které se pojmenované místo nachází, dostaneme příkazem

```
\pageref{nazev}
```

Význam je tentýž jako u příkazu `\ref`. Oba příkazy můžeme kombinovat, což se hodí zejména, pokud pojmenované místo a odkaz jsou na různých stránkách:

```
...je v kapitole \ref{nazev} na straně \pageref{nazev}
```

Odkazy na prostředí označené číslem se dělají úplně stejně – za začátek prostředí umístíme příkaz `\label` a pro odkaz použijeme příkaz `\ref`, případně `\pageref`.

Příklad 2.10 Pokud je v preambuli dokumentu definováno prostředí `priklad` jako na začátku výpisu 2.23, můžeme se na ně jednoduše odkazovat podle textu dále.

Výpis č. 2.23

```
\newtheorem{tpriklad}{Příklad}[chapter]
% (ve třídě article by místo chapter bylo section)
\newenvironment{priklad}%
  {\begin{tpriklad}\rule{1pt}{0pt}\normalfont}%
  {\end{tpriklad}}
...
\label{kap:vytvoreniodkazu}
...
\begin{priklad}\label{pri:vytvoreniodkazu}
text nějakého příkladu
\end{priklad}
...
V~příkladu \ref{pri:vytvoreniodkazu} na straně
\pageref{pri:vytvoreniodkazu} v~kapitole
\ref{kap:vytvoreniodkazu} jsme mohli vidět, že ...
```

Příklad 2.10 text nějakého příkladu

...

V příkladu 2.10 na straně 70 v kapitole 2.8 jsme mohli vidět, že ...

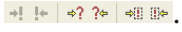
2.9 Jak na chyby

Snadnost nalezení chyb taky trochu závisí na editoru, který používáme. Po přeložení LaTeX vytváří soubor s příponou `LOG`, ve kterém jsou uvedeny veškeré informace o překladu včetně určení chyb. Tento soubor většina editorů přímo určených přímo pro LaTeX zobrazí v okně nebo v podokně, v případě programu TeXnicCenter jde o oblast u spodního okraje okna. Dovíme se zde především, zda jsou v dokumentu chyby, varování a přesahy:

- *chyba* (Error) je takové selhání, které (většinou) znemožní vytvořit výsledný soubor, třeba překlep v příkazu, u začátečníků je častou chybou „připojení“ příkazu bez parametrů k následujícímu slovu (například `\largenějakéslovo`),
- *varování* (Warning) je „menší“ selhání, které obvykle nezabrání vytvoření výsledného souboru, ale „něco není v pořádku“; často se jedná o použití příkazu na nevhodném místě (třeba příkaz pro matematický režim se objeví mimo matematická prostředí) nebo chybějící řez pro používaný font, tyto problémy LaTeX sám obvykle nějak vyřeší (provede náhradu nebo místo objektu vytvoří prázdné místo),
- *přesah* (Bad Box, vadný box) vzniká přesáhnutím objektu (včetně textu) tam, kde už nemá co dělat; jde většinou o slova, která se nedaří rozdělit na konci řádku tak, aby mezery mezi slovy zůstaly opticky vhodně malé či velké, ostatně přesáhnutí okraje stránky o milimetr většina lidí ani nepostřehne.

Chyby bychom měli rozhodně řešit, jen málokterá je zcela nedůležitá. Varování a přesahy je doporučeno řešit, ale není to už tak bezpodmínečně nutné. Některá varování a přesahy se obvykle neřeší nebo je řešení triviální:

- varování „There were undefined references.“ a „Label(s) may have changed ...“ se objeví vždy, když použijete příkaz pro pojmenování místa v dokumentu a případně toto pojmenování hned použijete pro křížový odkaz; stačí znovu provést překlad a tato varování zmizí (pokud nezmizí, máme buď v pojmenování nebo v křížovém odkazu překlep),
- varování „LaTeX Font Warning: Font shape ... not available ...“ se někdy objevuje, pokud nastavíme jiný než standardní font a chceme v něm používat řezy (některé fonty nejsou pro určité řezy definovány); tento problém LaTeX vyřeší přesně tak, jak píše dále v chybovém hlášení – na daném místě použije jiný (podobný) řez,
- hlášení o přesahu „Underfull `\hbox` ...“ znamená většinou problém s rozdělením slova; můžeme pomoci buď příkazem `\hyphenation` (viz nápověda), kde jako parametr použijeme problémová slova s naznačením možnosti rozdělení (na místech, kde lze slovo rozdělit, vložíme spojovník -, slova oddělujeme mezerou), nebo jednodušeji přímo ve slově, které vyvolává hlášení, na vhodném místě rozdělení vložíme dvojznak `\-` (například `větši\-nu`) – na konci řádku se slovo rozdělí, ale kdyby se náhodou posunulo dál od konce řádku, spojovník se nevypíše,
- při použití některých balíčků občas dostáváme varování a hlášení o vadných boxech, i když je víceméně všechno v pořádku; je to většinou způsobeno nekonzistentnostmi mezi balíčky, obvykle si těchto problémů nemusíme všímat.

Zvláště pokud je problémů více, hodí se možnost pohybu mezi nimi. U programu TeXnicCenter najdeme tlačítka na horním panelu nástrojů napravo od „překládacích tlačítek“ –  . Klepáním na tato tlačítka se pohybujeme ve zdrojovém souboru mezi chybami, varováními a přesahy a opravujeme. První dvě tlačítka nás v dokumentu přesouvají mezi chybami, další dvě mezi varováními a poslední dvě mezi přesahy.

Tímto způsobem se dostaneme vždy k *odstavci*, ve kterém je chyba, ne přímo ke konkrétnímu řádku. Najít konkrétní místo chyby nám pomůže výpis již zmíněného `LOG` souboru, ve kterém se přesouváme automaticky zároveň s přesouváním mezi chybami ve zdrojovém souboru.

2.10 Oživení textu

2.10.1 Vkládané obrázky

Pro vkládání již hotových obrázků jsou v LaTeXu dvě základní možnosti. První, použití balíčku `graphics`, je považována za jednodušší, ale jsme omezeni na vkládání obrázků s příponou `PS` a `EPS` při překladu přes formát `PS` a na obrázky s příponou `PDF` při překladu přímo do `PDF`. Druhou možností je balíček `graphicx`, kde můžeme při překladu přímo do `PDF` vkládat kromě jiného také obrázky s příponou `JPG` a `PNG`.

Nejdřív se podíváme na balíček `graphics`. Načítá se s nepovinným argumentem určujícím typ překladu. Pokud překládáme nejdřív do formátu `PS` a pak do `PDF` (v TEXnicCenter `LaTeX=>PS=>PDF`), bude příkaz pro načtení vypadat takto:

```
\usepackage[dvips]{graphics}
```

Při překladu přímo do `PDF` (v TEXnicCenter `LaTeX=>PDF`) použijeme jako parametr `pdftex`[8].

Hotový obrázek vložíme do dokumentu příkazem

```
\includegraphics{název souboru}
\includegraphics*[x1,y1][x2,y2]{název souboru}
```

Druhý tvar příkazu umožní vložit do dokumentu pouze výřez obrázku (zadáme nejdřív souřadnice levého dolního rohu a pak pravého horního rohu v souřadnicích původního obrázku, počátek soustavy je vlevo dole).

Příkaz	*Význam*
`\rotatebox{úhel}{objekt}`	otočí objekt o zadaný úhel
`\scalebox{hnásobek}[vnásobek]{objekt}`	roztázne objekt dle násobků
`\resizebox{šířka}{výška}{objekt}`	roztáhne objekt na zadané rozměry

Tabulka 2.6: Další příkazy balíčku `graphics`

První příkaz v tabulce 2.6 otočí objekt (cokoliv – text nebo třeba obrázek vložený pomocí `\includegraphics`) o zadaný úhel ve stupních. Druhý a třetí příkaz roztázne nebo smrští objekt podle zadaných parametrů.

V příkazu `\scalebox` zadáváme násobnost, kdežto v příkazu `\resizebox` cílové rozměry. Parametr `vnásobek` je nepovinný, když ho neudáme, použije se i pro vertikální násobnost také parametr `hnásobek` (tedy poměry stran zůstanou zachovány).

U příkazu `\resizebox` může být jedním z parametrů `šířka` a `výška` symbol `!`. Tento parametr se dopočítá tak, aby si obrázek zachoval poměry stran.

Příklad 2.11 Na výpisu 2.24 vidíme způsob použití výše uvedených příkazů. Obrázek je nejdřív vložen bez jakýchkoliv úprav a potom pozměněn zadáním výřezu a příkazy `\rotatebox`, `\scalebox` a `\resizebox`. Potom je příkaz `\rotatebox` použit na běžný text. Abychom mohli pracovat s více řádky najednou, je text vložen v příkazu `\parbox`. Obrázky jsou odděleny příkazem `\hfill`, aby byly rovnoměrně rozloženy na celou šířku stránky.

Výpis č. 2.24

```
\includegraphics{obr/slunecnice.eps} \hfill
\includegraphics*[10,0][80,100]{obr/slunecnice.eps} \hfill
\rotatebox{15}{\includegraphics*[30,30][80,70]{obr/slunecnice.eps}} \hfill
\scalebox{.5}{\includegraphics{obr/slunecnice.eps}} \hfill
\resizebox{5em}{!}{\includegraphics{obr/slunecnice.eps}} \hfill
\rotatebox{90}{\parbox{8em}{\large\itshape Obrázek\\slunečnice}}
```

[8] Parametr nemusíme uvádět, pokud ho napíšeme jako jeden z nepovinných parametrů příkazu `\documentclass`, například: `\documentclass[dvips,11pt,a4paper]{article}`.

Obrázek slunečnice

V příkladu 2.11 jsme si mohli všimnout, že náš obrázek je uložen v adresáři (složce) `obr`, který se pak nachází v tom adresáři, kde je překládaný soubor s příponou `TEX`.

Jak bylo výše naznačeno, přípona souboru s obrázkem musí být PS nebo EPS při překladu přes formát PS, a nebo PDF při překladu přímo do formátu PDF. . Konverzi můžeme provést v běžných lepších grafických editorech, pokud je máme k dispozici (jsou obvykle dost drahé), nebo pomocí některého z volně dostupných programů *ImageMagick*[9] a *Gimp*[10].

Gimp má grafické prostředí, tedy rozhodně není problém konverzi provést, ale je hodně rozsáhlý (také místem, které zabere na disku) a je považován za snadno dostupný ekvivalent Adobe Photoshopu. ImageMagick je souhrn malých nenáročných programů, pro konverzi budeme používat program `convert`. Konverze do formátu `EPS` probíhá takto:

```
convert původnísoubor.přípona novýsoubor.eps
```

Obrázek, se kterým jsme pracovali na straně 73, měl původně příponu `JPG`. Proto bylo třeba ho převést do `EPS` takto:

```
convert slunecnice.jpg slunecnice.eps
```

Pokud místo balíčku `graphics` použijeme balíček `graphicx`, získáme některé další možnosti (například u otáčení lze kterýkoliv bod obrázku zvolit za střed otáčení), tvar parametrů je trochu jiný, navíc při překladu přímo do `pdf` máme větší svobodu volby přípony souboru. Balíček se načítá stejným způsobem (tj. obvykle zadáváme `dvips` nebo `pdftex`).

Nejdůležitějším příkazem balíčku je příkaz obdobný předchozímu, jen parametry jsou trochu jiné:

```
\includegraphics[parametry oddělené čárkou]{soubor.přípona}
\includegraphics*[parametry oddělené čárkou]{soubor.přípona}
```

[9] Program ImageMagick zvládá nejrůznější úpravy (nejen) obrázků. Získáme ho například na adrese http://www.imagemagick.org/script/binary-releases.php, vybereme variantu pro Linux nebo Windows, pro Windows raději statickou variantu. Na uvedených WWW stránkách je také návod k instalaci a použití.

[10] České stránky programu Gimp včetně nejrůznějších užitečných informací najdeme například na stránkách http://www.gimp.cz/, při stahování volíme raději stabilní verzi.

V seznamu parametrů oddělených čárkami se píše vždy název parametru následovaný rovnítkem a hodnotou. Jako parametry můžeme použít:

- výřez obrázku, který se má vložit, zadáváme postupně čtyři čísla – souřadnice levého dolního rohu a pak pravého horního rohu; formát je trochu jiný než u předchozího: `bb= x_levý y_dolní x_pravý y_horní`, pro správnou funkci tohoto parametru použijeme variantu příkazu s hvěhdičkou,
- úhel otočení ve stupních: `angle=stupně`,
- šířka (obrázek se roztáhne nebo smrští), musíme zadat i jednotku (`pt`, `cm`, atd.): `width=šířka`,
- výška (podobně jako šířka): `height=výška`,
- zvětšení nebo zmenšení obrázku: `scale=poměr` (výchozí hodnota je 1, tedy zadáváme násobnost původní velikosti),
- atd. (viz dokumentace balíčku `graphicx`).

Například:

```
\includegraphics*[bb=10 0 80 100,scale=0.75]{slunecnice.eps}
```

V balíčku najdeme i další příkazy – `\rotatebox`, `\scalebox`, atd. Tyto příkazy mají podobný význam jako stejně pojmenované příkazy v balíčku `graphics`, ale parametry jsou trochu jiné a také máme k dispozici více možností nastavování vlastností zpracovávaných objektů.

2.10.2 Nákresy

Pro vytváření jednoduchých grafických objektů používáme prostředí `picture`. Používá se s udáním rozměrů „plátna“, které (trochu nezvykle) píšeme do *kulatých závorek* jako parametr prostředí, a to bez udání jednotky (standardně je jednotkou typografický bod).

```
\begin{picture}(šířka,výška)
...
\end{picture}
```

Dovnitř prostředí píšeme příkazy vykreslující jednotlivé objekty (rámečky, čáry–úsečky, vektory, kružnice, ovály, atd.). Každý objekt musíme umístit na určitou pozici příkazem

```
\put(x,y){objekt}
```

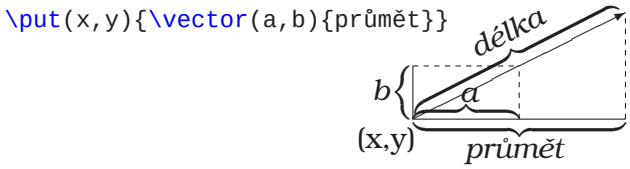
kde `x` a `y` jsou souřadnice, na které chceme `objekt` umístit. Bod `(0,0)` je v levém dolním rohu.

Dvojčíslí v kulatých závorkách se používá také v některých příkazech pro kreslení objektů. Někde znamená rozměry (šířku a výšku) objektu, jako třeba u rámečků (příkazy `\framebox`, `\dashbox` a `\makebox`), u oválů (příkaz `\oval`) délky vodorovné

a svislé osy, jindy zase určení úhlu sklonu, a to pro čáry a vektory (příkazy `\line` a `\vector`).

Čísla pro úhel sklonu a parametr pro výpočet délky čáry a vektoru se určují podle obrázku 2.7. Čísla `a` a `b` musí být *celá* čísla z intervalu $< -6, 6 >$ u čar a z intervalu $< -4, 4 >$ u vektorů. Různými kombinacemi těchto čísel dosáhneme různých sklonů.

Můžeme si všimnout, že nezadáváme přímo délku (kromě svislé čáry či vektoru, tam ano), ale *délku průmětu* na osu x.

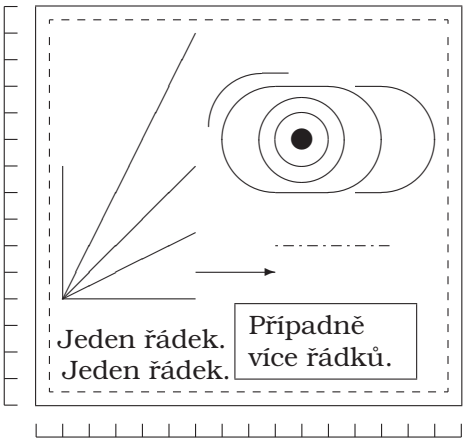


Obrázek 2.7: Určení sklonu čáry a vektoru

Příklad 2.12 Ukážeme si použití prostředí `picture` a vkládání různých typů objektů. Vysvětlení parametrů příkazů najdeme v tabulkách 2.7, 2.8 a 2.9 od strany 76. V příkazech si všimněte, kdy se u čísla udává jednotka.

Výpis č. 2.25

```
\begin{picture}(160,150)
\put(0,0){\framebox(160,150){}}
\put(5,5){\dashbox{3}(150,140){}}
\put(10,10){\small Jeden řádek.}
\put(10,20){\makebox(60,10){\small Jeden řádek.}}
\put(75,10){\framebox(68,28)%
  {\parbox{58pt}{\small Případně více řádků.}}}

\put(10,40){\line(0,1){50}}
\put(10,40){\line(1,0){50}}
\put(10,40){\line(1,1){50}}
\put(10,40){\line(1,2){50}}
\put(10,40){\line(2,1){50}}

\put(60,50){\vector(1,0){30}}

\put(100,100){\circle{20}}
\put(100,100){\circle{30}}
\put(100,100){\circle*{8}}

\put(100,100){\oval(60,40)}
\put(120,100){\oval(60,40)[r]}
\put(95,105){\oval(60,40)[lt]}

\multiput(90,60)(9,0){5}{\line(1,0){1}}
\multiput(93,60)(9,0){5}{\line(1,0){4}}
\end{picture}
```



Pro zpřehlednění obrázku jsou připojena „pravítka“ s díly po deseti bodech. Je to zároveň ukázka toho, že můžeme bez problémů kreslit i mimo oblast vyhrazenou v parametru prostředí `picture`. Pravítka jsou vykreslena těmito příkazy:

Výpis č. 2.26

```
\multiput(0,-12)(10,0){17}{\line(0,1){5}}     % vodorovné pravítko
\put(0,-12)\line(1,0){160}}
\multiput(-12,0)(0,10){16}{\line(1,0){5}}     % svislé pravítko
\put(-12,0)\line(0,1){150}}
```

To, jaké rozměry pro prostředí `picture` zadáme, určuje, kolik místa bude pro ně na stránce rezervováno. Protože není problém kreslit i mimo prostředí (to můžeme vidět na výpisu 2.26, kde jsou použity záporné souřadnice), záleží na nás, kolik místa chceme mít vynecháno. Dokonce není problém vytvořit prostředí `picture` s nulovými rozměry (v parametru `(0,0)`). Toho se využívá například při umístění obrázku do záhlaví nebo vytvoření „vodoznaku“ na pozadí stránky.

Příkaz	*Význam*
`\put(x,y){objekt}`	vloží objekt na dané souřadnice
`\multiput(x,y)(dX,dY){počet}{objekt}`	vloží objekt počet-krát, začíná na souřadnicích `(x,y)` a v každém kroku se posune o `dX` na ose X a o `dY` na ose Y
`\qbezier[počet](x1,y1)(x2,y2)(x3,y3)`	vykreslí Bezierovu křivku (začíná v prvním bodě, končí v třetím, druhý určuje zakřivení), v nepovinném parametru můžeme určit, že se nevykreslí celá křivka, ale pouze daný počet bodů z ní

Tabulka 2.7: Příkazy pro kreslení, které se píšou přímo do prostředí `picture`

Příkaz	*Význam*
`\line(poměrX,poměrY){průmět}`	čára (úsečka) o zadaném poměru stran a délce průmětu na osu X (u svislé čáry její délce), čísla pro poměr jsou celá čísla z intervalu $< -6, 6 >$
`\vector(poměrX,poměrY){průmět}`	vektor, jako u čáry, jen interval je $< -4, 4 >$
`\circle{průměr}`	kružnice o daném průměru
`\circle*{průměr}`	vyplněný kruh o daném průměru
`\oval(osaX,osaY)[část]`	ovál se zadanými délkami vodorovné a svislé osy, v nepovinném parametru můžeme určit část, která se má vykreslit (jedno nebo dvě písmena z `t`, `b`, `l`, `r`)
`\shortstack[poz]{řádky}`	řádky textu, `poz` je zarovnání vnitřku (`l`, `r`, `c`)

Tabulka 2.8: Příkazy objektů vkládaných do parametrů příkazů `\put` a `\multiput`

V tabulce 2.8 máme seznam některých příkazů, které vytvářejí různé objekty. Není to úplný seznam, protože zde můžeme používat prakticky (téměř) cokoliv, co jsme až dosud z LaTeXu probrali včetně formátovaného textu, vkládaných obrázků i různě upravovaných (oříznutí, rotace, apod.), dokonce i vnořené prostředí `picture`. To je výhoda zejména tehdy, když máme skupinu objektů, jejichž umístění je vzájemně závislé a zatím není definitivní. Když je umístíme do vnořeného prostředí `picture`, přesouváme celé toto prostředí bez „rozházení“ objektů v něm umístěných.

Příkaz	*Význam*
`\framebox(š,v)[poz]{obsah}`	rámeček s plným orámováním o daných rozměrech (šířka, výška), do obsahu lze dát cokoliv (i text nebo čáru), nepovinný parametr určuje zarovnání obsahu uvnitř (jedno nebo dvě písmena z `t`, `b`, `l`, `r`, `c`)
`\dashbox{element}(š,v)[poz]{obsah}`	rámeček s čárkovaným orámováním, platí pro něj totéž co pro `\framebox`, jen navíc v parametru `element` určujeme délku čárek
`\makebox(š,v)[poz]{obsah}`	rámeček bez orámování, používá se pro přesnější umísťování například textu

Tabulka 2.9: Příkazy pro rámečky v prostředí `picture`

Pro víceřádkový text se používá buď kombinace příkazů `\makebox` a `\parbox` (ten známe z kapitoly 2.7) nebo příkaz `\shortstack`. Na výpisu 2.27 vidíme drobné rozdíly v používání těchto možností. První možnost dovoluje snadnější manipulaci s textem samotným včetně výchozího zarovnání do bloku, řádky jsou zalamovány automaticky. Při použití druhé možnosti určujeme zarovnání řádků v nepovinném parametru, zalamování musíme provést ručně (ale zato nemusíme určovat šířku číselně) a po každém zalomení řádku je třeba znovu uvést formátovací příkazy.

Výpis č. 2.27

```
\begin{picture}(98,80)
\put(0,0){\framebox(98,80){}}
\put(1,45){\makebox(95,30){\parbox{90pt}{\small Toto je
nějaký text zarovnaný \itshape do více řádků.}}}
\put(3,3){\shortstack[l]{\small Toto je nějaký text\\
\small zarovnaný \itshape do více\\ \small\itshape řádků.}}
\end{picture}
```

Toto je nějaký text zarovnaný *do více řádků.*

Toto je nějaký text zarovnaný *do více řádků.*

Zde probírané možnosti prostředí `picture` nejsou plným popisem, podrobnější informace jsou například v nápovědě, [T31], [T43] (krásné příklady, v odkazech nahoře) a [T39]. Existují také balíčky, které rozšiřují možnosti prostředí `picture` – například `epic`, `eepic`, `pictex`, `xypic`, `pmgraph`, sada balíčků `pstricks` (zajímavé jsou tutoriály na [T44]).

2.10.3 Tabulátory

Pro tabulátory máme v LaTeXu prostředí `tabbing`. Uvnitř prostředí používáme především příkazy na nastavování tabulátorových zarážek a přecházení mezi nimi a samozřejmě příkazy pro přechod na nový řádek. Máme možnost také uložit pozice tabulátorů a kdykoliv je znovu využít, ale pouze v tomtéž prostředí `tabbing`. Toho se využívá především při změně nastavení zarážek a opětovném návratu k předchozímu nastavení.

Příkaz	*Význam*
`\=`	nastavení tabulátorové zarážky na této pozici
`\>`	přechod na další zarážku (jako bychom stiskli klávesu Tab)
`\'`	*předchozí* text je svým pravým okrajem (s určitou malou mezerou) zarovnán k předchozí zarážce
`\‘`	všechen další text až do symbolu konce řádku posune k pravému okraji, na řádku nesmí následovat žádné další tabulátorové příkazy
`\\`	přechod na nový řádek, v prostředí `tabbing` zůstávají tabulátorové zarážky nastaveny
`\kill`	pokud je tento příkaz použit na konci řádku místo `\\`, tento řádek se nevysází, ale všechny na něm nastavené tabulátorové zarážky zůstávají nastaveny
`\pushtabs`	uloží pozice všech zarážek do zásobníku tabulátorových zarážek
`\poptabs`	vyjme poslední uložené pozice zarážek ze zásobníku tabulátorových zarážek a nastaví je

Tabulka 2.10: Příkazy pro tabulátory

Příkazy `\pushtabs` a `\poptabs` jsou párové (ke každému `\pushtabs` musí být jeden `\poptabs`) a můžeme je vnořovat podobně jako symboly hranice skupiny { }.

Obvyklý způsob použití je nastavení zarážek pomocí `\=` na zvláštním řádku, kde pro jednotlivé části použijeme vždy z daného „sloupce“ nejširší část přes všechny řádky (případně ještě rozšíříme) a na konci tohoto zvláštního řádku napíšeme příkaz `\kill`. Tento řádek bude sloužit pouze pro nastavení, nevysází se. Další řádky již píšeme běžným způsobem, pro pohyb k dalším zarážkám používáme příkazy `\>`, `\'` a `\‘` a řádky končíme příkazem `\\`.

Příklad 2.13 Zarovnání k tabulátorovým zarážkám si ukážeme na fragmentu úlohy ze strany I. Text je zarovnán k zarážce vlevo, čísla vpravo.

Výpis č. 2.28

```
\begin{tabbing}
Počet xx \= zajíců xx 126 \= x \kill
Počet \> sobů \> 126 \'\\
\> lišek \> 12 \'\\
\> zajíců \> 54 \'\\[.8ex]
\textbf{Celkem zvířat}\>\> \textbf{192}\'
\end{tabbing}
```

Počet	sobů	126
	lišek	12
	zajíců	54
Celkem zvířat		**192**

Prostředí `tabbing` nelze vnořovat, kromě případu, kdy je „vnořené“ prostředí uzavřeno do příkazu `\parbox`.

Stejného vizuálního efektu, jako pomocí prostředí `tabbing`, dosáhneme například použitím tabulek (viz kapitola 2.10.4) nebo skládáním prostředí `minipage` či příkazů `\parbox`. Zde pak máme například možnost používat pružné čáry plné či tečkované, u tabulek také podtržení a shrnutí více buněk. Ve výpisu 2.29 vidíme kombinaci dvou použití příkazu `\parbox` a pružné čáry.

Výpis č. 2.29

```
\parbox{3em}{ Počet}
\parbox[t]{8em}{%
  sobů\dotfill 126\\
  lišek\dotfill 12\\
  zajíců\dotfill 54 }
```

Počet sobů 126
lišek 12
zajíců 54

2.10.4 Tabulky

Pro sazbu tabulek používáme prostředí `tabular` (pozor, neplést si s prostředím pro tabulátory) s následujícím způsobem použití:

```
\begin{tabular}{určení sloupců} ... \end{tabular}
```

V parametru prostředí musíme určit počet sloupců s jejich zarovnáním a oddělovače.

Zarovnání sloupců se zadává písmeny `l`, `r`, `c`, `p` (jeden řádek zarovnaný doleva, doprava, vycentrovaný, a nebo odstavec zarovnaný do bloku), při použití `p` určujeme šířku odstavce v parametru, například `p{2cm}` je sloupec široký 2 centimetry, ve kterém jsou odstavce zarovnány do bloku.

Oddělovač může být buď čára (zadáme symbol `|`) nebo jakýkoliv jiný symbol včetně matematických. Dvojitou čáru zadáme dvěma symboly pro čáru za sebou, tedy `||`, jiný oddělovač než čáru určíme příkazem `@{znaky}`. Když oddělovač nezadáme vůbec, sází se pro oddělení sloupců mezera, pokud nechceme ani tu mezeru, zadáme jako oddělovač `@{}`.

Protože oddělovače oddělují sloupce a sloupce jsou určeny svým zarovnáním, parametr `určení sloupců` se zadává například takto:

`{|l|c|p{.3\textwidth}|r|}` první sloupec je zarovnán vlevo, druhý na střed, třetí je odstavec o šířce přibližně třetiny stránky a čtvrtý je zarovnán doprava, mezi nimi jsou jako oddělovače použity čáry, stejně jako na stranách celé tabulky,

`{|l||r@{$+$}r@{$+$}r@{$=$}r|}` první sloupec je zarovnán doleva, pak následuje dvojitá čára, zbylé sloupce jsou zarovnány doprava (zřejmě pro čísla) a mezi nimi jsou použity jako oddělovače symboly + a =,

`{l|r@{,}l}` první sloupec je zřejmě použit pro běžný text, další dva dohromady tvoří reálné číslo s desetinnou čárkou (druhý sloupec zaznamenává celou část, třetí desetinnou část).

Vodorovnou čáru tvoříme příkazem `\hline`, dvojitou čáru zdvojením tohoto příkazu. Vodorovnou čáru přes daný počet sloupců získáme příkazem `\cline{od-do}`, kde zadáme čísla sloupců, mezi kterými má čára vést.

Více sloupců na jednom řádku (buněk) sloučíme příkazem

```
\multicolumn{počet sloupců}{zarovnání}{obsah buňky}
```

Uvnitř tabulky používáme k oddělení sloupců příkaz `\\` případně následovaný příkazy `\hline` či `\cline`, buňky na řádku oddělujeme symbolem `&`.

Příklad 2.14 Použití základních možností tabulky si ukážeme na jednoduchých příkladech vycházejících z výpisu 2.28 na straně 78. Ve výpisu 2.32 je v zarovnání na okrajích příkaz `@{}`, aby podtrhávací čára „nepřečnívala“. Bez použití tohoto příkazu by se místo okrajové čáry vložila mezera, jenže my nechceme ani tu mezeru.

Výpis č. 2.30

```
\begin{tabular}{|ll|r|}\hline
Počet & sobů & 126\\
  & lišek & 12 \\
  & zajíců & 54 \\ \hline\hline
\multicolumn{2}{|l|}{\bfseries Celkem zvířat}&
  \bfseries 192\\ \hline
\end{tabular}
```

Počet	sobů	126
	lišek	12
	zajíců	54
Celkem zvířat		**192**

Výpis č. 2.31

```
\begin{tabular}{|l|r|}\hline
\bfseries Zvíře & \bfseries Počet\\ \hline\hline
sob & 126\\ \hline
liška & 12 \\ \hline
zajíc & 54 \\ \hline\hline
\bfseries Celkem zvířat & \bfseries 192\\ \hline
\end{tabular}
```

Zvíře	**Počet**
sob	126
liška	12
zajíc	54
Celkem	**192**

Výpis č. 2.32

```
\begin{tabular}{@{}lp{7.7em}@{}}
Počet & sobů \dotfill 126\\
  & lišek \dotfill 12 \\
  & zajíců \dotfill 54 \\[.3ex] \hline
\multicolumn{2}{@{}l@{}}{\bfseries Celkem zvířat
  \dotfill 192}
\end{tabular}
```

Počet sobů 126
lišek 12
zajíců 54
Celkem zvířat 192

Příklad 2.15 V tomto příkladu se podíváme na zarovnání reálných čísel podle desetinné čárky. Celou část zarovnáme doprava, desetinnou doleva a před sečtením podtrhneme.

Výpis č. 2.33

```
\begin{tabular}{lr@{,}l}
Výpočet: & 1\,284&52\\
&716&4\\
&0&977\\
&35&0\\ \cline{2-3}
&2\,036&897
\end{tabular}
```

Výpočet: 1 284,52
716,4
0,977
35,0
2 036,897

Příklad 2.16 Ukážeme si vytvoření vlastních oddělovačů na jednoduchém „sčítacím“ příkladu. Ve výpisu 2.34 je kód příkladu, v tabulce 2.11 pak výsledek.

Výpis č. 2.34

```
\begin{tabular}{l||c@{~+~}c@{~+~}c@{~=~}c}
\bfseries Jméno & Příklad 1 & Příklad 2 & Příklad 3 & Celkem\\ \hline\hline
\itshape Maximum: &\itshape 6&\itshape 7&\itshape 7&\itshape 20\\ \hline\hline
Blažková & 3 & 7 & 4 & 14\\ \hline
Hrubý &2&3&1&6\\ \hline
Novák & 4 & 5 & 7 & 16\\ \hline
\dots
\end{tabular}
```

Jméno	Příklad 1	+	Příklad 2	+	Příklad 3	=	Celkem
Maximum:	*6*	+	*7*	+	*7*	=	*20*
Blažková	3	+	7	+	4	=	14
Hrubý	2	+	3	+	1	=	6
Novák	4	+	5	+	7	=	16
...							

Tabulka 2.11: Ukázka tabulky s vlastními oddělovači

Ukázali jsme si pouze základní možnosti tvorby tabulek. Možností je více a navíc je můžeme rozšířit použitím různých balíčků, například `tabularx`, `supertabular`, `slashbox`, apod. Obvykle jde především o možnost vytvářet tabulky zasahující na více stran (zde ukázané tabulky mohou být pouze na jedné straně). Podrobnosti viz například [T31] nebo na prakticky každém odkazu ze seznamu literatury tohoto dokumentu vedoucímu na některou učebnici LaTeXu.

2.10.5 Plovoucí prostředí

V tomto dokumentu jsme si mohli všimnout způsobu vkládání obrázků a tabulek. Tyto objekty jsou obvykle pojmenované (mají popisek s číslem), lze se na ně odkazovat a jsou umístěny tak, že sice někdy nebývají přesně tam, kde se o nich píše, ale v případě, že by přes ně mohl vést stránkový zlom (konec stránky), přesunou se na další stránku bez toho, aby místo nich na předchozí stránce zůstalo kus místa nevyužitého.

Tuto funkci v LaTeXu plní *plovoucí prostředí*. Nazývá se tak, protože „plave“ na stranách tak, aby se vyhnulo stránkovému zlomu a umístilo se tak, jak si autor přeje (pokud je to možné). Rozlišujeme dvě základní plovoucí prostředí – pro tabulky je to `table` a pro obrázky a jiné grafické objekty je to `figure`. Používají se stejně:

```
\begin{table}[umístění]
\centering     % vycentrování

... tady bude tabulka,
třeba prostředí tabular

\caption{popisek tabulky}
\label{pojmenování pro odkazy}
\end{table}
```

```
\begin{figure}[umístění]
\centering     % vycentrování

... tady bude objekt,
třeba prostředí picture

\caption{popisek objektu}
\label{pojmenování pro odkazy}
\end{figure}
```

V parametru `umístění` určujeme písmeny `h`, `t`, `b`, `p`, kde se má na stránce objekt umístit, pokud to půjde (na místě, na horním okraji stránky, dolním okraji a nebo na zvláštní stránce pouze pro plovoucí prostředí, obvykle na konci kapitoly nebo dokumentu). Píšeme více možností, LaTeX je zkouší zleva – když se nepovede umístit prostředí na místě, posune ho na horní okraj následující stránky apod., například `[htbp]`.

Tabulka 2.11 na straně 81 byla umístěna tímto kódem:

Výpis č. 2.35

```
\begin{table}[htb]
\centering
\begin{tabular}[...
...
\end{tabular}
\caption{Ukázka tabulky s~vlastními oddělovači}
\label{tab:scitani}
\end{table}
```

Na tuto tabulku se můžeme odkázat takto: `v tabulce \ref{tab:scitani} na straně \pageref{tab:scitani} ...` – v tabulce 2.11 na straně 81 ...

Příklad 2.17 Umístíme obrázek slunečnice s popiskem. Použijeme přitom obrázek, který jsme používali v kapitole 2.11 na straně 73.

Obrázek 2.8: Slunečnice

Kdyby tento obrázek byl uprostřed strany, byl by vložen a umístěn tímto kódem:

Výpis č. 2.36

```
\begin{figure}[htb]
\centering
\includegraphics{obr/slunecnice.eps}
\caption{Slunečnice}
\label{obr:slunecnice}
\end{figure}
```

Uvnitř plovoucího prostředí nemusí být jen jedna tabulka či obrázek. Můžeme tam vložit prakticky cokoliv (i text) a v jakémkoliv počtu, pak se objekty naskládají vedle sebe. Například v kapitole o MS Wordu je často používáno prostředí `figure`, ve kterém je kromě obrázku vloženého příkazem `\includegraphics` také prostředí `picture` o velmi malých rozměrech pro popisování jednotlivých částí vloženého obrázku.

Stejně jako u jiných témat, také u plovoucích prostředí najdeme mnoho rozšíření v podobě balíčků. Například balíčky `wrapfig`, `picins` a `floatflt` nám umožňují vkládat plovoucí prostředí do vícesloupcového textu, prostředí `minipage` či příkazu `\parbox`, nebo je obtékat.

Balíček `picins` má bohužel dokumentaci jen v němčině, ale některé jeho možnosti při vkládání objektů jsou pěkně popsány v [T31], v nápovědě nebo na internetu (stačí na `http://google.com` zadat heslo „picins documentation“ nebo jen „picins“). Zajímavý je především příkaz `\parpic`, který kromě zajištění obtékání objektu do něj vloženého umí například také kolem objektu vytvořit rámečky různých tvarů včetně oválného.

Balíček `floatflt` obsahuje prostředí `floatingfigure` a `floatingtable`, balíček `wrapfig` obsahuje prostředí `wrapfigure` a `wraptable`. Jsou podobné, ale o něco více možností nabízí balíček `wrapfig`. Obrázek 2.8 byl ve skutečnosti vložen tímto kódem (je načten balíček `wrapfig`):

Výpis č. 2.37

```
\begin{wrapfigure}[9]{l}{115pt}   % přes 9 řádků, vlevo, šířka 115 bodů
\centering
\includegraphics{obr/slunecnice.eps}
\caption{Slunečnice}
\label{obr:slunecnice}
\end{wrapfigure}
Kdyby tento obrázek byl uprostřed strany ...
```

Balíčky `subfig` a `subfigure` dovolují členit plovoucí prostředí na menší pojmenované části, tedy máme v jednom prostředí více obrázků a každý má vlastní popisek. Pomocí balíčku `subfloat` vytváříme víceúrovňové číslování různých plovoucích prostředí, `captcont` rozděluje dlouhá plovoucí prostředí na více stránek a `ccaption` dovoluje kromě jiného vytvářet i nečíslované nebo vícejazyčné popisky.

2.10.6 Boxy

Box (rámeček) je oblast stránky, se kterou lze zacházet do určité míry samostatně. Existuje více různých druhů boxů a každý má své specifické vlastnosti. S některými jsme se už setkali, kromě příkazu `\fbox` a `\parbox` především v prostředí `picture` v kapitole 2.10.2 (jsou v tabulce 2.9 na straně 77)[11]. Stejně pojmenované boxy, ale s trochu jinou syntaxí (způsobem použití) používáme i mimo nákresy v textu, a také některé další.

Z boxů již známe ten, do kterého dáváme objekty (včetně textu), které chceme srovnat na určitou šířku, objekty mohou být i na více řádcích:

```
\parbox[zarovnání od předchozího]{šířka}{obsah}
```

Další boxy používáme, když potřebujeme text či jiné objekty vhodně zarovnat či umístit, případně orámovat

```
\makebox[šířka][zarovnání]{obsah}
\mbox{obsah}
\framebox[šířka][zarovnání]{obsah}
\fbox{obsah}
```

Příkaz `\makebox` prostě zarovná (`c`, `l`, `r`) text v dané šířce (třeba vycentruje na tuto šířku). Zjednodušenou variantou je příkaz `\mbox`, kde nepotřebujeme žádné nepovinné parametry, používáme ho třeba tehdy, když některé slovo nebo několik slov určitě nechceme rozdělit na konci řádku, chceme je udržet celé „v kuse“. Další dva příkazy mají podobný význam, jen navíc přidávají rámeček kolem svého obsahu.

Nevýhodou těchto příkazů je, že nedokážou pracovat s více řádky. To se řeší vnořením obsahu do příkazu `\parbox` nebo do prostředí `minipage`, například takto:

```
\fbox{\parbox[zarovnání od předchozího]{šířka}{obsah}}
```

Další typ boxu slouží k vertikálnímu posouvání objektů. Někdy potřebujeme část textu, obrázek či cokoliv jiného posunout nahoru nebo dolů, pak použijeme tento příkaz:

```
\raisebox{zvýšení}{obsah}
```

Parametr `zvýšení` je délka, o kolik se má `obsah` posunout nahoru, když je délka záporná, posouvá se dolů. Opět při práci s více řádky musíme použít kombinaci s příkazem `\parbox` nebo prostředím `minipage`.

V boxech můžeme jako šířku či jinou míru použít také některé délkové registry. Často se používají například `\width` (šířka obsahu), `\height` (výška obsahu od základní čáry řádku nahoru), `\depth` (hloubka obsahu od základní čáry řádku dolů), `\totalheight` (celková výška obsahu – součet předchozích dvou).

[11] Pozor na rozdíl mezi stejně pojmenovanými boxy pro text a pro prostředí `picture` – mají jiné parametry!

Příklad 2.18 Vyzkoušíme si použití příkazu `\framebox` s nepovinným parametrem, jeho kombinaci s příkazem `\makebox` a také zvyšování a snižování textu. Všimněte si u druhého boxu zadání dvojnásobku šířky obsahu. Pro orámování výsledku kódu výpisu je použita kombinace příkazu `\fbox` a prostředí `minipage`.

Výpis č. 2.38

```
Píšeme orámovaný text na
\framebox[10em]{danou šířku}
10 čtverčíků, a další v boxu širokém dvojnásobek
tohoto \framebox[2\width]{textu}.

Máme orámované písmenko \fbox{j} a pak písmenko
\fbox{\makebox[\totalheight]{j}}
ve čtvercovém rámečku.

% zvyšování a snižování objektů:
Píšeme \raisebox{3pt}{zvýšený,} normální a také
\raisebox{-3pt}{snížený text.}
```

Píšeme orámovaný text na danou šířku 10 čtverčíků, a další v boxu širokém dvojnásobek tohoto textu.

Máme orámované písmenko j a pak písmenko j ve čtvercovém rámečku.

Píšeme zvýšený, normální a také snížený text.

Boxy lze také ukládat a znovu používat, to se týká i boxů v prostředí `picture`. Podrobnosti najdeme například v nápovědě nebo v [T31].

2.10.7 Barvy

Podporu barev přidáme balíčkem `color` s nepovinným parametrem `dvips` nebo `pdftex` podle toho, jestli chceme překládat přes formát `DVI` a `PS` (`LaTeX=>DVI` nebo `LaTeX=>PS=>PDF`) nebo přímo do `PDF` (`LaTeX=>PDF`). Máme možnost používat předdefinované barvy (s anglickými názvy) pro text, pozadí textu, boxy i kreslení, a také vytvářet si vlastní barvy.

Následující příkazy slouží k nastavení určité barvy pro úsek textu, a to pro samotný text, pozadí textu nebo obojí najednou:

```
\textcolor{barva}{text}
\colorbox{barva pozadí}{text}
\fcolorbox{barva rámečku}{barva pozadí}{text}
```

Existují i příkazy platné od místa jejich uvedení až do konce skupiny, slouží pro nastavení barvy textu (nebo barvy pro kreslení v prostředí `picture`) nebo pro nastavení barvy pozadí celé stránky (změna barvy pozadí stránky se projeví až po překladu do formátu `PS` nebo `PDF`):

```
\color{barva}
\pagecolor{barva pozadí}
```

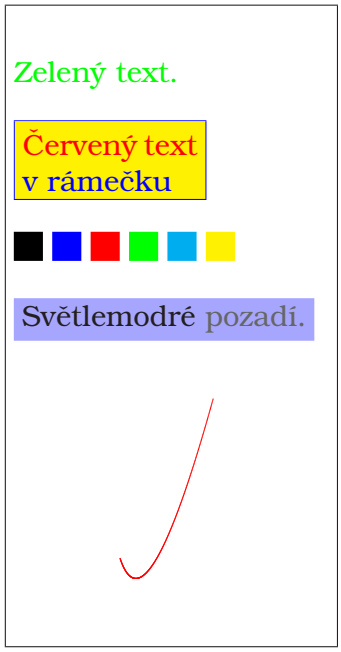
Novou barvu vytvoříme následujícím příkazem, ve kterém zadáme název nové barvy, číselnou specifikaci a pokud možno také typ (obvykle se použije zkratka `rgb` pro barevnou kombinaci červené, zelené a modré nebo nebo `gray` pro odstín šedé):

```
\definecolor{název barvy}{typ barvy}{specifikace}
```

Příklad 2.19 Ukážeme si použití příkazů pro barvení textu, jeho pozadí a kreslení, a také vytvoření vlastní barvy.

Výpis č. 2.39

```
\textcolor{green}{Zelený text.}

\fcolorbox{blue}{yellow}{\parbox{6em}{%
  \textcolor{red}{Červený text} %
  \color{blue} v~rámečku}}

\def\ctverecek{\rule{1em}{1em}}
\textcolor{black}{\ctverecek}
\textcolor{blue}{\ctverecek}
\textcolor{red}{\ctverecek}
\textcolor{green}{\ctverecek}
\textcolor{cyan}{\ctverecek}
\textcolor{yellow}{\ctverecek}

% barvy se definují čísly z intervalu <0,1>,
%  vyšší číslo znamená světlejší barvu:
\definecolor{svmodra}{rgb}{.65,.65,1}
\definecolor{seda}{gray}{.4}
\colorbox{svmodra}{Světlemodré \color{seda} pozadí.}

\begin{picture}(1,100)
\color{red}    % odteď se kreslí červenou
\qbezier(40,30)(50,0)(75,90)
\end{picture}
```



2.10.8 Hypertextové odkazy

Hypertextové odkazy vytváříme pomocí balíčku `url` nebo `hyperref`.

Použití balíčku `url` je jednoduché. Načteme ho příkazem `\usepackage` (není třeba zadávat žádné další parametry) a hypertextový odkaz vložíme příkazem

```
\url{adresa}
```

Zadaná adresa je vypsána neproporcionálním písmem („psací stroj“) a jsou zobrazeny znaky, které by jinak bylo nutné zobrazovat pomocí příkazů (například vlnovka nebo zpětné lomítko). Pokud si přejeme ještě dále adresu formátovat, musíme to udělat ručně. Například modrou barvu a podtržení nastavíme takto:

```
{\color{blue}\underline{\url{http://nejaka.adresa.cz}}}
```

Vypadá to jako hypertextový odkaz, ale nechová se to tak. Pro dokument, který se má tisknout, je to dostačující, ale pokud chceme, aby byl použitelný i v elektronické podobě včetně možnosti používat odkazy, musíme volit jiné řešení. Existuje program `dvipdfm`, který dokáže vyrobit dokument s těmito vlastnostmi, ale je nutno znát parametry, se kterými se má spustit.

Jednodušším řešením je použití balíčku hyperref, hypertextové odkazy fungují až v souboru PDF. Je důkladně popsán v dokumentaci, kterou snadno najdeme (viz kapitola 2.6.2). Je spíše určen pro překlad přímo do PDF (LaTeX=>PDF), ale lze ho použít i při překladu přes DVI a PS (LaTeX=>PS=>PDF) s tím, že některé volby budou dělat trochu problémy.

Balíček hyperref nabízí mnoho voleb (některé se vlastně ani netýkají hypertextových odkazů), i proto stojí za to prostudovat si dokumentaci. Volby se zadávají buď při načítání balíčku příkazem \usepackage, nebo v příkazu \hypersetup.

Při překladu přes PS můžeme balíček načíst například s těmito volbami:

```
\usepackage[dvips,colorlinks,linkcolor={blue},%
  citecolor={green},urlcolor={blue}]{hyperref}
```

Volba dvips je nutná při překladu přes DVI a PS, volba colorlinks znamená, že se odkazy mají zobrazovat barevně (byly by však aktivní i bez barevného zvýraznění), další volby určují barvy pro jednotlivé druhy odkazů. V našem případě jsou odkazy vedoucí dovnitř dokumentu (třeba na určitou stranu, kapitolu nebo v obsahu) modré, odkazy do seznamu literatury zelené a odkazy „ven“ (internetové) modré.

Názvy barev (předdefinovaných) můžeme používat i bez načtení balíčku color, ale pokud chceme vytvářet vlastní barvy, musíme tento balíček do dokumentu načíst (a barvu samozřejmě definovat ještě před jejím případným použitím ve volbách balíčku hyperref).

Balíček hyperref také umožňuje vytvářet záložky podle kapitol dokumentu, tyto záložky se pak zobrazí v některých prohlížečích (například Adobe Reader) vlevo vedle dokumentu na záložce Záložky. Při překladu přímo do PDF nebývají problémy, ale pokud překládáme přes PS, musíme zajistit, aby se správně zobrazovaly české znaky (tím zároveň vyřešíme i problémy s češtinou při používání aktivních odkazů). Použijeme balíček inputenc s vhodnou volbou (například latin2) a u balíčku hyperref volbu unicode.

Celkově po vyřešení vlastních barev, češtiny a záložek (včetně čísel kapitol) může načtení balíčku vypadat podle výpisu 2.40.

Výpis č. 2.40

```
\usepackage[dvips]{color}
  \definecolor{barvaodkaz}{rgb}{.1,.1,.4}
\usepackage[latin2]{inputenc}
\usepackage[dvips,bookmarksnumbered,unicode,colorlinks,%
  linkcolor={barvaodkaz},citecolor={green},urlcolor={blue}]{hyperref}
```

Když máme načten balíček hyperref, odkazy „dovnitř dokumentu“ (obsah, odkazy na seznam literatury, na tabulky apod.) jsou označeny a aktivovány automaticky, odkazy na WWW stránky označíme například následujícími příkazy (druhý příkaz použijeme, pokud nechceme do dokumentu vložit celou adresu, ale odkaz má být funkční):

```
\url{http://nejaka.adresa.cz}
\href{http://nejaka.adresa.cz}{slovní odkaz}
```

2.11 Seznam literatury a rejstřík

Seznam literatury se vkládá na dané místo prostředím `thebibliography`, jehož parametrem je „nejširší číslo“ použité při číslování, obvykle se použije 99. Položky začínají příkazem `\bibitem`, kde jako povinný parametr zadáváme název pro křížové odkazy, jako nepovinný parametr můžeme použít slovní pojmenování položky (místo číslování). Ve výpisu 2.41 vidíme způsob vytvoření číslovaného seznamu literatury, za výpisem je vysázený seznam.

Výpis č. 2.41

```
\begin{thebibliography}{99}
\bibitem{linuxnoviny} \textsc{Dvořák}, V. \emph{WIN95 + RH6.2 = 10GB HDD.}
  Linuxové noviny $[$online$]$. 2001\\
  URL: \url{http://www.linux.cz/noviny/2001-08/clanek05.html}
  $[$cit. 18. 10. 2006$]$
\bibitem{vaznyfeynman} \textsc{Feynman}, R. P. \emph{To snad nemyslíte vážně!}
  Praha, Mladá Fronta, 1989.
\bibitem{libgaher} \textsc{Gahér}, F. \emph{Logické hádanky, hlavolamy
  a paradoxy.} Bratislava, Iris, 1997.
\end{thebibliography}
```


[1] DVOŘÁK, V. *WIN95 + RH6.2 = 10GB HDD.* Linuxové noviny [online]. 2001 URL: http://www.linux.cz/noviny/2001-08/clanek05.html [cit. 18. 10. 2006]

[2] FEYNMAN, R. P. *To snad nemyslíte vážně!* Praha, Mladá Fronta, 1989.

[3] GAHÉR, F. *Logické hádanky, hlavolamy a paradoxy.* Bratislava, Iris, 1997.

Na položky seznamu se odkazujeme příkazem `\cite[text]{odkaz}`. Povinný parametr `odkaz` je odkazem na položku, v nepovinném parametru můžeme zadat třeba číslo stránky nebo kapitolu. Podle výpisu 2.41 můžeme vytvořit třeba tyto odkazy:

```
\cite{linuxnoviny}
\cite[s. 25]{vaznyfeynman}
```

Položky seznamu literatury můžeme mít pojmenovány i slovně, obvykle se používá příjmení autora s rokem vydání, při více knihách jednoho autora ve stejném roce přidáváme ještě písmenka a, b, ... Použití vidíme na výpisu 2.42.

Výpis č. 2.42

```
\begin{thebibliography}{99}
...
\bibitem[Feynman89]{vaznyfeynman} \textsc{Feynman}, R. P. \emph{To snad nemyslíte
vážně!} Praha, Mladá Fronta, 1989.
...
\end{thebibliography}
```

Do seznamu literatury dostaneme:

```
...
[Feynman89] FEYNMAN, R. P. To snad nemyslíte vážně! Praha, Mladá Fronta, 1989.
...
```

Odkaz ve tvaru `\cite{vaznyfeynman}` se vysází jako [Feynman89], odkaz s nepovinným parametrem `\cite[s. 25]{vaznyfeynman}` jako [Feynman89, s. 25].

Pro vysázení *rejstříku* potřebujeme balíček `makeidx` nebo `csindex`. Ten první je určitě ve všech distribucích, ten druhý bude asi třeba doinstalovat.

Tedy načteme balíček, dále ještě v preambuli (třeba hned za příkazem pro načtení balíčku `makeidx`) napíšeme příkaz `\makeindex`, kterým dáme najevo, že se má vytvářet pomocný soubor pro vytvoření rejstříku. Na místě, kde chceme mít rejstřík vysázený, dáme příkaz `\printindex`.

Musíme také označit místa a pojmy, které chceme do rejstříku zahrnout. Na daném místě dokumentu pak použijeme příkaz `\index{pojem}`. Máme k dispozici i další možnosti strukturování rejstříku, najdeme je například v nápovědě. Užitečný může být také balíček `index`.

2.12 Dokumentové seznamy podle stylů

Vytváření dokumentových seznamů je v LaTeXu poměrně dobře automatizované. Technicky probíhá tak, že existují speciální soubory s tím, co má v daném seznamu být (s příponou `toc` pro obsah, `lot` pro seznam tabulek, `lof` pro seznam obrázků, `idx` pro rejstřík, apod.), a při překladu se údaje berou právě z těchto souborů.

Seznamy se vytvářejí z toho, co bylo vygenerováno při minulém překladu. Proto je nutné při změně údajů v souborech (třeba přidání kapitoly) *provést překlad alespoň dvakrát*, aby seznamy byly aktuální, případně i vícekrát (když se vložením nebo rozšířením seznamu mění rozložení stránek, například obsah se rozšíří na další stránku).

Obsah, *seznam tabulek* a *seznam obrázků* vložíme na dané místo příkazy

```
\tableofcontents
\listoftables
\listoffigures
```

Dokonce tyto příkazy mohou následovat přímo za sebou (nemusíme mít samozřejmě všechny, jen ty, které potřebujeme), ale pokud používáme třídu `article` nebo z ní odvozenou, je vhodné mezi ně vložit příkaz `\newpage`.

Do obsahu se automaticky neřadí nečíslované nadpisy. Protože však občas potřebujeme mít v obsahu kapitolu, kterou nechceme číslovat, nebo dokonce chceme přidat řádek, který se ke kapitolám vůbec nevztahuje, máme k dispozici dva příkazy, a to příkaz `\addcontentsline`, který používáme při vložení „běžného“ nadpisu do určité úrovně obsahu (zadáváme úroveň slovem `chapter`, `section`, `subsection`

apod.), a příkaz \addtocontents sloužící ke vložení prakticky jakéhokoliv textu do obsahu (můžeme také používat formátovací příkazy, ale ne úplně libovolně).

Výpis č. 2.43

```
\chapter*{Úvod}
\addcontentsline{toc}{chapter}{Úvod}
...
\chapter*{Přílohy}   % nadpis můžeme i formátovat nebo oživit obrázkem, což jsme
\addtocontents{\bigskip\noindent\bfseries{Přílohy}}              % zde nepoužili
\thispagestyle{empty}
...
\newpage
\addcontentsline{toc}{chapter}{Rejstřík}
\printindex
```

Ve výpisu 2.43 máme nejdřív vložení nečíslované kapitoly s názvem Úvod do obsahu (zkratka `toc`) na první úroveň (`chapter`, ve třídě `article` bychom zde použili `section`). Dále vkládáme do obsahu řádek oddělující přílohy od běžných kapitol, ale nechceme na tomto řádku obsahu číslo stránky, které jsme ostatně odbourali i na stránce samotné, proto použijeme příkaz \addtocontents a sami si určíme, co a jak bude zobrazeno (inspiraci můžeme hledat přímo v souboru s příponou `toc` stejně nazvaném jako samotný dokument).

Třetí zásah do obsahu je vložení odkazu na kapitolu s rejstříkem. Tato kapitola se sice v dokumentu správně nazve, ale není zahrnuta do rejstříku. Nemůžeme použít přesně stejný postup jako u kapitoly Úvod, protože při použití příkazu \addcontentsline až za příkazem načítajícím rejstřík by se do obsahu vložilo číslo poslední strany rejstříku místo strany s jeho začátkem, proto příkaz umístíme před příkaz pro načtení rejstříku, ale aby se nevložilo číslo o 1 menší než chceme, musíme ještě použít příkaz \newpage.

Zatím jsme v příkazech \addcontentsline a \addtocontents používali zkratku `toc`, která znamenala, že pracujeme s obsahem. Tyto příkazy můžeme použít také na seznam tabulek a seznam obrázků, ale místo `toc` musíme použít `lot` (tabulky) nebo `lof` (obrázky).

2.13 Rozměry stránky

Protože evropské dokumenty mají obvykle poněkud jiné rozměry potištěné oblasti stránky než americké (TEX je americký produkt), můžeme použít balíček `a4wide`, který nastaví evropské rozměry podle formátu A4, ale u třídy dokumentu musíme mít uveden nepovinný parametr `a4paper`, například takto:

```
\documentclass[a4paper,12pt]{article}
\usepackage{czech,a4wide}
```

Jestliže potřebujeme upravit rozměr (délku) pouze jedné stránky, vystačíme si s příkazem \enlargethispage{vzdálenost}, kde `vzdálenost` je délka, o kterou

chceme stránku prodloužit (nebo zkrátit, když použijeme záporné číslo). Například pokud se nám nevejde na stránku něco, co nechceme mít posunuto až na další stránku, použijeme tento příkaz s parametrem `1em` (samozřejmě na té stránce, kterou chceme prodloužit, raději výše). Když to nestačí, zvážíme, zda opravdu nechceme daný objekt nebo odstavec umístit až na další stránce, měli bychom brát v úvahu i estetické hledisko.

Veškeré rozměry týkající se stránky jsou uloženy v délkových registrech. Některé z nich můžeme měnit, jiné ne, obvykle z důvodu vzájemné závislosti. Více se podíváme na tuto problematiku až v příloze , teď si ukážeme na příkladu způsob nastavení některých rozměrů stránky.

U některých údajů najdeme záporná čísla. Je to proto, že u horního a levého okraje se ještě navíc počítá „prázdná“ vzdálenost přibližně 1 palec (`1in`), kterou můžeme případně záporným číslem zmenšit. Například horní (prázdný) okraj nad textem je 1 palec plus (nebo mínus) obsah určitého délkového registru.

Příklad 2.20 Použijeme délkové registry pro přímé nastavení rozměrů stránky. Všechny níže uvedené příkazy musí být v preambuli dokumentu, tedy ještě před příkazem `\begin{document}`.

Výpis č. 2.44

```
\setlength{\topmargin}{-0.2in}  % horní okraj
\setlength{\topskip}{0.3in}  % mezi záhlavím a textem
\setlength{\textheight}{9.6in}  % výška textu
\setlength{\leftmargin}{1cm}  % (+1in) levý okraj
\setlength{\textwidth}{6in}  % šířka textu
```

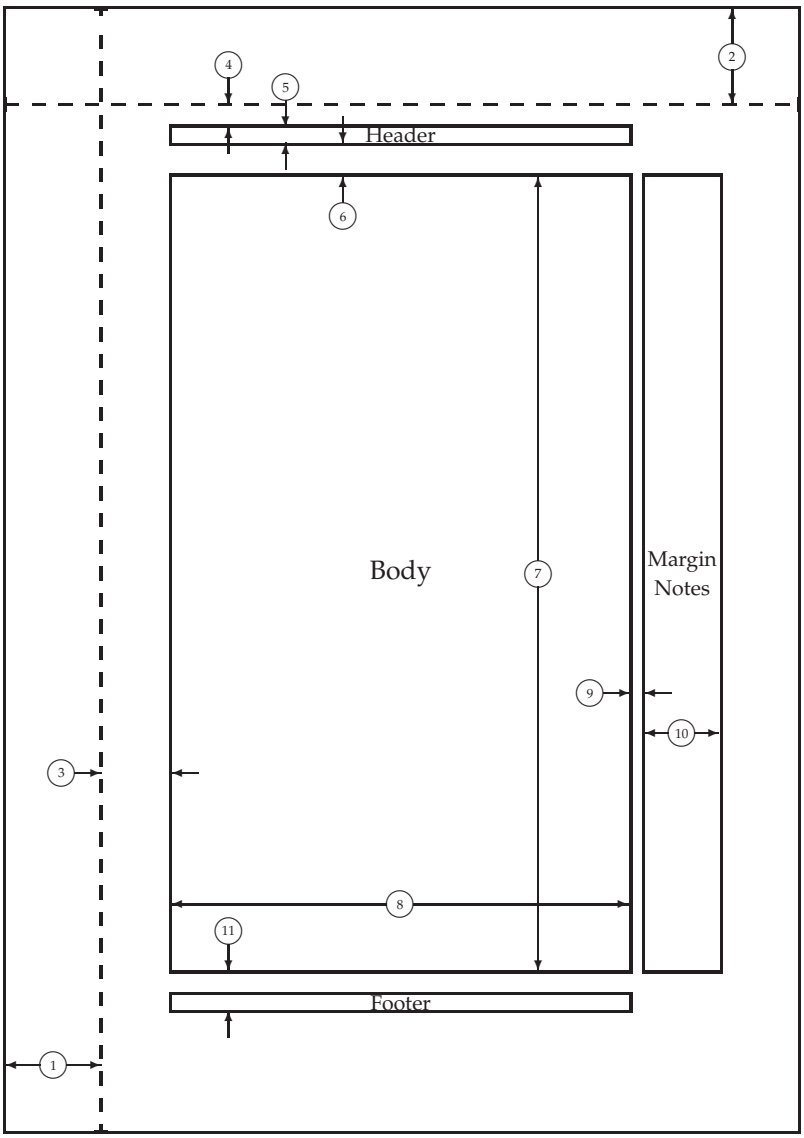
Z těchto registrů už známe příkaz `\textwidth`, který jsme používali pro zjištění šířky potištěné části stránky, ostatní už nejsou z tohoto hlediska tak důležité, ale také použitelné. Například když chceme vynechat půl strany místa (vertikálně), použijeme tento příkaz:

```
nějaký text
\vspace{.5\textheight}

další text po vynechané polovině strany
```

Délkové registry nemusíme nastavovat přímou hodnotou, můžeme jen upravovat o určitou relativní hodnotu. Například když chceme rozšířit textovou část o jeden centimetr, napíšeme příkaz

```
\addtolength{\textwidth}{1cm}
```

Různé možnosti nastavení rozměrů stránky máme v balíčku `geometry`. Užitečný může být také balíček `layout`, ve kterém je definován příkaz `\layout`, ukázku jeho výstupu vidíme na obrázku 2.9 na straně 92.




```
 1   one inch + \hoffset        2   one inch + \voffset
 3   \oddsidemargin = 53pt      4   \topmargin = 17pt
 5   \headheight = 12pt         6   \headsep = 25pt
 7   \textheight = 598pt        8   \textwidth = 345pt
 9   \marginparsep = 11pt      10   \marginparwidth = 57pt
11   \footskip = 30pt               \marginparpush = 5pt (not shown)
     \hoffset = 0pt                 \voffset = 0pt
     \paperwidth = 597pt            \paperheight = 845pt
```

Obrázek 2.9: Vykreslení rozměrů stránky příkazem `\layout`

2.14 Píšeme pro vědu

Tato kapitola není určena jen matematikům, ale všem, kdo píšou dokument s větami, důkazy, příklady či jinými podobnými prostředími (to se týká téměř všech vědních oborů), vytvářejí struktury či diagramy (také třeba lingvisté) nebo píšou postupy (algoritmy).

2.14.1 Definice, věty a důkazy

Exaktní věda je postavena na logice a logickém deduktivním vyvozování. To znamená, že v každém vědním oboru máme základní pojmy a základní (odpozorované) vztahy, a pak všechno ostatní – další pojmy, které definujeme pomocí základních nebo již definovaných pojmů, a odvozené vztahy (věty, teorémy, lemmata), které jsme odvodili pomocí základních nebo již odvozených vztahů.

Pokud se pohybujeme i v teoretické části některého vědního oboru, potřebujeme občas psát vlastní definice pojmů (i dříve definovaných, pro připomenutí) a také vlastní věty, teorémy („důležitější“, nové věty), lemmata (pomocné věty) a zapisovat jejich důkazy. Dalším prvkem z této oblasti jsou příklady, ukázky, úlohy a poznámky vysvětlující či doplňující věty a definice.

Pro zápis těchto prvků existují určité konvence. *Definice*, *různé druhy vět* a *poznámky* se sázejí kurzívou, vyznačuje se v nich zrušením kurzívy (tedy v LaTeXu prostě použijeme příkaz `\emph`), jsou číslovány. V článku číslování nezávisí na čísle kapitoly, ale v rozsáhlejším textu (třeba diplomové práci) ano.

Důkazy se nečíslují, příslušejí vždy k určité větě a hned za ní bývají uváděny (pokud z nějakého důvodu musí být dál, třeba když za větu chceme vložit lemma s důkazem ulehčující důkaz té věty, bývá za označením důkazu v hranatých závorkách odkaz na větu, kterou dokazujeme). Důkazy již nesázíme kurzívou, ale normálním písmem.

Příklady, *ukázky* či *úkoly* bývají číslovány, pro číslování platí totéž co pro věty, a stejně jako u důkazů používáme normální řez písma.

Pro číslování se používají čítače, jakési proměnné, které se při každém použití zvýší o 1. Různé prvky mohou mít různé čítače, ale často se setkáváme s tím, že třeba všechny druhy vět používají tentýž čítač.

LaTeX nám dává možnost vytvářet si svá vlastní prostředí s takovým označením a číslováním, jaké budeme chtít. Tuto problematiku jsme už trochu nakousli v kapitolách 2.6.2 o stylech a 2.8 o křížových odkazech.

Nové prostředí tohoto typu vytvoříme pomocí příkazu `\newtheorem`, kde zadáváme titulek prostředí a určujeme způsob číslování, které může být závislé na číslu kapitoly, případně lze číslování propojit s jiným takovým prostředím. Ukázku máme ve výpisu 2.45.

Výpis č. 2.45

```
\newtheorem{veta}{Věta}[section]  % prostředí závislé na číslování kapitol section
\newtheorem{lemma}[veta]{Lemma}  % prostředí číslované zároveň s prostředím veta
\newtheorem{pozorovani}{Pozorování}[section]
\newtheorem{definice}{Definice}[section]  % prostředí pro definice
\newtheorem{thpriklad}{Příklad}  % „meziprodukt“, použito dále:
\newenvironment{priklad}%
  {\begin{thpriklad}\normalfont}%
  {\end{thpriklad}}
\newenvironment{dukaz}%
  {\textbf{Důkaz:\quad}}%
  {$\Box$}  % pro tento příkaz je nutno načíst balíček latexsym nebo amssymb
```

Ukážeme si použití prostředí definovaných ve výpisu 2.45. Všimněte si ukončení prostředí pro důkazy – prázdný čtvereček se pro tyto účely používá dost často, má

zde význam „hotovo“.

Výpis č. 2.46

```
\begin{definice} Říkáme, že sněží, pokud
padají sněhové vločky.
\end{definice}

\begin{definice} Sníh je hmota složená
ze sněhových vloček.
\end{definice}

\begin{pozorovani}\label{poz:snihmraky}
Sněhové vločky padají z~tmavě
šedých mraků.
\end{pozorovani}

\begin{lemma} Když padá sníh, jsou na
obloze šedé mraky.
\end{lemma}

\begin{dukaz} Plyne z~Pozorování
\ref{poz:snihmraky}).
\end{dukaz}

\begin{veta} Když venku sněží, je
teplota menší než 5~\celsius.
\end{veta}

\begin{dukaz} Sněhové vločky se pohybují
vzduchem. Pokud je teplota blízko země
5~\celsius nebo vyšší, vločky roztávají
ještě ve vzduchu a~padá déšť.
\end{dukaz}

\begin{priklad} Podíváme se na oblohu.
Co tam vidíme? Pokud šedé mraky, můžeme
očekávat sněžení.
\end{priklad}
```

Definice 2.1 *Říkáme, že sněží, pokud padají sněhové vločky.*

Definice 2.2 *Sníh je hmota složená ze sněhových vloček.*

Pozorování 2.1 *Sněhové vločky padají z tmavě šedých mraků.*

Lemma 2.1 *Když padá sníh, jsou na obloze šedé mraky.*

Důkaz: Plyne z Pozorování 2.1. □

Věta 2.2 *Když venku sněží, je teplota menší než 5 °C.*

Důkaz: Sněhové vločky se pohybují vzduchem. Pokud je teplota blízko země 5 °C nebo vyšší, vločky roztávají ještě ve vzduchu a padá déšť. □

Příklad 1 Podíváme se na oblohu. Co tam vidíme? Pokud šedé mraky, můžeme očekávat sněžení.

Existují balíčky, které rozšiřují naše možnosti, například `ntheorem`. Dále v téměř každém balíčku se třídami nebo styly pro odborné články nebo rozsáhlejší práce bývají tato prostředí předdefinována.

2.14.2 Vzorce nejen matematické

Matematici, fyzikové, chemici, ekonomové a další obyvatelé této planety občas potřebují zapsat vzorec či více zarovnaných vzorců, a to buď samostatně na řádku (případně s číslováním), nebo v odstavci s textem.

Jednoduché vložení takového výrazu do textu jsme už vlastně používali – například výraz $x = y + 2/5$ vysázíme sekvencí `$x=y+2/5$`. Když tuto sekvenci ohraničíme místo symbolů `$` jejich dvojicemi (tedy `$$x=y+2/5$$`), vysází se výraz na samostatný

řádek.

Výrazy na samostatném řádku by správně měly být ve většině případů číslovány, aby bylo možné na ně odkazovat. Pak *místo* symbolů `$` použijeme matematická prostředí `equation` (jeden řádek) nebo `eqnarray` (i více řádků s možností určit zarovnání pomocí symbolu `&`, viz příklad 2.21, ale nemusíme použít). Na řádky můžeme přidat příkaz `\label` a pak jeho argument používat pro získání čísla rovnice. Když určitý řádek v soustavě nechceme mít číslovaný, použijeme na tomto řádku příkaz `\nonumber`.

Pro samotný obsah těchto matematických úseků nebo prostředí využíváme nejen matematické operátory, ale také funkce, speciální symboly a případně i řeckou abecedu. To nejzákladnější najdeme obvykle v menu nebo panelech nástrojů editoru, u editoru TEXnicCenter buď na jednom panelu nástrojů nebo v menu Insert → Formulas (ohraničení matematických prostředí) a Math (všechno ostatní).

Příklad 2.21 Ukážeme si zápis číslované rovnice a soustavy rovnic, a také odkaz na rovnici. Soustava rovnic bude zarovnána podle operátorů rovnosti a nerovnosti. Všimněte si způsobu zarovnání pomocí `&` v prostředí `eqnarray`.

Výpis č. 2.47

```
Rovnice:
\begin{equation}
x=y+2/5\label{eq:rovnice}
\end{equation}

To byla rovnice (\ref{eq:rovnice}).

Soustava:
\begin{eqnarray}
y-5+\frac{-x}{3} & \leq & \sqrt{2}*x
  \label{eq:deleniodmocninou}\\
x/2+y & < & 0 \nonumber\\
\sin(x)+1 & \geq & \cos(y)\\
x^2+y^{2n}-x^{2^t}& = & vysl_3\\
vysl_2 & = & \sum_{i=1}^n x^i\nonumber
\end{eqnarray}

První rovnice soustavy je
(\ref{eq:deleniodmocninou}).
```

Rovnice:

$$x = y + 2/5 \tag{2.1}$$

To byla rovnice (2.1).

Soustava:

$$\begin{aligned} y - 5 + \frac{-x}{3} &\leq \sqrt{2} * x && (2.2)\\ x/2 + y &< 0 \\ \sin(x) + 1 &\geq \cos(y) && (2.3)\\ x^2 + y^{2n} - x^{2^t} &= vysl_3 && (2.4)\\ vysl_2 &= \sum_{i=1}^{n} x^i \end{aligned}$$

První rovnice soustavy je (2.2).

To, že máme k dispozici spoustu různých symbolů z různých vědních oborů, už víme (viz strana 50 v kapitole 2.4.4), ale podpora pro tyto účely je v LATEXu mnohem širší. Ve třídách a stylech pro specializované odborné časopisy a jiné publikace najdeme mnohá užitečná nastavení, prostředí a příkazy, a také existuje mnoho balíčků se styly pro různé účely. Například balíček `xymtex` je určen pro vytváření formulí s chemickými strukturami, `tensor` obsahuje podporu sázení tenzorů s horními i dolními indexy správně zarovnanými, `xlop` dokáže nejen formátovat, ale také provádět matematické výpočty, atd.

2.14.3 Stromové struktury, diagramy a grafy

Pro strukturovaná data samozřejmě můžeme použít prostředí picture nebo některý balíček, který rozšiřuje jeho možnosti, ale máme také k dispozici balíčky určené přímo pro tyto účely.

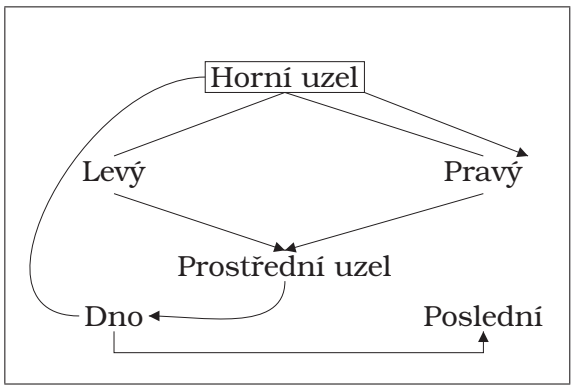
Stromové (nebo jakkoliv rozvětvené) struktury můžeme vytvořit pomocí různých balíčků, například `tree-dvips`, `xyling`, `trees`, `qtree`, `ps-trees`, atd. Většina z nich používá pro vykreslování čar a křivek Postscript, proto je třeba formát `PDF` tvořit přes formát `PS` (v TEXnicCenter profil překladu `LaTeX=>PS=>PDF`).

Příklad 2.22 Ve výpisu a na výsledku máme ukázku použití balíčku `tree-dvips`. Vykreslíme celkem šest uzlů, které jsou nazvány `x`, `y`, `z`, `u`, `v`, `p` (na tyto názvy se odkazujeme v dalších příkazech) a každý má svůj text (například uzel `x` má text `Horní uzel`). Uzly umístíme pomocí tabulky.

Pak propojíme uzly čárami, šipkami, křivkami (zde musíme zadat, jak moc bude „vykřivená“) a křivkami se šipkou. U všech těchto prvků můžeme v nepovinných parametrech určit body, ze kterých a do kterých čára nebo křivka povede. Například písmenko `t` určuje horní část uzlu, `bl` spodní levou (bottom-left) část.

Výpis č. 2.48

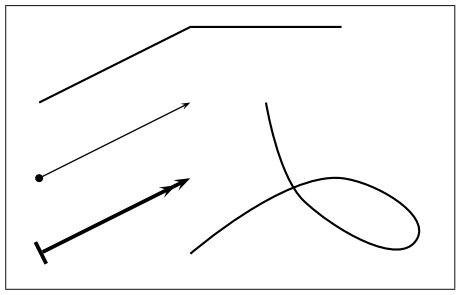
```
% Nejdřív vytvoříme a umístíme uzly:
\begin{tabular}{ccc}
    & \node{x}{Horní uzel}\\[2em]
\node{y}{Levý}&&\node{z}{Pravý}\\[2em]
    & \node{u}{Prostřední uzel}\\[1ex]
\node{v}{Dno}&&\node{p}{Poslední}
\end{tabular}
% Orámujeme první uzel:
\nodebox{x}
% Nakreslíme čáry:
\nodeconnect{x}{y}
\nodeconnect{x}{z}
% Nakreslíme šipky:
\anodeconnect{y}{u}
\anodeconnect{z}{u}
\anodeconnect[br]{x}[tr]{z}
% Nakreslíme křivku a křivku s šipkou:
\nodecurve[l]{x}[l]{v}{1.6cm}
\anodecurve[b]{u}[r]{v}{.7cm}
% „Rovná“ šipka dole:
\abarconnect[-8pt]{v}{p}
```



Dále si ukážeme použití některých příkazů balíčku `pstricks`. Jak název napovídá, opět budeme překládat přes formát `PS`. Samotný balíček nabízí spoustu příkazů souvisejících s grafikou, my si zde ukážeme jen to nejzákladnější – čáry s různými zakončeními a křivku. Tento balíček nelze používat zároveň s balíčkem `color`, což ale nevadí, protože má vlastní (a dokonalejší) správu barev. Máme k dispozici prostředi `pspicture` podobné již známému prostředí `picture`, předdefinované jednotky jsou však centimetry, ne body.

Výpis č. 2.49

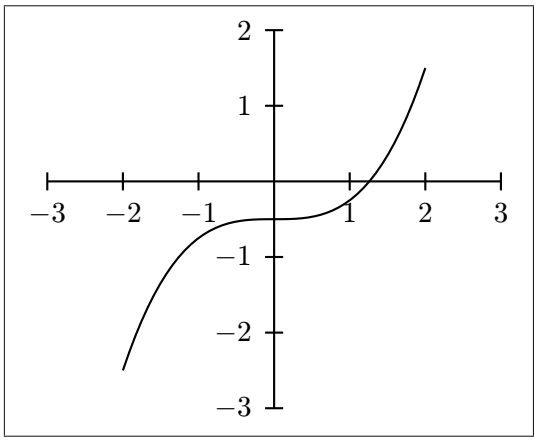
```
\begin{pspicture}(6,3)
% Lomená čára:
\psline(0,2)(2,3)(4,3)
% Šipky:
\psline[arrows=*->,linewidth=.5pt](0,1)(2,2)
\psline[arrows=|->>,linewidth=.5mm](0,0)(2,1)
% Křivka:
\pscurve(2,0)(4,1)(5,.2)(3.5,.7)(3,2)
\end{pspicture}
```



Další možnosti nám poskytne kombinace balíčku `pstricks` a některého dalšího, například `ps-plot` (tj. příkazem `\usepackage` načteme nejdřív `pstricks` a pak ještě `ps-plot`), v našem případě máme k dispozici kromě jiného příkazy k vykreslování funkcí:

Výpis č. 2.50

```
\begin{pspicture}(0,0)(4,5)
% Soustava souřadnic:
\psaxes(0,0)(-3,-3)(3,2)
% Funkce:
\psplot{-2}{2}{x 3 exp 4 div 0.5 sub}
\end{pspicture}(6,3)
```



První příkaz vykreslí soustavu souřadnic, druhý příkaz funkci $x^3/4 - 0.5$ na intervalu $x \in < -2, 2 >$ (to jsou první dva parametry)[12].

Pro vkládání grafů a jiných složitých nákresů opět existuje více možností. Některé balíčky dokážou vykreslit jednoduché funkce přímo (to jsme viděli u `pstricks + ps-plot`), pro složitější nebo vícerozměrné funkce je lepší využít specializované nástroje. Máme dvě základní možnosti:

a) použijeme kterýkoliv program pracující s grafikou nebo přímo s grafy a uložíme či exportujeme obrázek do vhodného formátu (většina těchto programů zvládá

[12] Zápis funkce je v postfixu, kdy se nejdřív píšou oba operátory a pak název funkce nebo operátor, například $4 + 5$ se zapíše jako $4\ 5\ +$ nebo výraz $2 * x + 4$ jako $2\ x\ *\ 4\ +$.

Postfixový zápis vytvoříme jednoduše: nejdřív si „uzávorkujeme“ celý výraz, například $a + 2 * x + 4 - z$ uzávorkujeme na $(((a + (2 * x)) + 4) - z)$, postupujeme podle toho, které operátory mají přednost, v případě několika se stejnou předností (třeba +,-) postupujeme zleva. Pak jdeme od nejvnitřnějších závorek – nejdřív napíšeme $2\ x\ *$, pak řešíme + s výsledkem $a\ 2\ x\ *\ +$ (operátor přidáváme na konec, operandy podle pořadí v původním výrazu), atd., v našem případě bude výsledek $a\ 2\ x\ *\ +\ 4\ +\ z\ -$.

export do EPS, PDF nebo něčeho převeditelného na tyto formáty) – *Mathematica*, *Inkscape* ([P20]), *Corel*, apod., výborným a přitom volně šiřitelným programem speciálně pro tyto účely navrženým je *GNUPlot* ([P21], pěkný úvod na [P22]),

b) existují programy vytvářející přímo TEXový kód – *LATEXPiX* ([P23]), *TEXcad32* ([P24]), *TEXCAD* ([P25]), atd.

2.14.4 Postupy, algoritmy, předem zformátovaný text

V odbornějších publikacích včetně diplomových prací je někdy třeba vložit *strukturovaný postup* nějaké činnosti. Nejde jen o algoritmus při programování, ale obecně o jakýkoliv postup nebo způsob konstrukce, který chceme napsat co nejpřehledněji. LATEX nám opět nabízí několik balíčků, z nichž jsou zajímavé například algorithm2e, algorithmicx, pseudocode, newalg. Některé z nich, například algorithm2e, nabízejí možnost vygenerovat seznam algoritmů podobně jako generujeme třeba obsah nebo seznam obrázků (příkaz je `\listofalgorithms`, v případě uvedeného balíčku má i podporu češtiny).

Příklad 2.23 Ukážeme si základní možnosti balíčku `alorighm2e`. Balíček načteme následujícím příkazem (v preambuli) s nepovinnými parametry pro vzhled a nastavení češtiny v popiscích a seznamu algoritmů:

```
\usepackage[ruled,vlined,czech]{algorithm2e}
```

Dále vytvoříme dva algoritmy. První popisuje postup při přecházení křižovatky (bez světelné signalizace), druhý postup čtení detektivky. U druhého postupu budeme chtít očíslovat řádky.

Výpis č. 2.51

```
\begin{algorithm}[H]
příchod ke křižovatce\;
\Repeat{nejede auto}{
  pohlédni vlevo\;
  \While{jede auto}{čekej\;
  pohlédni vpravo\;
  \While{jede auto}{čekej\;
  pohlédni vlevo\;
}
přejdi\;
\caption{Přechod křižovatky}
\label{alg:krizovatka}
\end{algorithm}
```

Algoritmus 1: Přechod křižovatky

příchod ke křižovatce;
repeat
  pohlédni vlevo;
  while *jede auto* **do**
    čekej;
  pohlédni vpravo;
  while *jede auto* **do**
    čekej;
  pohlédni vlevo;
until *nejede auto*;
přejdi;

Výpis č. 2.52

```
\linesnumbered
\begin{algorithm}[H]
vezmi knihu\;
otevři knihu na první straně\;
\While{není konec knihy}{
  čti kapitolu\;
  \eIf{rozumíš kapitole}{
    přesuň se na další kapitolu\;
    \If{víš kdo je vrah}{
      přestaň číst\;}
      přesuň se na konec knihy\;
    }
  }{
    zpět na začátek kapitoly\;
  }
}
\caption{Hledání vraha}
\label{alg:hledanivraha}
\end{algorithm}
```

Algoritmus 2: Hledání vraha

```
1  vezmi knihu;
2  otevři knihu na první straně;
3  while není konec knihy do
4      čti kapitolu;
5      if rozumíš kapitole then
6          přesuň se na další kapitolu;
7          if víš kdo je vrah then
8              přestaň číst;
9              přesuň se na konec knihy;
10     else
11         zpět na začátek kapitoly;
```

Jak vidíme, vlastně jde o nový typ plovoucích prostředí. Pokud se prostředí nevejde celé na stránku, kde je zadáno, vysází se až na další stránku, „poplave“ v textu o něco dál. Pokud tomuto chování chceme zabránit (nebo při umístění do prostředí `minipage`), použijeme nepovinný parametr `H`.

K postupům charakteru „programovacího“ nebo výpisům textových souborů patří možnost vložit text „tak jak je“, tedy včetně případných speciálních znaků, na které by LaTeX nějakým způsobem reagoval, mezer a konců řádků. K tomu slouží buď příkaz `\verb` nebo prostředí `verbatim` či `alltt`.

Příkaz `\verb` má trochu zvláštní použití. To, co chceme pomocí tohoto příkazu vložit, ohraničíme jakýmkoliv symbolem, který se v ohraničovaném textu nenachází (stejný symbol před i za textem, tedy ne levá a pravá závorka!), například:

Výpis č. 2.53

```
\verb+http://www.adresa.cz+\\
příkaz \verb␣\verb␣\\
soubor \verb|\C:\dokumenty\zivotopis.txt|
```

```
http://www.adresa.cz
příkaz \verb
soubor
C:\dokumenty\zivotopis.txt
```

Prostředí `verbatim` pracuje jako jiná prostředí, tedy text, který chceme nechat zachovaný tak, jak je (včetně konců řádků apod.), umístíme mezi `\begin{verbatim}` a `\end{verbatim}`.

Prostředí `alltt` z balíčku `alltt` (tento balíček musíme samozřejmě načíst) je svou funkcí podobné `verbatim`, ale navíc interpretuje některé příkazy LaTeXu. Toto prostředí používáme, pokud chceme zachovávat mezery a konce řádků, ale navíc používáme barvy, řezy a další příkazy.

Úlohy:

1. Vytvořte nový dokument třídy `article` se základní velikostí písma 10 bodů.
2. Do preambule dokumentu napište tento příkaz (vytvoření příkazu pro kapitolu): `\newcommand{\kapitola}[1]{\section{#1}}`
3. Nadefinujte si podobný příkaz pro podkapitolu:
   `\podkapitola{Název podkapitoly}`
   odkazující pro změnu na příkaz `\subsection` s předáním parametru.
4. Oba tyto příkazy pro kapitolu a podkapitolu použijte v dokumentu (překládejte do DVI), přidejte také běžný text, v něm některá slova vyznačte *vhodným* řezem.
5. Změňte třídu dokumentu na `report` a dále v definicích příkazů `\kapitola` a `\podkapitola` změňte odkazy na příkazy `\chapter` a `\section` (opět s předáním parametrů), aby nadpisy odpovídaly použití v této třídě.
6. Vytvořte a umístěte titulní stranu. Zvolte nadpis a sebe napište jako autora.
7. Na konci dokumentu vytvořte seznam literatury s alespoň dvěma položkami (vhodně formátovanými). Do první kapitoly umístěte citaci některé z položek seznamu literatury.
8. V některé z kapitol vytvořte jednoduchý obrázek (nákres – pomocí prostředí `picture`), přidejte titulek (tj. vložte tento obrázek do příslušného plovoucího prostředí).
9. V některé z kapitol vytvořte jednoduchou tabulku s obsahem podle vlastního výběru (alespoň dva sloupce a alespoň dva řádky), orámování si zvolte sami. Tabulku umístěte do příslušného plovoucího prostředí.
10. Za titulní stranu (tj. před kapitoly) umístěte *obsah*, *seznam tabulek* a *seznam obrázků*.
11. Z tohoto dokumentu vytvořte `PDF` soubor.

LITERATURA

— Citace a typografie —

[C1] BOLDIŠ, P.: *Bibliografické citace dokumentů podle ČSN ISO 690 a ČSN ISO 690-2: Část 1, Citace: metodika a obecná pravidla.* Verze 3.3 [online].
URL: http://www.boldis.cz/citace/citace1.ps,
http://www.boldis.cz/citace/citace1.pdf
[cit. 20. 10. 2006]

[C2] BOLDIŠ, P.: *Bibliografické citace dokumentů podle ČSN ISO 690 a ČSN ISO 690-2: Část 2, Modely a příklady citací u jednotlivých typů dokumentů.* Verze 3.0 [online].
URL: http://www.boldis.cz/citace/citace2.ps,
http://www.boldis.cz/citace/citace2.pdf
[cit. 20. 10. 2006]

[C3] JANÁK, R.: *Typografie.* [online], 2001.
URL: http://www.typografie.unas.cz/
[cit. 4. 12. 2006]

[C4] SÁRKÖZI, R.: *Jak se dělá školní časopis (3) – Počítačová typografie.* Česká škola [online], 2004.
URL: http://www.ceskaskola.cz/Ceskaskola/AR.asp?ARI=3212&CAI=2125
[cit. 5. 2. 2007]

[C5] OLŠÁK, P.: *Kam se poděla dobrá typografie?* In: Mensa 5/96, s. 37–42. [online], 1996.
URL: http://petr.olsak.net/ftp/olsak/typo/mensa.pdf
[cit. 5. 2. 2007]

— Diplomové práce —

[D6] NOVÁK, J.: *Doporučení pro úpravu závěrečných prací.* Diplomová práce. Brno, Mendelova zemědělská a lesnická univerzita [online], 2006.
URL: http://old.mendelu.cz/~rybicka/zpract/jaknadip.pdf
[cit. 25. 10. 2006]

[D7] POLÁCH, E.: *Pravidla sazby diplomových prací.* České Budějovice, Pedagogická fakulta Jihočeské univerzity [online], 1998.
http://home.pf.jcu.cz/~edpo/pravidla/pravidla.pdf
[cit. 25. 10. 2006]

[D8] *Studentův wikiprůvodce životem na ČVUT – Diplomová práce.* Praha [online], 2006.
URL: http://www.student.cvut.cz/cwut/, do pole Hledat v levé části okna zadejte text Diplomová práce
[cit. 25. 10. 2006]

[D9] SYSEL, P.: *Šablona pro diplomové práce.* Fakulta elektrotechniky a komunikačních technologií, VUT Brno [online].
URL:
http://www.utko.feec.vutbr.cz/latex/index.php?page=download#template
[cit. 8. 11. 2006]

[D10] Vysoká škola báňská-Technická univerzita, Fakulta elektrotechniky a informatiky. *Závazné pokyny pro vypracování bakalářské a diplomové práce.* Ostrava [online], 2005.
URL: http://fei.vsb.cz/www/dokumenty/szz_dp.html
[cit. 25. 10. 2006]

— Programy, instalace a návody —

[P11] České stránky firmy Adobe [online].
URL: http://www.adobe.cz
[cit. 2. 11. 2006]

[P12] *Nakladatelství Computer Press* [online].
URL: http://knihy.cress.cz
[cit. 28. 10. 2006]

[P13] Slunečnice [online].
URL: http://www.slunecnice.cz
[cit. 2. 11. 2006]

[P14] VAVREČKOVÁ, Š.: *Instalace a počeštění systému MikTEX verze 2.5 ve Windows.* Opava [online], 2006.
URL: http://fpf.slu.cz/~vav10ui/obsahy/dipl/instalace25.pdf
[cit. 8. 11. 2006]

[P15] *LyX for Windows.* [online].
URL: http://wiki.lyx.org/Windows/Windows
[cit. 29. 11. 2006]

[P16] MATOUCH, P.: *Instalace TEXu pro Windows.* [online].
URL: http://dce.felk.cvut.cz/roubal/teaching/LaTeX/TeXinst.pdf
[cit. 1. 12. 2006]

[P17] *Specifika vývoje českých dokumentů v LaTeXu, konfigurace vývojového prostředí a jeho individuální přizpůsobení potřebám a zvyklostem* [online].
URL: http://www.fi.muni.cz/~sojka/PB029/2006cv2.html
[cit. 1. 12. 2006]

[P18] Nakladatelství Springer: *Lecture Notes in Computer Science.* [online].
URL:
http://www.springer.com/east/home/computer/lncs?SGWID=5-164-7-72376-0
[cit. 5. 12. 2006]

[P19] The CTAN team: *CTAN Search.* Searching TEX and LATEX packages. [online].
URL: http://tug.ctan.org/find.html
[cit. 11. 1. 2007]

[P20] *Inkscape Homepage.*[online]
URL: http://www.inkscape.org/
[cit. 5. 2. 2007]

[P21] *GNUPlot Homepage.* [online], 2006.
URL: http://www.gnuplot.info/
[cit. 5. 2. 2007]

[P22] PINKAS, P.: *GNUPlot: seznamte se.* Článek na Root.cz, [online], 2001.
URL: http://www.root.cz/clanky/gnuplot-seznamte-se/
[cit. 5. 2. 2007]

[P23] *LATEXPiX Homepage.* [online]
URL: http://members.home.nl/nickvanbeurden/latexpix.htm
[cit. 5. 2. 2007]

[P24] *TEXCad32 Homepage.* [online]
URL: http://www.das-gelbe-rechenbuch.de/Texcad32/Index_e.html
[cit. 5. 2. 2007]

[P25] *TEXCAD Homepage.* [online]
URL: http://homepage.sunrise.ch/mysunrise/gdm/texcad.htm
[cit. 5. 2. 2007]

— MS Word —

[W26] KUBÁLEK, T., TOPOLOVÁ, I.: *Manažerská informatika. Textový procesor Microsoft Word verze 2000 CZ.* Praha, VŠE [online], 2001.
URL: http://fph.vse.cz/fakulta/informatizace/skripta_MSWord2000.asp
[cit. 20. 10. 2006]

[W27] RYBIČKA, J.: *Zpracování textů počítačem.* [online]
URL: old.mendelu.cz/~rybicka/zpract/mikrotyp.doc
[cit. 5. 2. 2007]

[W28] OLŠÁK, P.: *TEX kontra Word.* [online]
URL: http://petr.olsak.net/ftp/olsak/typo/texword.pdf
[cit. 5. 2. 2007]

[W29] VOJÁČEK, K.: *Zpracování textů textovým procesorem MS Word.* KAVOJ Publishing, [online], 2006.
URL: http://www.mgplzen.cz/download/ivt_word.pdf
[cit. 5. 2. 2007]

— TEX a LATEX —

[T30] KROB, J.: *Stránky o TEXu*[online].
URL: http://www.phil.muni.cz/~jokr/tex.html
[cit. 6. 11. 2006]

[T31] RYBIČKA, J. *LATEX pro začátečníky.* Brno, Konvoj, 2003.

[T32] ŠVEJDAR, V.: *O bibliografickém systému BibTEX.* UK Praha [online], 2005.
URL: http://www1.cuni.cz/~svejdar/texdev/BibTeXInfo.pdf
[cit. 8. 11. 2006]

[T33] PAKIN, S.: *The Comprehensive LATEX Symbol List.* [online], 2005.
URL: odkaz
(http://www.ctan.org/tex-archive/info/symbols/comprehensive/ ↩
symbols-a4.pdf) [cit. 29. 11. 2006]

[T34] HARDERS, H.: *Symbols in Textcomp.* [online], 2005.
URL: http://www.tug.org/tex-archive/info/symbols/text/textcomptst.ps
[cit. 29. 11. 2006]

[T35] *Československé sdružení uživatelů TEXu.* [online].
URL: http://www.cstug.cz/
[cit. 29. 11. 2006]

[T36] BEDNÁŘ, R.: *LATEX manuál.* [online].
URL: http://www.cstug.cz/latex/lm/frames.html
[cit. 29. 11. 2006]

[T37] SADOVSKÝ, P. a kol.: *Jak nejrychleji napsat svůj první dokument v LATEXu.* [online].
URL:
http://fyzika.ufyz.feec.vutbr.cz/~petrsad/download/LaTeX/JakPsat.pdf
[cit. 29. 11. 2006]

[T38] OETIKER, T., PARTL, H., HYNA, I., SCHLEGL, E., KOČER, M., SÝKORA, P.: *Nepříliš stručný úvod do systému LATEX 2ε.* [online].
URL:
http://fyzika.ufyz.feec.vutbr.cz/~petrsad/LaTeX/lshort_czech.pdf
[cit. 29. 11. 2006]

[T39] SOPOUCH, P., FORMÁNEK, P.: *LATEX v kostce.* [online].
URL: http://www.it.cas.cz/manual/latex/
[cit. 29. 11. 2006]

[T40] KOVÁŘ, P.: *Úvod do TEXu (pro každého něco).* [online].
URL: http://homel.vsb.cz/~kov16/talks/tex/
[cit. 29. 11. 2006]

[T41] ROBERTS, A.: *Getting to Grips with LATEX.* [online].
URL: http://www.andy-roberts.net/misc/latex/index.html
[cit. 29. 11. 2006]

[T42] DOOB, M.: *A Gentle Introduction to TEX.* A Manual for Self-study. Department of Mathematics, The University of Manitoba, Winnipeg, Manitoba, Canada. [online].
URL: ftp://ftp.cstug.cz/pub/tex/CTAN/info/gentle/gentle.pdf
[cit. 10. 1. 2007]

[T43] OSWALD, U.: *LATEX Graphics.* Petersburg. [online].
URL:
http://www.ursoswald.ch/LaTeXGraphics/overview/overview.html
[cit. 10. 1. 2007]

[T44] Indian TEX Users Group: *Online Tutorials in LATEX.* [online].
URL: http://www.tug.org.in/tutorials.html
[cit. 10. 1. 2007]

— OpenOffice.org —

[O45] OpenOffice.org Homepage. [online]
URL: http://www.openoffice.org/
[cit. 7. 2. 2007]

[O46] Domovská stránka OpenOffice.org v češtině. [online]
URL: http://www.openoffice.cz/
[cit. 7. 2. 2007]

[O47] Open Document. Článek na wikipedia.org, [online], 2006.
URL: http://cs.wikipedia.org/wiki/OpenDocument
[cit. 7. 2. 2007]

REJSTŘÍK POJMŮ

Z

REJSTŘÍK PŘÍKAZŮ A PROSTŘEDÍ

REJSTŘÍK BALÍČKŮ, TŘÍD A VOLEB