

Typografická příručka s přihlédnutím k \TeX

Josef Tkadlec, 18. 9. 2002

Souhrn typografických pravidel a jejich implementace do standardního \LaTeX u či \plainTeX u. Pracovní text bez záruky správnosti a bezchybnosti — nejsem ani typograf ani \TeX pert, i když k obojímu mám blízko. Vycházel jsem z různých typografických příruček, knih D. E. Knutha a P. Olšáka a z textu D. Nečase. Uvítám připomínky směřující ke zlepšení. Text může být šířen a využíván bez jakéhokoliv omezení.

Konkrétní číselné hodnoty se týkají písma `csr10`, pro jiné velikosti či typy písma mohou být odlišné. V ukázkách, které mají tři sloupce (dva výstupy a zdrojový text), odpovídá zdrojovému textu prostřední sloupec, první sloupec ukazuje, jak by situace vypadala bez dodržení příslušného pravidla či při použití kritizovaného zápisu.

Doporučovaná pravidla se snažím dodržovat. Konkrétně: kolem pomlček je doplněna mezera $0,08\bar{3}$ em, ve významu interpunkce je kresba rozšířena na $0,75$ em; jsou zúženy boxy dvojitých i jednoduchých uvozovek (odstraněna mezera na vnější straně), je doplněn kerning mezi uzavíracími uvozovkami a tečkou/čárkou; používá se pevná mezera $0,25$ em; používá se zúžená mezera, zužuje se za tečkami nekončícími větou, za apostrofem, na vnější straně uvozovek a kolem pomlčky (tam se zužování sčítá); používá se výpustka s mezerami $0,11\bar{1}$ em; před vykřičník a otazník se v textu doplňuje mezera $0,05\bar{5}$ em.

Obsah

1	Rozvrh stránky	2
1.1	Textové zrcadlo	2
1.2	Řádkový rejstřík	2
2	Lámání řádků	2
3	Znaky	3
3.1	Pomlčka	3
3.2	Uvozovky	4
4	Textové mezery	6
4.1	Nedělitelná mezera	6
4.2	Základní mezera	6
4.3	Verzálková mezera	7
4.4	Mezera šířky tečky, pevná mezera	7
4.5	Rozšířená mezera	8
4.6	Zúžená mezera	8
4.7	Pevná zúžená mezera	10
4.8	Výpustková mezera	10
4.9	Mezera před interpunkcí	11
5	Matematické výrazy	11
5.1	Typ písma	11
5.2	Typy matematických objektů	12
5.3	Mezery	12
5.4	Speciální symboly	14
5.5	Závorky	15
5.6	Vektory	16
5.7	Opakování operací	16
6	Problémy s definicemi	16

1. Rozvrh stránky

1.1. Textové zrcadlo

Názory jsou různorodé.

- *Optický střed* (pro nadpisy apod.) je ve $2/3$ výšky.

1.2. Řádkový rejstřík

Je vhodné dodržovat *řádkový rejstřík*, tj. účarí řádků tvoří pravidelnou osnovu. To se moc netýká matematických textů se spoustou vzorců.

- Je třeba mít „nepružně“ definovaný `\baselineskip`.
- Aby nedocházelo k vkládání mezer mezi odstavci, je vhodné vynulovat `\parskip`. Chceme-li zabránit vkládání mezer mezi řádky (raději se smíříme s případným překrytím), vynulujeme `\lineskip`. Aby nedocházelo k chybovým hláškám při nezaplnění stránky, je vhodné použít `\raggedbottom` nebo nadefinovat výšku textového zrcadla (v \LaTeX u) tak, aby se tam vešel celočíselný počet řádků (např. 25). Můžeme tedy použít např. (něco se významově překrývá):

```
\parskip=0pt \lineskip=0pt \raggedbottom
\textheight=24\baselineskip \addtolength{\textheight}{\topskip}
```

- Úprava záhlaví v \LaTeX u (místo 2 může být jiné celé číslo):

```
\headsep=2\baselineskip \addtolength{\headsep}{-\topskip}
```

- Možné předefinování sekcí v \LaTeX u:

```
\makeatletter
\renewcommand{\section}{\@startsection
  {section}{1}{0pt}{2\baselineskip}{\baselineskip}{\lineskip=0pt\Large\bf}}
\renewcommand{\subsection}{\@startsection
  {subsection}{2}{0pt}{\baselineskip}{\baselineskip}{\lineskip=0pt\bf}}
\makeatother
```

- Nastavení vhodného mezerování v \LaTeX ovských seznamech 1. a 2. úrovně:

```
\makeatletter
\partopsep\baselineskip
\def\@listi
  {\leftmargin=\leftmargini \topsep=0pt \parsep=0pt \itemsep=\baselineskip }
\let\@listI\@listi
\def\@listii{\leftmargin=\leftmarginii \labelwidth=\leftmarginii
  \advance\labelwidth-\labelsep \topsep=0pt \parsep=0pt \itemsep=0pt }
\makeatother
```

2. Lámání řádků

- Po rozdělení slova by na konci i začátku řádku měly zůstat alespoň 3 znaky (včetně rozdělovníku, interpunkce, uvozovky). Při sazbě do úzkých sloupců lze toto pravidlo nedodržet a předefinovat:

```
\lefthyphenmin=1 \righthyphenmin=2
```

Je zapotřebí to provést až po zavedení příslušného jazyka, tj. v \LaTeX u buď za `\begin{document}` nebo v příkazu `\AtBeginDocument{...}`. Čísla udávají maximální počty písmen (bez rozdělovníku)

na konci/začátku řádku po rozdělení. Započítání např. uvozovek do počtu znaků je v \TeX u teoreticky možné, pak by ale bylo třeba do seznamu dělení slov přidat i varianty s jednou a s oběma uvozovkami, čímž by se několikanásobně zvětšil. (Byl by to zdrojový seznam pro program `patgen`, který by vygeneroval schemata dělení.)

- *Spojovník* se při dělení zopakuje i na začátku dalšího řádku. To lze zařídit (v \LaTeX u) použitím podmínky `split` (`\usepackage[split]{czech}`), což má ale za následek problémy se znakem `-` (je předefinován jako aktivní znak): např. ho nelze použít v příkazech `\input`, `\looseness`, `\cline` v prostředí `tabular` v \LaTeX u. (Obecně jsou problémy s tímto znakem v horizontálním modu, lze místo něj použít příkaz `\minus`.) Problémy lze lokálně odstranit příkazem `\standardhyphens`. Další řešení je použít ve zdrojovém textu místo spojovníku zvláštní příkaz, např.:

```
\def\={\discretionary{-}{-}{-}}
```

Přijdeme tím o akcent, je třeba dořešit problémy s dělením slov okolo spojovníku(???)

- V textu by měla být nejvýše 3 rozdělená slova pod sebou [2] (někdy se toleruje až 6), k dělení by nemělo dojít na konci stránky.
- Poslední (východový) řádek odstavce by měl končit mezerou alespoň 1 em, někde se preferuje velikost odstavcové zarážky (pokud je nenulová). V případě, že je nenulová odstavcová zarážka, připoustí se plný řádek. Východový řádek odstavce by měl mít navíc délku alespoň odstavcové zarážky. Nejjednodušší je zajištění mezery alespoň 1 em:

```
\parfillskip=1em plus 1fil
```

Nevýhodou je, že pro krátké odstavce (1–2 řádky), ve kterých by výchozí řádek vyšel skoro plný, může být problém text patřičným způsobem stáhnout. Takovou situaci je nespíše vhodné řešit předefinováním `\parfillskip` pro daný odstavec na nulovou hodnotu. V \LaTeX u se s tímto příkazem někdy pracuje, takže nezabírá (například v prostředích typu seznam).

Trochu složitější konstrukce zajistí i požadavek na nejmenší délku výchozího řádku:

```
\parfillskip=\hsize
\advance\parfillskip by -\parindent
\advance\parfillskip by Opt minus \parfillskip
\advance\parfillskip by Opt minus -1em
```

Problém je, že východový řádek je tlačěn k tomu, aby byl krátký (délka nad hodnotu `\parindent` je penalizována).

- Zlom stránky by měl na konci i začátku stránky nechat alespoň 2 řádky odstavce — netýká se situace, kdy odstavec má jen jeden řádek.

3. Znaky

3.1. Pomlčka

Pomlčka se používá v různých situacích:

- Jako interpunkce ve větě se píše s mezerami (zúženými, viz část 4.6) a nesmí být na začátku řádku.
- K uvození přímé řeči se naopak používá na začátku řádku (obvykle za odstavcovou zarážkou) a odděluje se pevnou mezerou. Lze jí přímou řeč i ukončit.
- Při označení celých měnových hodnot (Kč 5,-) se přikládá těsně za desetinnou čárku.
- Jako spojovník ve významu „a“, „až“, „od do“, „versus“ (pár Malá–Malý, Bolzanovy–Cauchyovy podmínky, strany 4–6, zápas Sparta–Slavia), se píše bez mezer a nesmí být ani na začátku ani na konci řádku.
- K vyznačení valenční čáry v chemických vzorcích.

V české sazbě se používaly pomlčky 0,5 em, 0,75 em a 1 em, v současné počítačové sazbě bývají k dispozici jen krajní dvě velikosti. Pomlčka 1 em se používá jako interpunkce v anglické sazbě (bez mezer), pro českou se považuje za příliš širokou (je navíc oddělena mezerou), i když se s ní lze setkat (spíše v akcidenčních tiskovinách). Pomlčka 0,5 em je pro použití jako interpunkce dost krátká.

Pro valenční čáru je asi vhodné použít spíše znak minus (pro vícenásobné vazby lze pak použít rovnítko či znak pro kongruenci), jehož kresba má šířku 0,6 em a okraje v boxu jsou 0,083 em. V ostatních případech lze použít stejně dlouhou pomlčku, nebo o něco delší pomlčku o šířce kresby 0,666 em nebo 0,75 em — ta je v případě „neinterpunkčního“ použití moc dlouhá, asi by pak bylo lepší použít pomlčky různých délek.

V \TeX u je za pomlčkou implicitně povolen zlom řádku, což je nevhodné, takže je lépe uzavřít ji do boxu. V cm -fontech jsou pomlčky přes celou šířku boxu, takže se někdy překrývají s kresbami sousedních znaků, což by bylo vhodné odstranit. Níže uvádím definice, které tyto problémy řeší. Příkaz $\backslash=$ navíc zužuje kolem pomlčky (zúžení odpovídá doplněnému okraji, který ale zůstává při případném zlomu řádku), případně vybírá vhodnou délku.

```
\def\okrajpoml {0.08333em }
\def\poml {\leavevmode\hbox{\kern\okrajpoml--\kern\okrajpoml}}
\def\Poml {\leavevmode\hbox{\kern\okrajpoml--\kern-0.4em--\kern\okrajpoml}}
\def\POml {\leavevmode\hbox{\kern\okrajpoml--\kern-0.33333em--\kern\okrajpoml}}
\def\POMl {\leavevmode\hbox{\kern\okrajpoml--\kern-0.25em--\kern\okrajpoml}}
\def\={\ifvmode \POMl \else \ifdim\lastskip=0pt \poml \else
\kern-\okrajpoml\POMl \fi\fi \afterassignment\ZrusZaPo\let\tmp= }
\def\ZrusZaPo {\expandafter\ifx\space\tmp \kern-\okrajpoml \else
\ifx\tmp \kern-\okrajpoml \fi\fi\tmp}
```

strany 4–6, Malá–Malý, Kč 5,–,	strany 4--6, Malá--Malý, Kč~5,--,
strany 4–6, Malá–Malý, Kč 5,–,	strany 4\poml6, Malá\poml Malý, Kč~5,\poml,
strany 4–6, Malá–Malý, Kč 5,–,	strany 4\Poml6, Malá\Poml Malý, Kč~5,\Poml,
strany 4–6, Malá–Malý, Kč 5,–,	strany 4\POml6, Malá\POml Malý, Kč~5,\POml,
strany 4–6, Malá–Malý, Kč 5,–,	strany 4\POMl6, Malá\POMl Malý, Kč~5,\POMl,
je to – asi – těžké	je to~-- asi~-- těžké
je to – asi – těžké	je to~\Poml{} asi~\Poml{} těžké
je to – asi – těžké	je to~\POml{} asi~\POml{} těžké
je to – asi – těžké	je to~\POMl{} asi~\POMl{} těžké
je to — asi — těžké	je to~--- asi~--- těžké
$H-C\equiv N, O=C=O$	$\$ \backslash \text{rm } H\{\backslash \text{equiv}\}C\{=\}N\$, \$O\{=\}C\{=\}O\$\$

3.2. Uvozovky

České uvozovky, ať už „dvojitě“ či „jednoduché“, se od amerických „dvojitých“ či „jednoduchých“ na první pohled liší. V `czech.sty` jsou definovány potřebné znaky: dvojitě `\clqq... \crqq` („...“) a jednoduché `\clq... \crq` („...“).

Potřeba nových otevíracích uvozovek je jasná, méně viditelná je nutnost nových dvojitých uzavíracích uvozovek — uvozovky v cm -fontech obsahují ve znaku na vnější straně navíc mezeru asi 0,1 em, takže při použití americké otevírací jako české uzavírací je od textu příliš oddělena. V cs -fontech je to řešeno tak, že je kresba této uvozovky posunuta. Problém je ale v tom, že české uvozovky takovouto mezeru navíc mít nemají (dokonce se kolem nich má mezeru zužovat, viz část o mezerách).

Mezi další závady v `czech.sty` patří mezeru vpravo u popisu obou jednoduchých uvozovek — u `\clq` malá (asi vylepšení), u `\crq` obrovská (asi aby mohla přímo následovat dvojitá bez explicitního doplnění mezery). Pokud to z nějakých důvodů nemůžete opravit v `czech.sty`, lze to nouzově opravit následujícím způsobem:

```
\def\allowhyphens {\nobreak\hskip0pt\relax}
\def\UVkor{\kern-0.07em }
\def\clq {\allowhyphens{,}}
```

```

\def\crq  {{{}}
\def\clqq {\leavevmode\hbox{}\UVkor\allowhyphens\char254 }
\def\crqq {\char255 \UVkor\hbox{}}

```

Příkaz `\clq` by bylo lepší vyrobit z apostrofu příkazem `\lower` (v \LaTeX u lze `\se@low@box`), čárka je v `\tt` jiná. Složené závorky zabrání nevhodným ligaturám (například „‘!‘“ dá „i“), příkaz `\allowhyphens` umožní dělení slova za levou uvozovkou; je před znakem uvozovky, aby zůstal zachován kerning, může tam být, protože tento znak má nulové `\lccode`.

V \LaTeX u je v některých starších instalacích pokus započítávat dvojité uvozovky do počtu znaků, které musí zůstat na jednotlivých stranách při dělení slova — to by ale vyžadovalo doplnění verzí slov s jednou či oběma uvozovkami do slovníků dělení, což uděláno není, takže může dojít k nesprávnému dělení slov.

V některých instalacích se problém s nadbytečnou mezerou řeší různými konstrukcemi — buď se kresba uvozovky „vycentruje“ (nepovažuji za vhodné) nebo se „posunutí“ kresby řeší pomocí `\kern`, což má za následek, že přebytečná mezera „zmizí“ alespoň na začátku a konci řádku (zůstane ale třeba na začátku odstavce).

V následujících ukázkách jsou kapitálkami označeny původní varianty příkazů.

začátek řádku	začátek řádku
„ahoj“, řekl ,ahoj‘, řekl	„,ahoj“, řekl ,ahoj‘, řekl
„ahoj“, řekl ,ahoj‘, řekl	<code>\CLQQ</code> ahoj <code>\CRQQ</code> , řekl <code>\CLQ</code> ahoj <code>\CRQ</code> , řekl
„ahoj“, řekl ,ahoj‘, řekl	<code>\clqq</code> ahoj <code>\crqq</code> , řekl <code>\clq</code> ahoj <code>\crq</code> , řekl
začátek řádku	začátek řádku

Existuje řada způsobů, jak uvozovky zapisovat ve zdrojovém textu.

- Dají se používat výše uvedené příkazy, to ale není moc čitelné. Je třeba si navíc uvědomit, že „polykají mezery“, tj. je třeba psát třeba `... \crqq{}` ...
- V `czech.sty` je zaveden příkaz `\uv{...}`, který dá text uvnitř do dvojitých uvozovek. To také není moc přehledné (zvláště u delších výrazech) a mohou nastat problémy pro složitější argument. Současná definice se snaží o to, aby uvnitř fungoval `verbatim`, zato nefunguje kerning mezi uvozovkami a textem. Tento příkaz lze jednoduše předefinovat (pak ale nefunguje `verbatim`):

```
\def\uv #1{\clqq #1\crqq}
```

- Lze používat znaky pro palec s následující definicí (výhoda i nevýhoda je v tom, že uvozovky musí být spárovány).

```

\catcode'\="=\active
\def"{\bgroup\def"{\crqq\egroup}\clqq}

```

- Lze používat standardní zápis ‘...’, ‘...’ s předefinováním významu uvozovek pomocí příkazu `\csprimeson` (původní význam se zavede příkazem `\csprimesoff`) z `czech.sty`. Zde se objevují problémy s kerningem a s apostrofem v matematickém módu.
- Lze používat ,,...‘, ,,...‘. V \CSTeX u je dvojice čárek brána jako ligatura vedoucí na otevírací dvojitou uvozovku, ale ‘‘ jsou otevírací americké, tj. mají vlevo mezeru 0,1 em navíc. Otevírací jednoduchá uvozovka nevypadá vždy jako čárka (např. při `\tt`). Další problém je použití ‘ za vykřičníkem a otazníkem — dvojice ?‘ a !‘ dávají znaky $\grave{?}$ a $\grave{!}$. To lze vyřešit vsunutím nějakého neviditelného materiálu, např. `?{}`.
- Lze používat např. “...” s definicí:

```

\catcode'\="=\active
\def"#1{\ifx'#1\clqq \else \ifx'#1\crqq \fi\fi}

```

- Nejlepší řešení pro dvojité uvozovky je nejspíš používat 8bitové kódování vstupu, který je obsahuje,

zápis	šířka v em	název/použití
Původní mezery		
<code>␣</code>	0,333̄ (0,222–0,500)	základní (další se ignoruje)
<code>\␣</code>	0,333̄ (0,222–0,500)	základní (skládají se)
<code>~</code>	0,333̄ (0,222–0,500)	základní nedělitelná
<code>\,</code>	0,166̄	pevná zúžená (v L ^A T _E Xu)
Možná do- či předefinování		
<code>-</code>	0,250 nebo 0,222	pevná
<code>\␣</code>	0,250 (0,166̄–0,416̄)	zúžená
<code>\~</code>	0,250 (0,166̄–0,416̄)	zúžená nedělitelná
<code>\.</code>	0,111̄	výpustková

Tabulka 1: Textové mezery (v závorce rozmezí pro pružné).

a zadefinovat si je jako aktivní znaky s patřičným významem.

Výhodou použití příkazů je i to, že můžeme automatizovat vhodné mezerování kolem uvozovek — odstranit přebytečnou mezeru a případně mezery kolem nich zužovat (viz sekce 4.6).

4. Textové mezery

V T_EXu se používá celá řada mezer. Základní typografický rozměr, který nás bude zajímat, je *šířka kuželky*, což je v T_EXu 1 em (zhruba šířka M). Takřka všechny mezery, se kterými se zde setkáme, vznikly jako podíl malých celých čísel (přesněji zaokrouhlený asi na pět desetinných míst), obvykle 1/18 em, v českých pravidlech se pracuje obvykle s násobky 1/12 em. Pro větší přehlednost (aby byly lépe vidět rozdíly) je nebudeme uvádět ve tvaru zlomku, ale ve tvaru desetinného čísla na 3 desetinná místa pro font cmr10. Mezery se používají pro oddělení slov či výrazů — těm budeme říkat *mezislovní* — a pro různé korekce. Přehled základních mezer je v tabulce 1 — včetně možných zavedení dalších mezer, které nejsou v T_EXu standardně zahrnuty, ale v českých textech se používají.

Základní pravidla pro psaní mezer (v češtině, čísla jsou orientační):

Mezislovní mezera se nedělá před interpunkcí (., ,, : , ; , ! , ?), naopak dělá za ní. Výjimkou je pomlčka, před kterou se mezera dělá (zúžená, viz 4.6), ale v této mezeře nemá dojít ke zlomu řádku. Někdy se před :, ; , ! , a ? doporučuje malá mezera, ale pak by měla být spíše součástí daného znaku.

4.1. Nedělitelná mezera

Používá se v případech, kdy chceme zabránit zlomu řádku v příslušném místě — např. pokud se nějaký (krátký) text váže k dalšímu, aby konce řádků byly „přirozenými“ mezerami ve čtení, aby tam opticky nezůstávaly „zbytky“. Příklady:

- za jednopísmennými předložkami (k, o, s, u, v, z), spojkami „i“ a „a“; za spojkou „a“ se povoluje výjimka, zejména pokud je to souřadící větná spojka a pokud nenásleduje za interpunkcí nebo závorkou; ve všech případech je povolena výjimka v úzkých sloupcích;
- před pomlčkou;
- mezi řády čísel (10 000, 3,141 592);
- mezi číslem a popisem, ke kterému patří (2 km, 16 žab, kapitola 5, r. 2001, 1. místo, 5. únor, 5. 2. — u data se připouští zlom za údajem měsíce);
- u ustálených zkratk (př. n. l, a. s.).

4.2. Základní mezera

Má mít základní délku 0,333̄ em [1] a má být v rozmezí mezi šířkami písmen „l“ a „n“, tj. obvykle 0,250 em až 0,500 em.

V T_EXu se zadává obvykle znakem mezery, nedělitelná varianta znakem pro vlnku (~). Standardně se použije mezera navržená ve fontu těmito parametry: základní velikost (`\fontdimen2`), největší „dobré“

rozpálení (`\fontdimen3`) a největší stažení (`\fontdimen4`). Pro `csr10` je to po řadě $0,33\bar{3}$ em, $0,16\bar{6}$ em a $0,11\bar{1}$ em, nejmenší velikost je tedy $0,22\bar{2}$ em a největší „dobrá“ velikost je $0,500$ em, což odpovídá šířkám kreseb písmen „l“ a „n“ (šířky boxů jsou $0,27\bar{7}$ em a $0,55\bar{5}$ em).

Mezera se v \TeX u může rozpálit na více než největší „dobrou“ velikost — ta odpovídá situaci, kdy `\hbadness` má hodnotu 100. Při větším rozpalování ale tato hodnota výrazně (se třetí mocninou) narůstá, navíc o ní máme informaci při překladu, takže to lze pohlídat.¹ Lze také nastavit základní velikost na $0,25$ em a největší stažení na $-0,25$ em, což by vedlo k těsnější sazbě tak, jak je tomu u některých jiných sázecích programů.

Chceme-li nastavit nejmenší velikost základní mezery na $0,25$ em, můžeme předdefinovat největší povolené stažení (v `csplainu` je `\tenrm` už zavedeno):

```
\font\tenrm=csr10 \fontdimen4\tenrm=0.08333em
```

Lze také předdefinovat `\spaceskip` na nenulovou hodnotu — v takovém případě má přednost před mezerou danou hodnotami `\fontdimen`.

4.3. Verzálková mezera

Při psaní verzálkami (velkými písmeny) je třeba používat větší mezery, protože místa mezi slovy jsou méně prosvětlená. Její základní velikost má být $0,5$ em a neměla by být širší než $0,66\bar{6}$ em. Podle [5] ji lze stáhnout na $0,33\bar{3}$ em. Při vyrovnávání (doplňování malých mezer mezi písmena) se úměrně zvětšuje.

Pokud není verzálková mezera pro dané písmo navržena, je asi nejvhodnější vyjít ze základní mezery a zvětšit ji o $0,16\bar{6}$ em — bere se tím do úvahy typ a velikost písma. Nejpřirozenější zavedení v \TeX u je asi předdefinování hodnoty použité mezislovní mezery:

```
\def\CapCor {0.16667em\relax}
\def\capspaces {\spaceskip=\fontdimen2\font plus \fontdimen3\font
  minus \fontdimen4\font \advance\spaceskip by \CapCor}
\def\nocapspaces {\spaceskip=0pt\relax}
```

Takto zavedený příkaz `\capspaces` je nutno použít při každé změně fontu. Změna hodnoty `\spaceskip` je na rozdíl od změny `\fontdimen` lokální, takže příkaz `\nocapspaces` většinou není zapotřebí.

Pro krátké texty psané explicitně verzálkami se někdy doplňuje mezera navíc příkazem `_`. Pokud text nezarovnáváme do bloku (například v nadpisech), použije se základní velikost základní mezery a dospějeme k přijatelné hodnotě $0,66\bar{6}$ em. Je-li příkaz `_` předdefinován na zúženou mezery (viz část 4.6), dostaneme $0,58\bar{3}$ em a můžeme navíc upravit mezery např. mezi písmeny „A“ a „V“. Lze také upravit definici `_` tak, aby se po mezeře přidalo právě $0,16\bar{6}$ em. (V následujících ukázkách je `_` předdefinováno na zúženou mezery).

MEZERA VE VELKÝCH PÍS.	MEZERA VE VELKÝCH PÍS.
MEZERA VE VELKÝCH PÍS.	<code>\capspaces</code> MEZERA VE VELKÝCH PÍS.
MEZERA VE VELKÝCH PÍS.	MEZERA \ \ VE \ VELKÝCH \ PÍS.
MEZERA VE VELKÝCH PÍS.	MEZERA \ VE \ VELKÝCH \ PÍS.
MEZERA VE VELKÝCH PÍS.	MEZERA {} VE {} VELKÝCH {} PÍS.

4.4. Mezera šířky tečky, pevná mezera

Mezera šířky tečky se používá při oddělování některých cifer v zápisu čísel (netýká se to čtyřciferných celých čísel v textu a letopočtů). Je nedělitelná (je to součást výrazu) a má šířku tečky (nejspíš boxu).

Pevná mezera se používá mezi číslem a označením jednotky (nebo symbolem, např. %, §) a za pomlčkou použitou na začátku řádku jako vyznačení přímé řeči. Je nedělitelná, doporučené velikosti odpovídají nejmenší velikosti základní mezery, tj. $0,25$ em.

¹V \LaTeX u se standardně vypisuje informace o „underfull“ boxu v případě, že je `\hbadness` větší než 1000, což pro `csr10` nastane při rozpálení na asi $0,7$ em.

Pevná mezera by nejspíš měla mít šířku právě nejmenší základní mezery — je nepružná a určitě by neměla být větší (chceme zdůraznit, že výrazy k sobě patří alespoň tak, jako slova), na druhou stranu má charakter základní mezery. Navíc by nejspíš neměla být menší než mezera šířky tečky — pokud napíšeme 10 000 km, pak k sobě cifry patří alespoň tak těsně, jako jednotka k číslu. Protože nejmenší základní mezera a šířka tečky jsou obvykle stejné (0,25 em), bývá velikost pevné mezery těmito podmínkami jednoznačně určena a bývá stejná jako šířka tečky.

V `csr10` je to složitější, protože tečka má šířku 0,277 em a nejmenší základní mezera 0,222 em. Je sice možné předefinovat stažitelnost základní mezery, ale nejspíš lze použít jak mezeru šířky 0,222 em, tak kompromisně mezeru 0,25 em — rozdíly jsou malé, navíc asi ne příliš často dojde ke stažení základní mezery pod 0,25 em.

Jako nejpřehlednější mi připadá použití „_“, případně s ponecháním původního významu `\sb` v matematickém módu. Méně problematické je použití příkazu `\:`. Při používání různých stupňů písma je asi lepší vyrobit pevnou mezeru z příslušných `\fontdimen`. To je ukázáno pro `\SPpevna`, kde se navíc ošetřují i verzálové mezery — předpokládá se, že pouze v takovém případě je použito nenulové `\spaceskip` podle části 4.3. Příkaz `\:` je zaveden jako mezera šířky tečky.

```
\def\:{\hphantom{.}}
\def\SPpevna {\kern\fontdimen2\font \kern-\fontdimen4\font
 \ifdim\spaceskip=0pt \else\kern\CapCor\fi}
\catcode'\_=\active
\def_{\ifmmode\sb\else\SPpevna\fi}
```

2 km, 100 000, 3,141 592	2~km, 100~000, 3,141~592
2 km, 100 000, 3,141 592	2_km, 100\:000, 3,141\:592

Někdo používá D. E. Knuthem doporučenou mezeru `\`, o šířce 0,166 em (v matematice, v \LaTeX u je i v textu), ta mi však připadá malá, což vynikne např. v následujících spojeních:

stojí to 100 000,00 Kč	stojí to 100\,000,00\,Kč
stojí to 100 000,00 Kč	stojí to 100_000,00_Kč
rychlostí 25 km/h urazil 1/2 km	rychlostí 25\,km/h urazil 1/2\,km
rychlostí 25 km/h urazil 1/2 km	rychlostí 25_km/h urazil 1/2_km
5% roztok obsahuje 5 % látky	5\% roztok obsahuje 5\,% látky
5% roztok obsahuje 5 % látky	5_% roztok obsahuje 5_% látky

4.5. Rozšířená mezera

V některých jazycích (i v některých českých dokumentech) se používá na konci věty. Opticky odděluje větné celky.

Toto je v \TeX u implicitní nastavení, které lze potlačit příkazem `\frenchspacing` (volá se automaticky při použití `czech.sty`) a obnovit příkazem `\nofrenchspacing`.

4.6. Zúžená mezera

Používá se v případě, kdy některý z objektů po jejich stranách je malý, např. za tečkou či čárkou, u uvozovky či apostrofu nebo kolem pomlčky. Smyslem je, aby mezera opticky vypadala stejně jako jiné (volný prostor kolem uvedených znaků ji opticky zvětšuje). Zúžení by mělo být o 0,083 em (příklad pro 12bodové písmo) či o polovinu šířky tečky (asi se myslí kresba) pro větší písmo.

V \TeX u je tečka obvykle o něco širší, má 0,277 em, naopak mezera je užší (nejušší má 0,222 em), takže je asi vhodnější zužovat méně (tečka je např. za písmenem „v“ příliš daleko): o 0,055 em (aby nejmenší mezera byla 0,166 em), případně „pružně“, jako je v tomto textu:

```
\def\zuzeni {\hskip-.08333em minus -.02778em\relax}
```

V \TeX u lze zužování mezery za tečkou, čárkou či apostrofem řešit přes aktivní znaky. V příkladu pro apostrof je ukázáno, jak zajistit standardní chování tohoto znaku v matematickém módu — je třeba mj. přepsat definici `\pr@m@s` v místě, kde je tento znak aktivní. Pokud to nepotřebujeme, můžeme si odpustit definici `\PRIME` a zjednodušit definici apostrofu na `\apost\TestZa`.

```
\def\apost {'} \catcode'\=' \active \let\PRIME=
\def'\{\ifmmode\expandafter\PRIME\else\apost\expandafter\TestZa\fi}
\def\TestZa {\afterassignment\ZuzZa\let\tmp= }
\def\ZuzZa {\expandafter\ifx\space\tmp \tmp\zuzeni \else\ifx~\tmp
\tmp\zuzeni \else\expandafter\expandafter\expandafter\tmp\fi\fi}
{\catcode'\@=11 \def\pr@m@s {\ifx'\next\let\nxt\pr@@@s
\else\ifx~\next\let\nxt\pr@@@t\else\let\nxt\egroup\fi\fi\nxt}}
```

Někomu mohou nezúžené mezery za čárkou nebo za tečkou končící větu vyhovovat nebo alespoň nevadit, rozhodně by se ale mělo zužovat za tečkou, která není na konci věty — nezúžená mezera v takovém případě (zejména při rozpálení) vypadá rušivě. Příklady: za číslem (určující, že se jedná o číslovku řadovou, např. u data) a u různých zkratk, např. za iniciálami, oslovenými, tituly a přívlastky (p., [?]Dr., dr., ing., Mgr., doc., prof., insp., sv., ml.), zkratkami označení roků (r., př. n. l.), údajů u tiskovin (r., č., sv., sb., str.), či organizací (a. s., s. r. o), větnými zkratkami (např., tzv., resp., příp., tzn., tj., aj., mj., ap., atp., atd., pozn., překl.).

Za tečkou nekončící větu se obvykle používá příkaz `_`, byť standardně s jiným výsledkem — místo rozšířené mezery se použije základní. Chceme-li tento příkaz předefinovat, je třeba přepsat i definici „~“. Pro zúženou nerozdělitelnou mezeru používám příkaz `\~`. Zúžení je provedeno až za mezerou, abychom mohli testovat, zda poslední mezera byla zúžena (viz dále). Problém může být i s tím, že `_` je primitiv a používá se v dalších definicích (např. právě `\~` nebo ve `verbatim` konstrukcích).

```
\let\SP=\ \def~{\nobreak\SP}
\def\ {\SP\ifmmode\else\zuzeni\fi}
\def~{\nobreak\ }
```

Je to ing. J. Antl. Sám.	Je to ing. J.~Antl. Sám.
Je to ing. J. Antl. Sám.	Je to ing.\ J.\~Antl. Sám.

Je asi vhodné zavést zvláštní příkaz pro psaní pomlčky (viz i část 3.1, kde je řešeno i zužování), tam je možné zužování sčítat (například pokud bude sousedit s uvozovkou), sčítání zužování např. mezi zkratkovou tečkou a uvozovkou by nemuselo dopadnout dobře (například proto, že chybí kerning ve spojení „v.“).

ti svatí — jako sv. Petr a jiní	ti svatí~\POMl{} jako sv.~Petr a~jiní
ti svatí — jako sv. Petr a jiní	ti svatí~\= jako sv.\~Petr a~jiní

Zvláštní případ jsou dvojité uvozovky v `cs-fontech`, které v samotné definici obsahují na vnější straně mezery — viz část 3.2. Zúžení kolem uvozevek pak můžeme doplňovat potřebnými příkazy (viz výše) nebo si ho můžeme doplnit do definic příkazů pro české uvozovky.

```
\def\cq #1{\ZuzPred\clq #1\crq \TestZaUV}
\def\cq #1{\ZuzPred\clqq#1\crqq\TestZaUV}
\def\TestZaUV {\afterassignment\ZuzZaUV\let\tmp= }
\def\ZuzZaUV {\ifx.\tmp\UVkern\else\ifx.\tmp\UVkern\fi\fi\ZuzZa}
```

Tato konstrukce zužuje mezeru před otevírací a za uzavírací uvozovkou. V prvním případě ne zúženou (ta v použitých definicích končí zápornou mezerou), ve druhém případě jsou ošetřeny mezery `_`, `\~` a `\space` (a možná další). Je zde doplněn i chybějící kerning mezi uvozovkou a tečkou/čárkou (`\clqq`, `\crqq` jsou předefinovány, aby byla odstraněna nadbytečná „vnější“ mezera, viz část 3.2):

Je to „pěkné“, přestože divné.	Je to <code>\CLQQ pěkné\CRQQ</code> , přestože divné.
Je to „pěkné“, přestože divné.	Je to <code>\clqq pěkné\crqq</code> , přestože divné.
Je to „pěkné“, přestože divné.	Je to <code>"'pěkné'"</code> , přestože divné.

4.7. Pevná zúžená mezera

Nejmenší zúžená mezera, používá se při psaní data číslu, dá se pužit i ustálených zkratek. Měla by to být nejmenší zúžená mezera, což je obvykle právě L^AT_EXovský příkaz `\`, (v plainT_EXu je jen v matematickém modu). Je-li datum psáno číslu, nemá mezi jednotlivými údaji dojít ke zlomu řádku (výjimka se připouští mezu měsícem a rokem pro úzké sloupce).

```
\def\,{\ifmmode\mskip\thinmuskip\else\thinspace\fi}
```

Narodil se 10. 10. 2000 v Praze.	Narodil se 10.~10.~2000 v~Praze.
Narodil se 10. 10. 2000 v Praze.	Narodil se 10.\~10.\~2000 v~Praze.
Narodil se 10.10.2000 v Praze.	Narodil se 10.\,10.\,2000 v~Praze.
Narodil se r. 56 př. n. l.	Narodil se r.\~56 př.\~n.\~l.
Narodil se r. 56 př. n. l.	Narodil se r.\~56 př.\~n.\~l.
Narodil se r. 56 př.n.l.	Narodil se r.\~56 př.\,n.\,l.
Technomat, a. s.	Technomat, a.\s.
Technomat, a. s.	Technomat, a.\s.
Technomat, a. s.	Technomat, a.\,s.

4.8. Výpustková mezera

Je nepružná a nedělitelná, má být mezi tečkami výpustky (elipsy, „tři teček“) a mezi výpustkou a přiléhajícím znakem. Výpustka obvykle přiléhá k předcházejícímu slovu, výjimkou jsou případy, kdy je použita na začátku věty (přiléhá k následujícímu slovu) nebo v neúplném výčtu (za čárkou je oddělena mezerou). Tuto mezeru lze také použít mezi dvěma uvozovkami, aby nesplývaly.

Nejllepší by bylo mít výpustku ve fontu (nejlépe jako ligaturu „...“) s mezerami upravenými kerningy. Současní typografové připouštějí, aby byly hodně malé, podstatné je, aby mezera mezi tečkami a mezi krajní tečkou a přilehlým znakem vypadala stejně. V příkazu `\dots` se používá polovina základní velikosti základní mezery (0,166 em), která je v matematických výrazech za interpunkcí. Je to zároveň velikost pevné zúžené mezery, takže v tomto případě má charakter mezer mezi slovy. Lepší je asi použít polovinu pevné (tj. nejmenší základní) mezery [3]. V `csr10` (ne v `cmr10`) je zaveden kerning mezi tečkami právě o této velikosti (0,111 em), takže lze psát „...“ a doplňovat pouze mezery kolem výpustky. Nabízí se také použití poloviny pevné zúžené mezery (nejmenší mezer mezi slovy za tečkou). Použití poloviny základní velikosti základní mezery mezi *kresbami* teček odpovídá pro `csr10` použití boxů bez mezer.

Zautomatizovat doplňování mezer kolem výpustky předefinováním příkazu `\dots` je komplikované, protože výpustka se může objevit v různých situacích a tento příkaz může být ve zdrojovém textu použit různými způsoby (`\dots`, `\dots{}`, `{\dots}`). Nejjednodušší by bylo přidat do boxu pro výpustku mezery po obou stranách, ty by ale byly rušivé na začátku odstavce a při zlomu řádku.

```
\def\.{\kern0.11111em } \def\Dots {\.\.\.\.}
\let\DOTS=\dots
\def\dots {\ifmmode\DOTS\else\ifvmode\else
\ifdim\lastskip=0pt \.\fi\fi\Dots\fi}
\everypar {\hskip1sp\hskip-1sp\relax}
```

V matematickém módu je ponechána původní definice — jednak kvůli velikosti mezery, jednak proto, že někdy bývá příkaz `\dots` předefinován tak, aby umístil výpustku do vhodné výšky. Test `\ifvmode` má zabránit mezeře před výpustkou na začátku odstavce — to nefunguje, pokud je použit příkaz `\noindent`. V takovém (dosti výjimečném) případě lze použít `\Dots` nebo lze v dokumentu využít uvedený příkaz

`\everypar` — pak by ale na začátku odstavce nefungovalo dobře makro pro psaní pomlčky se sčítáním zúžení, v \LaTeX navíc bývá tento příkaz v různých prostředích (včetně `document!`) využíván. Za výpustku se přidává poloviční mezera, která — oproti původnímu příkazu `\dots` — zmizí při zlomu řádku. Slovo následující za výpustkou bez mezislovní mezery nebude rozděleno — to lze umožnit použitím `\allowhyphens`. Mohli bychom také v definici mezery místo `\kern` použít `\nobreak\hskip`, to ale pro změnu zabrání zlomu řádku v mezeře za výpustkou.

Jejda... Raz, dva, ..., pět.	Jejda... Raz, dva, ..., pět.
Jejda... Raz, dva, ..., pět.	Jejda <code>\dots{}</code> Raz, dva, <code>\dots{}</code> , pět.
Jejda... Raz, dva, ..., pět.	Jejda <code>\DOTS{}</code> Raz, dva, <code>\DOTS{}</code> , pět.

4.9. Mezera před interpunkcí

Před dvojtečkou, středníkem (pokud nejsou za zkratkovou tečkou), vykřičníkem a otazníkem má být úzká mezera. Její velikost má být 1 bod do 12bodového písma a 2 body pro písmo 12bodové až 24bodové. V [3] je uvedeno 0,08 $\bar{3}$ em. Smyslem je opticky oddělit znak od slova, což v jiných případech (tečka, čárka, ...) není nutné. Od tohoto pravidla se někdy upouští (zvláště pro středník, případně i pro dvojtečku). V \TeX je asi lepší použít mezeru 0,05 $\bar{5}$ em (nejmenší násobek osmnáctiny), která se zdá být dostatečná a asi lépe odpovídá tomu, že nejmenší základní mezera je menší než doporučených 0,25 em.

Doplňovat mezeru před dvojtečkou, středníkem, vykřičníkem a otazníkem ve zdrojovém textu není vhodné, protože některé fonty mohou mít tuto mezeru již ve znaku (tzv. nálitek, v `cs-fontech` není). Asi nejlepší cesta je přes aktivní znaky, jak ukazuje příklad pro dvojtečku:

```
\def\SPint {\ifmmode\else\kern.05556em \fi}
\def\dvojtecka{:} \catcode'\:=\active \def:{\SPint\dvojtecka}
```

Test na matematický mód umožňuje použít dvojtečku v matematických výrazech v původním významu. Tato mezera by neměla být mezi znaky (například dvěma vykřičníky) ani za zkratkovou tečkou — to uvedená definice neřeší.

Ahoj! Jak se máš?	Ahoj! Jak se máš?
Ahoj! Jak se máš?	Ahoj! Jak se máš?

5. Matematické výrazy

5.1. Typ písma

- Text v matematických vzorcích píšeme obvykle typem písma, které je v okolí matematického výrazu. To zařídí příkaz `\hbox{...}` nebo `\text{...}` v `amsmath`.

$y = x^2$ pro $x > 0$	<code>\\$y=x^2\quad\hbox{pro \\$x>0\\$}</code>
-----------------------	---

- Identifikátory proměnných se sázejí matematickou italikou, stojatým písmem se sází označení speciálních funkcí, operátorů (například diferenciál), operací a konstant, slova a zkratky, značky jednotek, označení nematematických objektů (částic, chemických prvků).

V \TeX je matematická italika implicitní, stojaté písmo dostaneme přepnutím `\rm`.

$x_{\max} = \int_0^1 e^t \sin t dt$	<code>\\$x_{\rm max}=\int_0^1{\rm e}^t\sin t\,{\rm d}t\\$</code>
$pV = \text{konst.}$	<code>\\$pV={\rm konst.}\\$</code>
1 hl = 100 l	<code>\\$1\:{\rm hl}=100\:{\rm l}\\$</code>
$n \rightarrow p^+ + e^-$	<code>\\${\rm n}\to{\rm p}^+{\rm e}^-\\$</code>
$\text{H}_2\text{SO}_4 \rightarrow \text{SO}_3 + \text{H}_2$	<code>\\$\{\rm H\}_2\{\rm SO\}_4\to\{\rm SO\}_3+\{\rm H\}_2\\$</code>

5.2. Typy matematických objektů

V matematické textu se rozlišují různé typy objektů: obyčejný symbol (Ord, 0), unární operátor či název funkce (Op, 1), binární operace (Bin, 2), relace (Rel, 3), otevírací a uzavírací závorky (Open, 4, Close, 5), interpunkce (Punct, 6), výrazy typu `\left... \right` (Inner), které se liší mezerováním. Například (zjednodušeně) Ord se vedle sebe skládají bez mezery, Op mají kolem sebe mezeru `\,`, Bin `\>`, Rel `\;`, Punct má za sebou mezeru `\,`, Inner má (někdy) kolem sebe mezeru `\,`.

Jednotlivé typy se definují příkazy `\mathord`, `\mathop`, `\mathbin`, `\mathrel`, `\mathopen`, `\mathclose`, `\mathpunct`, `\mathinner`.

Některé znaky mají přednastavený typ, to je možné změnit změnou `\mathcode` příslušného znaku, například dvojtečku předefinujeme na operaci příkazem `\mathcode' := "203A`: číslo je nejpřehlednější zadat hexadecimálně (jak je uvedeno), první cifra je pak třída objektu (viz výše, 7 je pro proměnnou), druhé rodinu a poslední dvě pozici znaku.

Matematický materiál uzavřený do složených závorek `{...}` je vždy typu Ord.

- Chceme-li zavést novou funkci (např tangens), použijeme např.:

```
\def\tg{\mathop{\rm tg}\nolimits}
```

Příkaz `\nolimits` určuje, že indexy budou psány vždy vedle a nikoliv nad a pod nápis „tg“. Implicitně (nebo s příkazem `\displaylimits`) jsou psány nad a pod jedině v display módu, chceme-li, aby byly nad a pod vždy, použijem příkaz `\limits`.

V případě, že definovaný text se skládá z jediného znaku, je centrován na výšku. Tuto skutečnost odstraníme přidáním např. `{}`.

$d(x)$	$d(x)$	$\mathop{\rm{}}d(x)$
--------	--------	----------------------

- Chceme-li definovat operaci, použijem `\mathbin`, pro relaci použijeme `\mathrel`. Pro spojování znaků do relace můžeme použít `\joinrel`.

$a . b$	$\mathbin{.} a b$
$a R b$	$\mathrel{R} a b$
$p \vDash q$	$\mathrel{\vDash} p q$
$p \vDash q$	$\mathrel{\vDash} p \mathrel{\vDash} q$
$p \vDash q$	$\mathrel{\vDash} p \mathrel{\vDash} q$

5.3. Mezery

V matematickém módu se mezery ignorují, \TeX je doplňuje sám, což není vždy ideální. Někdy je třeba dodat informaci o obsahu, někdy dopravit výsledný vzhled. Používáme hlavně `\,`, `\quad` a `\qquad`, pro drobné korekce pak i zápornou mezeru `\!`. Přehled mezer je v tabulce 2. Pro matematické mezery lze používat jednotku „mu“, což je 1/18 em.

- V obyčejném textu ohraničujeme každý matematický výraz zvlášť, interpunkci a mezery necháváme „vně“ v textu. Někdy se kolem matematických výrazů přidává mezera, kterou můžeme v textu automaticky zařídit například předefinováním `\mathsurround=0.08333em` (druhý řádek). Tato mezera by se objevila nevhodně i na začátku odstavce, odstavec by ale neměl začínat matematickým výrazem.

Délky stran jsou a, b a c .	Délky stran jsou a , b a c .
Délky stran jsou a, b a c .	Délky stran jsou a , b a c .

- Mezi výrazem a interpunkcí se ve vzorcích píše mezera, podle [7] má být 0,25 em. Tam se ale tato mezeru doporučuje všude, v \TeX u je jemnější třídění (viz 5.2), nejspíš stačí `\,`. V některých nakladatelstvih používají `\;`, která má základní velikost trochu větší než 0,25 em (0,277 em) a je

zápis	velikost v em	název/použití
Původní mezery		
<code>\quad</code>	2	oddělení vzorců
<code>\quad</code>	1	oddělení vzorců menší
<code>\enspace</code>	0,500	šířka cifry
<code>_</code> , <code>\~</code>	0,333̄ (0,222̄–0,500)	mezislovní
<code>\;</code>	0,277̄ (0,277̄–0,555̄)	kolem relací
<code>\> [[:</code>	0,222̄ (0,000–0,333̄)	kolem operací
<code>\,</code>	0,166̄	kolem operátorů, za interpunkcí
<code>\!</code>	–0,166̄	korekce
Možná do- či předefinování		
<code>\:</code>	0,277̄	šířky tečky

Tabulka 2: Matematické mezery (v závorce rozmezí pro pružné).

navíc natahovací (k tomu v display modu asi nedojde). Mezera `\:` (v plainu `\>`), má možná vhodnější základní velikost 0,222̄ em, ale může se stáhnout (k tomu může snadno dojít), dokonce až na 0 em.

$$\begin{array}{ll}
 x = \frac{1}{2}, & a = 5. & \text{\texttt{\$x=\frac12,\quad a=5.\$}} \\
 x = \frac{1}{2}, & a = 5. & \text{\texttt{\$x=\frac12\,,\quad a=5\,,.\$}} \\
 x = \frac{1}{2}, & a = 5. & \text{\texttt{\$x=\frac12\:\,,\quad a=5\:\,.\$}} \\
 x = \frac{1}{2}, & a = 5. & \text{\texttt{\$x=\frac12\;\,,\quad a=5\;\,.\$}}
 \end{array}$$

- Píšeme-li několik výrazů za sebou, oddělujeme je pomocí `\quad` nebo `\quad`. Vpisujeme-li text, můžeme použít i oddělování pomocí `_`, je-li delší, pak příslušnou část uzavíráme do hboxu (příkaz `\text` v AMSI \LaTeX U).

$$\begin{array}{ll}
 x > 5, & y > x + 4. & \text{\texttt{\$x>5\,,\quad y>x+4\,,.\$}} \\
 t'' = 0, & t(0) = 1, & t'(0) = 2. & \text{\texttt{\$t''=0\,,\quad t(0)=1\,,\quad t'(0)=2\,,.\$}} \\
 y = f(x, y), & \text{pro } x > 0. & & \text{\texttt{\$y=f(x,y)\,,\quad \hbox{pro } x>0\$.}}
 \end{array}$$

- Mezery doplňujeme za argumentem funkce (či závorkou, je-li v ní), za faktoriálem, odmocninou, kolem diferenciálu, zlomků, mezi rovnými závorkami (pokud je tam součin)...:

$$\begin{array}{lll}
 u'(x)v(x) - u(x)v'(x) & u'(x)v(x) - u(x)v'(x) & \text{\texttt{\$u'(x)\,v(x)-u(x)\,v'(x)\$}} \\
 \log n(\log \log n)^2 & \log n(\log \log n)^2 & \text{\texttt{\$log n\,(\log\log n)^2\$}} \\
 n!m!(m+n)! & n!m!(m+n)! & \text{\texttt{\$n!\,m!\,(m+n)!\$}} \\
 xdy - ydy & xdy - ydy & \text{\texttt{\$x\,{\rm d}y-y{\rm d}y\$}} \\
 xdyh^2 & xdyh^2 & \text{\texttt{\$x\,{\rm d}y\,h^2\$}} \\
 \partial x \partial y & \partial x \partial y & \text{\texttt{\$\partial x\,\partial y\$}} \\
 \Delta n_1 \Delta n_2 & \Delta n_1 \Delta n_2 & \text{\texttt{\$\Delta n_1\,\Delta n_2\$}} \\
 \nabla a \nabla b & \nabla a \nabla b & \text{\texttt{\$\nabla a\,\nabla b\$}} \\
 \forall x \exists y & \forall x \exists y & \text{\texttt{\$\forall x\,\exists y\$}} \\
 \sqrt{2}x & \sqrt{2}x & \text{\texttt{\$\sqrt{2}\,x\$}} \\
 O(1/\sqrt{n}) & O(1/\sqrt{n}) & \text{\texttt{\$O\bigl(1/\sqrt{n}\,\bigr)\$}} \\
 \frac{1}{2} \frac{x}{x+y} & \frac{1}{2} \frac{x}{x+y} & \text{\texttt{\$\frac12\,\frac{x}{x+y}\$}} \\
 |xy| = |x||y| & |xy| = |x||y| & \text{\texttt{\$|xy|=|x|\,|y|\$}}
 \end{array}$$

- Někdy (asi lépe) se přidávají mezery i před název funkce (jak je tomu při zavedení pomocí `\mathop`).

$$\begin{array}{lll}
 yf(x) + 2g(y) & yf(x) + 2g(y) & \text{\texttt{\$y\,f(x)+2\,g(y)\$}} \\
 x + 4e^x e^y & x + 4e^x e^y & \text{\texttt{\$x+4\,{\rm e}^x\,{\rm e}^y\$}}
 \end{array}$$

- Obráceně rušíme automatickou mezeru před operátorem, např. po lomítku nebo odmocnině nebo při skládání operátorů:

$n/\log n$	$n/\log n$	$\$n/\!\log n\$$
$x/\sin x$	$x/\sin x$	$\$x/\!\sin x\$$
$\int_0^x \int_0^y dF$	$\int_0^x \int_0^y dF$	$\$\int_0^x\!\int_0^y\{\rm d}F\$$
\iint	\iint	$\$\!\int\!\int\$$

- Při použití konstrukce `\left(...\right)` se přidávají na krajích (ale až za případným indexem) mezery, které je někdy nutno zrušit.

$\Gamma(...)$	$\Gamma(...)$	$\$\Gamma\!\left(\dots\right)\$$
---------------	---------------	----------------------------------

- Drobné typografické korekce pomocí `\`, a `\!` – za exponentem (pokud je tam šikmý znak), indexy za některými znaky, výrazy poblíž závorek:

$x^2/2$	$x^2/2$	$\$x^2\!/2\$$
$\partial^2 f$	$\partial^2 f$	$\$\partial^2\!f\$$
$\Gamma_2 + \Delta^2$	$\Gamma_2 + \Delta^2$	$\$\Gamma_{\!2}+\Delta^{\!2}\$$
$R_i^j{}_{kl}$	$R_i^j{}_{kl}$	$\$R_{i\!}^{\!j}\!_{\!kl}\$$
$(\dots)^n$	$(\dots)^n$	$\$\Bigl(\dots\Bigr)^{\!n}\$$
$[0, 1]$	$[0, 1]$	$\$[\!,0,1]\$$
$\left(x + \frac{1}{2}\right)$	$\left(x + \frac{1}{2}\right)$	$\$\!\biggl(\!x+\frac{1}{2}\biggr)\!\!\$$
$\sqrt{\log x}$	$\sqrt{\log x}$	$\$\sqrt{\!,\log x}\$$

5.4. Speciální symboly

Někdy se najdou stejně vyhlížející symboly, které se používají v různém významu. Pak se používají buď speciální příkazy nebo můžeme symbol přetypovat (viz 5.2).

- *Tečka „.*“ je typu Ord, jako desetinnou tečku ji můžeme použít bez úprav. Ve významu Punct se píše `\ldotp` (to se ale používá snad jen jako část elipsy), ve významu Bin je třeba ji předefinovat nebo použít `\cdot` (ta neleží na řádku, což je s ohledem na ostatní operace lepší).

$3.2^x - 5$	$3.2^x - 5$	$\$3\mathbin{.}2^x-5\$$
$3.2^x - 5$	$3 \cdot 2^x - 5$	$\$3\cdot 2^x-5\$$

- *Čárka „,*“ je typu Punct. Chceme-li ji použít ve významu desetinné čárky, je třeba ji přetypovat na Ord:

$3,141$	$3,141$	$\$3\{,}141\$$
---------	---------	----------------

- *Dvojtečka „:*“ je typu Rel ($a : b = c : d$), jako Punct ji píšeme pomocí `\colon`. V češtině se asi nejvíc používá jako operace dělení (globálně ji můžeme předefinovat pomocí `\mathcode' := "203A`) a jako interpunkce v definicích množin a funkcí – tam by byla za ní měla být mezerka o něco větší, než je kolem operací (místo `\colon\;` můžeme psát `\, \{ : \} \, \;`; a případně povolit zlom řádku).

$1 + 6 : 3 = 3$	$1 + 6 : 3 = 3$	$\$1+6\mathbin{:}3=3\$$
$A = \{x : x > 5\}$	$A = \{x : x > 5\}$	$\$A=\{x\!,\colon\!;x>5\}\$$
$f : A \rightarrow B$	$f : A \rightarrow B$	$\$f\!,\colon\!;A\to B\$$

- *Svislá čára „|“* (`\vert`) je typu Ord. Pro absolutní hodnotu dává občas špatné mezerování (před unárním plus/minus, před operátorem), lepší je použití dvojici `\mathopen|` a `\mathclose|` nebo `\left|` a `\right|` (ta občas přidává mezery). Jako Rel se používá `\mid`, v definici množiny je to stejné, jako pro dvojtečku.

$ - x $	$ -x $	<code>\mathopen -x\mathclose </code>
$ \sin x $	$ \sin x $	<code>\left \sin x\right </code>
$2 16$	$2 \mid 16$	<code>2\mid 16</code>
$A = \{x x > 5\}$	$A = \{x \mid x > 5\}$	<code>A=\{x\, \,x>5\}</code>

- *Zdvojená svislá čára.* Nepoužívá se `||`, ale `\|` (`\Vert`) typu Ord. Pro normu dává občas špatné mezerování (viz svislá čára). Jako Rel se používá `\parallel`.

$\ x\ $	$\ x\ $	<code>\ x\ </code>
$\ - x \ $	$\ - x \ $	<code>\left\ -x\right\ </code>
$p\ q$	$p \ q$	<code>p\parallel q</code>

- *Lomené závorky, nerovnosti:* „<“ a „>“ jsou typu Rel a vyznačují nerovnosti, závorky se píší příkazy `\langle` a `\rangle`. Ne tak otevřené závorky se používají v kvantové mechanice (nutno speciální font). Pro vyjádření „mnohem menší/větší“ se používají příkazy `\ll`, `\gg`, v `amssymb` jsou definovány ještě příkazy `\lll`, `\ggg`.

$x \in \langle 0, 1 \rangle$	$x \in \langle 0, 1 \rangle$	<code>x\in\langle 0,1\rangle</code>
$x \ll 1$	$x \ll 1$	<code>x\ll 1</code>
$x \lll 1$	$x \lll 1$	<code>x\lll 1</code>

- *Velká písmena sigma a pi:* `\Sigma` a `\Pi` jsou typu Ord, podobně vypadající symboly pro součet a součin typu Op se píší `\sum` a `\prod`.

$\sum_{i=1}^n \prod_{k=1}^i (x - x_k)$	$\sum_{i=1}^n \prod_{k=1}^i (x - x_k)$	<code>\sum_{i=1}^n \prod_{k=1}^i (x-x_k)</code>
--	--	---

5.5. Závorky

Vnořené závorky se řadí takto: $\{[(\dots)]\}$. Lze taky mít závorky stejného typu, vnější poněkud větší. Závorky mají být vysoké zhruba tak, jako nejvyšší výraz v nich obsažený. Automaticky to lze zařídit použitím konstrukce typu `\left(\dots\right)` (pro kulaté závorky), která najde nejmenší dostupnou závorku větší (nebo stejnou?) než materiál uvnitř.

To ale z různých příčin někdy selhává a je třeba použít explicitního zvětšování pomocí `\big`, `\Big`, `\bigg`, `\Bigg`, obvykle s písmeny `l/r/m` (left/right/medium – levá/pravá/prostřední[tj. jako relace, přidává mezery] na konci. Velikosti jsou 1, 1,5, 2, 2,5 řádku, `\big` a `\bigg` jsou vhodné v případě použití zlomků (v textovém resp. display módu). V \LaTeX u bývají problémy s funkčností těchto příkazů pro jiné než 10pt písmo – to je vyřešeno v balících `amsmath` a `exscale`.

Příklady problémů spojených s použitím `\left\dots\right)`.

- Někdy přidává mezery (to by šlo odstranit použitím `\!`).

$f\left(\frac{1}{2}\right)$	$f\left(\frac{1}{2}\right)$	<code>f\bigl(\frac{1}{2}\bigr)</code>
-----------------------------	-----------------------------	---------------------------------------

- Neumožňuje rozdělit formuli na více řádků (uvnitř je jedna skupina). To lze obejít tím, že se formule rozdělí s doplněním „neviditelných“ závorek `\right.\left..` Aby byla zajištěna stejná výška, přidá se do „nižší“ části třeba `` obsahující nejvyšší část vyšší části.

- Vnořujeme-li do stejného typu závorek, chceme, aby vnější byly větší, i když už vnitřní jsou dosta-

tečně vysoké.

$$((a + b) + c) \qquad ((a + b) + c) \qquad \mathbf{\bigl((a+b)+c\bigr)}$$

- Vybraná velikost může být příliš velká, zatímco nižší by byla jen nepoznatelně menší. (? příklad)
- Někdy může být vhodnější i poznatelně menší závorka.

$$(0, \frac{\pi}{2}) \qquad (0, \frac{\pi}{2}) \qquad \mathbf{\bigl(0, \frac{\pi}{2}\bigr)}$$

$$\left(1 + \sum_{k=1}^n k\right) \qquad \left(1 + \sum_{k=1}^n k\right) \qquad \mathbf{\biggl(1 + \sum_{k=1}^n k\biggr)}$$

5.6. Vektory

Vektory by se měly psát polotučným kurzívním gilem, kterému se nejvíce blíží písmo vytvořené D. Nečasem (písmeno „a“ by mělo být *kurzívní*) a šířené v souborech `cmvec`. [`mf|tfm`]. Pro srovnání uvádím i další možnosti – šipky, tučná matematika, tučné stojaté bezpatkové písmo – a jejich zavedení v \LaTeX u.

- $\vec{a}, \vec{b}, \vec{c}, \vec{d}, \vec{e}, \vec{f}, \vec{g}, \vec{h}, \vec{i}, \vec{j}, \vec{k}, \vec{l}, \vec{m}, \vec{n}, \vec{o}, \vec{p}, \vec{q}, \vec{r}, \vec{s}, \vec{t}, \vec{u}, \vec{v}, \vec{w}, \vec{x}, \vec{y}, \vec{z}$
 $\vec{A}, \vec{B}, \vec{C}, \vec{D}, \vec{E}, \vec{F}, \vec{G}, \vec{H}, \vec{I}, \vec{J}, \vec{K}, \vec{L}, \vec{M}, \vec{N}, \vec{O}, \vec{P}, \vec{Q}, \vec{R}, \vec{S}, \vec{T}, \vec{U}, \vec{V}, \vec{W}, \vec{X}, \vec{Y}, \vec{Z}$
- `\DeclareMathAlphabet {\mathb}{OT1}{cmm}{b}{it}`
 $a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z$
 $A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z$
- `\DeclareMathAlphabet {\mathsfbx}{OT1}{cmss}{bx}{n}`
 $a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z$
 $A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z$
- `\DeclareFontShape {OT1}{cmss}{bx}{sl}`
`{ <5><6><7><8><9><10>cmvec10 <10.95><12><14.40><17.28>cmvec10 }`
`\DeclareMathAlphabet {\mathslsfbx}{OT1}{cmss}{bx}{sl}`
 $a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z$
 $A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z$

5.7. Opakování operací

Dojde-li ke zlomu řádku za operací, pak by se znak operace měl na začátku řádku zopakovat. (???)chybí)

6. Problémy s definicemi

Některé problémy řešíme různými definicemi, přitom mohou nastat jiné problémy.

- Pro konstrukce typu `verbatim` je třeba doplnit seznam speciálních znaků v `\dospecials` (v seznamu jsou znaky `\, \{, \}, \$, \&, \#, \^, _, \%, \~, \^A [??], \^K [tabelátor]`) o nově zavedené aktivní znaky a případně vrátit původní význam příkazu `_`. Příklad použití pro " a pro uložení původní definice `_` do `\SP`:

```
\let\DOSpecials=\dospecials \def\dospecials{\let\ =\SP\DOSpecials\do\}
```

- V některých definicích se používají konstrukce s `\if...` Tyto definice jsou pak tzv. „křehké“ a je problém s jejich použitím např. při zápisu do souboru (např. u \LaTeX ovských příkazů dávajících

nadpisy či popisy). Při zápisu se totiž expandují aniž by se podmínka testovala, je tedy nutno expanzi zabránit (v \LaTeX u příkazem \protect).

```
\catcode'\_=\active \def_{\ifmmode\sb\else\kern.25em\fi}
...
\section{...$a\protect_b$..}
```

V \LaTeX u lze též definovat příkaz pomocí $\text{\DeclareRobustCommand}$ — nelze tak ale přímo definovat aktivní znak (předefinoval by se i příkaz _).

```
\DeclareRobustCommand\pevna {\ifmmode\sb\else\kern.25em\fi}
\catcode'\_=\active \let_=\pevna
```

- S aktivními znaky může být řada problémů: jednak jejich účinnost je až od okamžiku předefinování (například předefinování dvojtečky se neprojeví v příkazu \caption), jednak mohou vadit v místě, kde chceme obyčejný znak, ne aktivní. Například podmínka \split u \czech.sty řeší opakování spojovníku na dalším řádku přes aktivní znak. Pak se ale například při použití příkazu \cline hlásí chyba. V podobných situacích je třeba vrátit znaku původní kategorii — v tomto případě je zaveden příkaz \standardhyphens (zpátky \splithyphens), obvykle se předefinuje na kategorii 12 (Other), podtržítka na kategorii 8 (Subscript). Ukažme si příklad, jak vyřešit, aby v názvu souboru mohlo být jinak aktivní podtržítka:

```
\let\INPUT=\input
\def\input {\catcode'\_ =8 \Input}
\def\Input #1 {\catcode'\_ =\active\INPUT #1 }
```

Reference

- [1] *Základní pravidla sazby*. ON 88 2503, 1975.
- [2] Havelka, J.: *Počítačová typografie pro každého*. Grada Publishing, Praha, 1995.
- [3] Kočíčka, P., Blažek, F., Mohelská, L.: *Praktická typografie*. Computer Press, Praha, 2000.
- [4] Nečas, D.: <http://www.physics.muni.cz/~yeti/tex/>
- [5] Pop, P., Fléger, J., Pop, V.: *Sazba I. Ruční sazba*. Státní pedagogické nakladatelství, Praha, 1984.
- [6] Tkadlec, J.: *Textové mezery v \TeX u*. Zpravodaj CSTUG **12** (2002), č. 1, str. 21–35.
<http://math.feld.cvut.cz/tkadlec/ostatni.htm>
- [7] Wick, K.: *Sazba matematických vzorců čtyřřádkovým způsobem*. Nakladatelství ČSAV, Praha, 1961.