Zpravodaj Československého sdružení uživatelů TeXu

Barbara Beeton
Co by každý (La)TeXový nováček měl znát

Zpravodaj Československého sdružení uživatelů TeXu, Vol. 33 (2023), No. 3-4, 139–152

Persistent URL: http://dml.cz/dmlcz/151996

Co by každý (LA)TEXový nováček měl znát

Barbara Beeton

O LATEXu se říká, že se v něm dají vytvořit kvalitně vypadající dokumenty, ale za cenu toho, že je složitější se jej naučit. To je sice pravda, nicméně tyto obavy je možné zmírnit, když pochopíme několik základních principů a naučíme se vyhýbat některým technikám, které se mohou zdát jasné, avšak nevedou k cíli. Cílem tohoto článku je ukázat začínajícím uživatelům dobré LATEXové návyky dříve, než se začnou projevovat ty špatné.

Klíčová slova: LATEX, nováček, chyby, log soubor

Úvod

V tomto článku budu popisovat situace, se kterými jsem se v průběhu svého pracovního života často setkávala, a to zejména na dvou místech:

- při práci v týmu TEXnické podpory významného matematického nakladatelství, kde bylo nutné odpovídat na dotazy autorů a psát uživatelskou dokumentaci,
- na online TEXovém fóru na StackExchange [2], kde uživatelé pokládají hromady dotazů od jednoduchých až po velmi pokročilé. Členové komunity sepsali na fóru seznam často odkazovaných odpovědí [3].

Několikrát zde budu opakovat obecnou radu: Přečtěte si dokumentaci.

Důležité pojmy

Existuje několik pojmů, které se k začínajícím uživatelům nedostanou, nebo se k nim dostanou, ale jsou pro ně nesrozumitelné. Pojďme se na ně podívat.

Šablona

Mnoho nových (LA)TEXových[1] uživatelů si myslí, že třída dokumentu (soubor s příponou .cls) je šablonou pro konkrétní typ dokumentu. I když jde tato myšlenka správným směrem, není to úplně pravda. Šablonou bývá zdrojový soubor

Z anglického originálu [1] přeložil Jan Šustek.

[1] Použijeme-li v článku výraz „(LA)TEX“, týká se příslušný text jak programu TEX, tak jeho nadstavby LATEX. Pokud závorky nepoužijeme, týká se text pouze LATEXu. (pozn. překl.)

DOI: 10.5300/2023-3-4/139

s příponou .tex, který obsahuje kostru dokumentu napsaného v daném stylu. Soubor začíná příkazem `\documentclass` a ukazuje použití základních příkazů v dokumentu. Do něj je pak možné přidat vlastní definice a text dokumentu. V ideálním případě lze šablonu zpracovat (LA)TEXem bez chyb, nicméně výsledek zřejmě nebude nijak užitečný.

Příkazový řádek

Většina nových uživatelů v současnosti spouští (LA)TEX z editoru nebo z jiného grafického rozhraní, přičemž jej spouští neinteraktivně a čekají, dokud program neskončí (s chybou, nebo bez), nebo se nezacyklí. Spouštění z příkazového řádku naopak umožňuje s programem komunikovat a v určitých situacích opravit chybu za běhu, nebo, pokud to není možné, ukončit program dříve, než začne vypisovat množství neužitečných chybových zpráv.

Jedním druhem chyb opravitelných za běhu je překlep v názvu příkazu.

```
! Undefined control sequence.
l.137 \scetion
                {Nadpis}
?
```

Uživatel může překlep opravit napsáním

```
i\section
```

a stiskem klávesy Enter. Samozřejmě je potom nutné chybu opravit také ve zdrojovém souboru.

Na druhou stranu, překlep v názvu prostředí takto opravit nelze. V takovém případě uživatel napíše `x` a stiskne Enter, čímž se program ukončí. Po opravě zdrojového souboru program spustí znovu.

Jestliže po neopravitelné chybě uživatel program neukončí, budou pravděpodobně následovat další chyby způsobené tím, jak se (LA)TEX snažil ze situace zotavit. Příslušné chybové zprávy už ale budou docela matoucí. Tomu je lepší se vyhnout a program ukončit.

Soubor .log

Vždy, když se spustí (LA)TEX, vytvoří se textový soubor s příponou .log. Soubor se vytvoří ve stejném adresáři jako hlavní zdrojový soubor – najděte si jej. Soubor obsahuje všechny chybové zprávy a varování, ale obsahuje také seznam všech načítaných souborů, u načítaných tříd a balíčků navíc i s uvedením jejich verze. Dále obsahuje čísla zpracovaných stran a informace o použitých zdrojích. V článku se zaměříme pouze na některé z těchto věcí. V dřívějším článku [4] (a jeho českém překladu [5]) je popsáno ladění zdrojového souboru také ve složitějších situacích.

Základní struktura

V této sekci probereme některé základní prvky (LA)TEXu.

Příkazy

Jednotlivé instrukce se (LA)TEXu předávají pomocí příkazů (tzv. řídicích sekvencí), které zpravidla začínají znakem \ (zpětné lomítko). Existují dva typy řídicích sekvencí:

- zpětné lomítko následované jedním znakem jiným než písmeno (tzv. řídicí znak),
- zpětné lomítko následované jedním nebo více písmeny (tzv. řídicí slovo), která mohou být malá i velká (přičemž na jejich velikosti záleží), avšak nemůže se jednat o číslice ani žádné speciální znaky.[2]

Řídicí sekvence mohou mít jeden nebo více argumentů (`\title{...}`) nebo mohou být bez nich (`\alpha`). Řídicí slovo bývá ukončeno mezerou nebo libovolným jiným nepísmenným znakem, přičemž takovou ukončovací mezeru TEX odstraní. Naopak mezeru za řídicím znakem TEX neodstraní a projeví se na výstupu. V (LA)TEXu jsou některé řídicí znaky již nadefinovány tak, že vysází příslušný znak. Například `\#`, `\%`, `\$` a `\&` vysází #, %, $ a &.

Uživatel může nadefinovat vlastní příkazy nebo může předefinovat stávající. LATEX nabízí příkaz `\newcommand` k nadefinování nového příkazu. Přitom se kontroluje, jestli příkaz se stejným jménem již nebyl nadefinován dříve, a pokud byl, pak definici neprovede a vypíše chybovou zprávu. (Primitivní TEXový příkaz `\def` takovou kontrolu neprovádí.) Pro předefinování existujícího příkazu slouží LATEXový příkaz `\renewcommand`.

Při předefinování si musíte být naprosto jistí, že víte, co děláte! Například předefinování příkazu `\par` je hodně riskantní, protože LATEX používá příkaz `\par` uvnitř svých příkazů skrytě před uživatelem a jeho jiná definice snadno způsobí mnoho chyb.

Řídicí slova obsahující jediné písmeno také není moc vhodné předefinovávat, protože mnoho z nich slouží k sazbě akcentů nebo k sazbě různých neanglických písmen. Tato písmena se mohou vyskytnout například v cizojazyčných jménech. Představme si, že v článku o komplexních číslech nadefinujeme příkaz pro imaginární jednotku a zároveň budeme citovat autora Cema Yıldırıma.

```
\renewcommand{\i}{\ensuremath{\sqrt{-1}}}
Cem Y\i ld\i r\i m
```

Pak se nemůžeme divit, když na výstupu dostaneme

Cem Y$\sqrt{-1}$ld$\sqrt{-1}$r$\sqrt{-1}$m

[2] Diskuse o kategoriích znaků a přesných pravidlech tvorby řídicích tokenů přesahuje rámec tohoto článku. Pro detaily může čtenář nahlédnout do [6]. (pozn. překl.)

Řídicí znaky obsahující cifru (`\0`, `\1` atd.) nejsou v LaTeXu nadefinovány a jsou uživatelům k dispozici pro vlastní příkazy.

Prostředí

Pro LaTeX je charakteristické používání prostředí, což je blok kódu mezi

`\begin{`⟨*prostředí*⟩`}` a `\end{`⟨*prostředí*⟩`}`

Název prostředí na začátku i konci se musí shodovat. V opačném případě se vypíše chybová zpráva

`! LaTeX Error: \begin{`⟨*prostředí1*⟩`} on input line` ⟨*číslo*⟩
`ended by \end{`⟨*prostředí2*⟩`}.`

Většinu prostředí je možné vnořit do jiných prostředí. Pak je ale nutné jednotlivá prostředí ukončovat ve správném pořadí.

K definování příkazů LaTeX obsahuje ještě příkaz `\newenvironment` a novější příkazy `\NewDocumentCommand`, `\NewDocumentEnvironment` a ke všem analogické příkazy pro předefinování. Pro jejich popis si přečtěte dokumentaci.[3]

Módy

Zjednodušeně řečeno, aktuální mód určuje, kde na výstupu se právě nacházíme. My se však na módy podíváme na úrovni vstupního souboru.

Módy jsou tři: vertikální, horizontální a matematický.

(La)TeX je ve vertikálním módu na začátku dokumentu, po příkazu `\par` nebo po vynechaném řádku ve vstupním souboru.

Do horizontálního módu se dostaneme, když začneme psát text. Pro přechod z vertikálního do horizontálního módu je také možné použít příkazy `\indent`, `\noindent` nebo `\leavevmode`. V horizontálním módu se několik po sobě jdoucích mezer považuje za jedinou mezeru. Tady je důležité, že se jedná o mezery jdoucí po sobě. Na začátku řádku jsou mezery ignorovány. Konec řádku se také považuje za mezeru, i když ve vstupním souboru není vidět. Některé editory při zalamování textu automaticky vkládají znak konce řádku, jiné ne, a také na různých operačních systémech je znak konce řádku definován různě. Tyto odchylky TeX ihned na vstupu zohlední a uživatel pak nepozná rozdíl. Více si o tom povíme později.

Matematický mód je určen pro sazbu vzorců. Vzorec můžeme vložit přímo do textu odstavce nebo vycentrovat na zvláštní řádek. Vzorce v textu vysázíme pomocí `$...$` nebo `\(...\)`. Jednořádkové nečíslované vycentrované vzorce vysázíme pomocí `$$...$$` nebo `\[...\]`. Pro víceřádkové vzorce jsou v LaTeXu

[3] V dokumentaci [7] se píše, že příkazy `\NewDocumentCommand`, `\NewDocumentEnvironment` a jim podobné byly do jádra LaTeXu přidány až v roce 2020, proto jejich popis v dřívější dokumentaci asi nenajdete. (pozn. překl.)

balíčky amsmath nebo mathtools. (Přečtěte si dokumentaci. Balíček amsmath se načítá uvnitř balíčku mathtools, proto není nutné načítat oba balíčky.) Vycentrované vzorce obvykle patří k předchozímu odstavci, proto nad vzorcem nevynechávejte prázdný řádek (to by mohlo vést i k nesprávnému zalomení stránky nad vzorcem).

Uvnitř matematického módu není dovoleno vložit prázdný řádek. To je rozhodnutí Knutha s cílem jednoduššího nalezení chyb ve zdrojovém souboru. Matematická sazba totiž nikdy nepokračuje do dalšího odstavce.

Skupiny

Dalším ze základních prvků je skupina. Ta umožňuje lokálně omezit platnost různých nastavení a definic.

Skupinu tvoří například matematický mód. Určité symboly a operace jsou možné pouze uvnitř matematického módu a jinde nejsou povoleny. V textu odstavce matematický mód obvykle začíná a končí znakem `$` a tyto znaky musí být spárovány. Začátek centrovaného vzorce přeruší tok textu a na jeho konci se TEX vrátí zpět do horizontálního módu, případně, pokud následuje prázdný řádek nebo příkaz `\par`, do vertikálního módu. Více si o tom povíme později.

Další možnost vytvoření skupiny je uzavřít text do složených závorek `{...}`. Uvnitř skupiny můžeme dočasně předefinovat příkaz a ten se po ukončení skupiny vrátí na svou původní definici. Namísto složených závorek můžeme se stejným výsledkem[4] uzavřít text mezi příkazy `\begingroup` a `\endgroup`.

Skupina se také automaticky vytvoří, když uzavřeme materiál do boxu. Materiál se uzavře do boxu také v prostředí `minipage`, příkazech `\mbox`, `\parbox` nebo například v příkazech balíčku tcolorbox.

Některá prostředí (avšak ne všechna) také vytvářejí skupiny. Například prostředí `theorem` – v něm se text přepíná do kurzívy a po jeho ukončení se přepne zpět.

Mezery v textu

Cílem vysoce kvalitní sazby je, aby všechny mezery v textu byly stejně široké. To je ale možné pouze v případě sazby na praporek (tzn. odstavec je zarovnaný pouze na jedné straně), kdy mezery mají svou přirozenou šířku. Obvykle se však preferuje zarovnání odstavce do bloku a TEX je naprogramovaný tak, aby i při zarovnání do bloku byly šířky mezer podobné.

V anglických dokumentech[5] má být mezera za koncem věty širší než ostatní mezislovní mezery. To TEX dělá. V dokumentech v jiných jazycích je možné toto

[4] Existují jasně popsané situace, ve kterých se výsledek mírně liší, jejich popis však přesahuje rámec tohoto článku. Pokročilejší čtenář může nahlédnout do [6]. (pozn. překl.)

[5] Tato podsekce se týká zejména anglických dokumentů, případně částí dokumentů, v nichž je jako jazyk nastavena angličtina. (pozn. překl.)

chování vypnout příkazem `\frenchspacing`.[6] Nicméně akademické dokumenty jsou často psány anglicky a obsahují dost zkratek, a proto není pro TEX vždy zřejmé, kde končí věta. Aby za zkratkou následovala normálně široká mezera, píšeme za zkratku `\␣` (zpětné lomítko a mezera). Porovnejte

`abc␣vs.␣xyz` (abc vs. xyz) oproti `abc␣vs.\␣xyz` (abc vs. xyz).

Jestliže se za zkratkou nemá zlomit řádek, použijte znak `~` (vlnka).

```
See Fig.~37 on p.~159.
```

Podobná, ale opačná situace může nastat, když tečka následuje za velkým písmenem. TEX předpokládá, že se jedná o počáteční písmeno jména, a za tečku pak vkládá normální mezislovní mezeru. Avšak někdy je velké písmeno součástí zkratky a ta se může vyskytnout na konci věty. V LATEXu je definován příkaz `\@`, který se v takových případech vkládá mezi ono velké písmeno a tečku, aby se mezera za tečkou chovala jako mezera za koncem věty.

```
I like CSTUG\@.
```

Z předchozích odstavců dostáváme jednoduché pravidlo: S výjimkou konce věty (a v menší míře také za dalšími interpunkčními znaménky nebo uvnitř matematických výrazů) mají být všechny mezery na jednom řádku stejně široké. Pokud nejsou, něco není v pořádku.

Zavlečené mezery

Může se stát, že na výstupu dostaneme několik mezer za sebou. V převážné většině je to důsledkem toho, že se snažíme definovat nové příkazy tak, aby definice byly dobře čitelné. Mějme například definici

```
\newcommand{\abc}{
  \emph{abc def}
}
```

Potom při vstupu

```
nějaký text \abc\ další text
```

dostaneme na výstupu

nějaký text *abc def* další text

Mezery navíc kolem šikmého textu nesprávně vložil náš příkaz `\abc`. Jak jsme si řekli dříve, konce řádku TEX považuje za mezery. Proto musíme použít znak `%`, ten uvozuje komentář a TEX ignoruje vše, co je za ním až do konce řádku, včetně znaku konce řádku. S opravenou definicí

[6] Pokud v (LA)TEXu máme nastaveno, že aktuální jazyk je čeština nebo slovenština, pak je `\frenchspacing` automaticky nastaveno vnitřními makry. (pozn. překl.)

```
\newcommand{\abc}{%
  \emph{abc def}%
}
```

dostaneme správný výsledek

nějaký text *abc def* další text

Další příčinou zavlečených mezer může být přítomnost několika po sobě jdoucích objektů, které nejsou součástí hlavního textu, například poznámek pod čarou nebo položek v rejstříku.

```
Důležitý pojem\index{pojem}
  \index{jiný název}
  \index{varianta názvu}
je v~rejstříku uveden na několika místech.
```

V tomto případě se každý znak konce řádku převede na mezeru, a protože tyto mezery nejsou na vstupu souvislé, na výstupu do jediné mezery nesplynou.

Důležitý pojem je v rejstříku uveden na několika místech.

Opět je řešením použít znak `%`. Přitom ale musíme dát pozor, aby nám zůstala přesně jedna mezera.

```
Důležitý pojem\index{pojem}%
  \index{jiný název}%
  \index{varianta názvu}
je v~rejstříku uveden na několika místech.
```

Ne vždy je procento řešením

Připomínám, že mezeru, která ukončuje řídicí slovo, TEX odstraní a použití znaku `%` není nutné. Na některých místech může použití znaku `%` dokonce způsobit problémy.

Při načítání číselných hodnot hlavní procesor TEXu pokračuje ve čtení hodnoty, dokud nenarazí na jiný znak než číslici. Proto když na konci řádku máme `\xyz=123`, neměl by následovat znak `%`. Kdyby totiž následoval a další řádek by začínal číslicí, byla by tato číslice stále součástí načítané hodnoty.

Podobně když TEX načítá hodnotu pružné délky, například `\parskip=2pc`, zjišťuje, zda následují klíčová slova `plus` nebo `minus`. Je lepší načítání ukončit prázdnou skupinou `{}`. Kdybychom takto načítání neukončili a zároveň by `plus` nebo `minus` bylo součástí navazujícího textu, ohlásil by TEX chybu a příslušná zpráva by pro uživatele byla dost matoucí.

Opravdu nečekané mezery

Některé situace nejsou na první pohled předvídatelné. Tento případ se objevil v jednom online dotazu. Dokument používal balíček colorbox[7] a pomocí něj autor podbarvil jeden znak uprostřed slova. Je j da! Kolem znaku se vytvořily mezery. K podbarvení byly použity následující příkazy.

```
\usepackage{colorbox}
\newcommand{\seda}[1]{\colorbox{black!20}{#1}}
Je\seda{j}da!
```

Dokumentace k balíčku nabízí jako řešení vynulovat velikost okraje `\colorbox`u. Přidala jsem k tomuto řešení ještě `\strut`, aby barva byla zřetelná také nad a pod písmenem. Jejda!

```
\renewcommand{\seda}[1]{{\fboxsep=0pt
  \colorbox{black!20}{#1\strut}}}
Je\seda{j}da!
```

Přestože se nejedná úplně o problém pro začátečníky, je dobré vědět, že i taková situace může nastat a že je třeba se nebát v takových situacích obrátit na odborníka.

Konce odstavců a vertikální mód

Na konci odstavce TEX přechází z horizontálního do vertikálního módu. Prázdný řádek nebo příkaz `\par` tento přechod zajistí. Je důležité vědět, v jakém módu se aktuálně TEX nachází, protože jsou činnosti, které je lepší provádět ve vertikálním módu. Nejdůležitější z těchto činností je vkládání plovoucích objektů (obrázků, tabulek, algoritmů).

Dále je třeba mít na paměti, že některá nastavení se provádějí až na konci odstavce. Jedním z nich je vertikální vzdálenost účaří, která závisí na velikosti fontu. Příliš mnoho začátečníků se snaží ukončovat odstavce dvěma zpětnými lomítky. Tím se ale odstavec neukončí a vzdálenost účaří se nenastaví. To může vyústit ve výsledek, který

[7] Stejná situace nastává s balíčkem tcolorbox.

je vidět v tomto odstavci.

```
\huge
Dále je třeba mít na paměti, že
...
který je vidět v~tomto odstavci.\\
```

Problém uvedený výše nezpůsobí žádnou chybu ani varování. Řešením je v takovém případě správně ukončit odstavec. Některá prostředí (avšak ne všechna) jsou definována tak, že se na jejich konci odstavec automaticky ukončí.

Vertikální vzdálenost mezi odstavci je určena hodnotou `\parskip`. Tato hodnota bývá definována ve třídě dokumentu, nicméně je možné ji dle potřeby změnit. Někdy je třeba vložit vertikální mezeru ručně. K tomu slouží příkaz `\vspace` nebo `\vskip` použitý ve vertikálním módu (tedy za prázdným řádkem nebo příkazem `\par`).

Dvě zpětná lomítka

Co opravdu neukončuje odstavec, je řídicí znak `\\` (dvě zpětná lomítka). Tento příkaz ukončuje řádek v tabulkách, víceřádkových vzorcích, v básních a v některých dalších situacích. Ale neukončuje odstavec a může vyvolat mnoho chybových zpráv nebo varování.

Jestliže se příkaz `\\` vyskytne ve vertikálním módu, vypíše se chybová zpráva

```
! LaTeX Error: There's no line here to end.
```

Jestliže se před příkazem `\\` vyskytuje vysázená mezera, vytvoří se na výstupu další prázdný řádek, který tam být nemá.

Pokud je řádek končící příkazem `\\` příliš krátký, vypíše se varování

```
Underfull \hbox (badness 10000) in paragraph
          at lines ⟨číslo⟩--⟨číslo⟩
```

Takové varování může být správné, ale je třeba ho zkontrolovat.

Jestliže po `\\` následuje text v hranatých závorkách, `[který se má vysázet]`, dostaneme podivnou chybovou zprávu

```
! Missing number, treated as zero.
```

Příkaz `\\` má totiž nepovinný parametr (ten se dává do hranatých závorek), který udává, jak velká vertikální mezera se má vložit pod příslušným řádkem. Abychom vysázeli příslušný text v hranatých závorkách, musíme před levou závorku vložit příkaz `\relax`.

K ručnímu ukončení řádku odstavce je vhodnější použít příkaz `\newline`.

Změny písma

Změna písma je jednoduchým způsobem, jak v dokumentu zvýraznit důležité pojmy nebo odlišit text se speciálním významem. Mnoho takových změn je přímo zakomponováno do tříd a balíčků. Například nadpisy se sázejí **tučně**, teorémy *kurzívou* a některé časopisy nastavují popisky obrázků bezpatkově, aby je odlišily od okolního textu.

LaTeX nabízí dvě možnosti, jak změnit písmo. První typ příkazů načte svůj argument a pouze ten vysází jiným písmem. Mezi takové příkazy patří `\textbf{...}` pro **tučné písmo**, `\textit{...}` pro *kurzívu* nebo `\textsf{...}` pro bezpatkové písmo. Druhý typ příkazů jsou přepínače, které změní aktuální písmo a tato změna platí až do další změny nebo do konce skupiny. Mezi takové příkazy patří `{\bfseries...}`, `{\itshape...}` nebo `{\sffamily...}`. K těmto příkazům doporučuji přečíst si příslušnou dokumentaci.

Několik příkazů pro změnu písma se chová různě v závislosti na kontextu. Příkaz `\emph{...}` přepne do *kurzívy*, pokud je okolní text antikvou, *a přepne do* antikvy*, pokud je okolní text kurzívou.* V matematickém módu příkaz[8] `\text{...}` vysází text ve stejném písmu, jaké bylo aktuální před začátkem matematického módu. Například uvnitř teorému příkaz `\text{...}` vysází text kurzívou. Když chceme, aby text byl vždy antikvou, použijeme příkaz `\textup`.

Základní TeX definoval pro většinu změn písma dvoupísmenné příkazy. Tyto příkazy jsou druhého uvedeného typu, tj. přepínače. V LaTeXu bychom se těmto dvoupísmenným příkazům měli vyhnout, protože výše uvedené LaTeXové příkazy obsahují některá vylepšení, například provádějí hladší přechod mezi kurzívou a antikvou.

Matematická sazba

Matematická sazba vždy tvoří skupinu. Jestliže matematická sazba začne, musíme ji jasně ukončit. V horizontálním módu matematická sazba začíná a končí znakem `$`. Ke stejnému účelu jsou v LaTeXu definovány také příkazy `\(...\)` a prostředí `math`. Balíčky amsmath a mathtools nabízejí množství prostředí pro matematickou sazbu. Stojí za to se je naučit z příslušné dokumentace.

V matematickém módu (LA)TeX všechny mezery na vstupu ignoruje. Ve vstupním souboru je můžeme použít dle libosti, aby byl soubor dobře čitelný. Pro zvýšení čitelnosti ale v matematickém módu nemůžeme použít prázdný řádek. To způsobí chybu a TeX vypíše zprávu

```
! Missing $ inserted.
```

[8] Příkaz `\text` je definován v balíčku amstext, který se načítá uvnitř balíčku amsmath. (pozn. překl.)

Stejná zpráva se vypíše, pokud matematický mód neukončíme před koncem odstavce nebo pokud v textovém módu použijeme symbol, který je možné použít pouze v matematickém módu.

Jestliže se prázdný řádek vyskytne v některém matematickém prostředí balíčku amsmath, způsobí to chybu a první chybová zpráva bude

```
! Paragraph ended before \⟨prostředí⟩ was complete.
<to be read again>
```

Za touto chybovou zprávou budou následovat další, ale ty už budou docela matoucí. Všechny tyto následující chybové zprávy zmizí, pokud se opraví první chyba, tedy v našem případě odstraní-li se prázdný řádek.

Jestliže skutečně potřebujeme prázdný řádek, abychom zvýšili čitelnost zdrojového souboru, můžeme použít řádek obsahující pouze znak `%`.

Stejně jako u všech prostředí, i tady se musí název prostředí v příkazu `\end` shodovat s názvem prostředí v příkazu `\begin`, a to ve správném pořadí.

Prostředí pro sazbu víceřádkových vzorců by se neměla používat pro jednořádkové vzorce. Pro nečíslované jednořádkové vzorce můžeme použít `$$...$$` nebo LaTeXové příkazy `\[...\]` nebo prostředí `displaymath`.

I když LaTeX nabízí prostředí `eqnarray` pro víceřádkové zarovnané vzorce, nepoužívejte toto prostředí. Pokud je rovnice dlouhá a má své číslo, pak se rovnice a číslo vysázejí přes sebe. Namísto tohoto prostředí používejte raději příslušná prostředí z balíčku amsmath.

Tabulky, obrázky a další plovoucí objekty

Dovolený počet plovoucích objektů na stránce, jejich pozice a mezery mezi nimi jsou určeny ve třídě dokumentu. Když něco nefunguje podle očekávání, pak každý, na koho se obrátíte pro radu, bude chtít vědět, jakou třídu dokumentu používáte.

Ve vstupním souboru musí být plovoucí objekt v místě, kdy je ještě na výstupní stránce pro příslušný objekt dostatek místa. Speciálně při dvousloupcové sazbě to znamená, že prostředí `figure*` a `table*` musejí být ve vstupním souboru dříve než cokoliv jiného na aktuální stránce. Mechanismy LaTeXového jádra neumožňují, aby při dvousloupcové sazbě byly objekty přes celou šířku stránky vysázeny jinde než na horním okraji stránky. Existují balíčky, které tyto mechanismy vylepšují, nicmíně o nich zde psát nebudeme.

Základní třída article nastavuje následující hodnoty:

- maximální počet plovoucích objektů na stránce obsahující text: 3
- maximální počet plovoucích objektů na horním okraji stránky: 2
  maximální pokrytí stránky těmito objekty: 70 %
- maximální počet plovoucích objektů na dolním okraji stránky: 2
  maximální pokrytí stránky těmito objekty: 30 %

- minimální pokrytí stránky textem: 20 %
- minimální pokrytí stránky objekty vysázenými na samostatnou stránku: 50 % výšky sazby[9]

Jestliže je vkládaný objekt malý, má být vložen na přesně dané místo a vejde se tam, pak jej nevkládejte jako plovoucí objekt. Je lepší jej vložit přímo příkazem `\includegraphics` nebo příkazem pro vysázení tabulky, případně můžeme příkaz zapouzdřit do `\begin{center}...\end{center}`.[10]

Balíček wrapfig umožňuje obtékání textu kolem vložených obrázků. Pro detaily si přečtěte dokumentaci k balíčku.

Je zvykem psát popisky nad tabulky, ale pod obrázky. Pokud vložený objekt není plovoucí, nemůžeme použít obvyklý příkaz `\caption`. Namísto něj použijte balíček caption a příkaz `\captionof`.

Třída dokumentu a preambule

Když začínáte psát nový dokument, nejdříve si zvolte třídu dokumentu. Jestliže píšete článek do konkrétního časopisu, přečtěte si pokyny redakce a zjistěte si, jakou třídu a případně další soubory máte použít. Často používané třídy pro různé časopisy jsou k dispozici na CTAN [8].

Jestliže píšete nějaký projekt nebo diplomovou práci, zjistěte si příslušné požadavky; pokud má vaše instituce k dispozici konkrétní šablonu, použijte ji. Pokuste se zjistit, zda se šablona průběžně vyvíjí a jestli je k ní dostupná technická podpora. Přečtěte si dokumentaci.

Třída dokumentu musí definovat základní strukturu a důležité příkazy používané v dokumentu. Jestliže se připravovaný dokument výrazně liší od dokumentů, pro které je určena příslušná třída, je právě teď nejlepší čas obrátit se na někoho zkušenějšího.

Ne všechny součásti jsou ve třídě dokumentu přímo definovány. Například výběr stylu bibliografie může být ponechán na autorovi. Pro tyto situace byly v LaTeXu vytvořeny balíčky.

Příprava dokumentu

Většina balíčků se načítá v preambuli dokumentu, což je ta část zdrojového souboru, která se nachází mezi `\documentclass` a `\begin{document}`. Výjimkou jsou balíčky načítané příkazem `\RequirePackage`, který se obvykle používá před `\documentclass` – může se stát, že na tomto místě bude nutné načíst některá speciální nastavení.

[9] Do výšky sazby se nezapočítávají záhlaví a zápatí.

[10] V plovoucím prostředí pro vycentrování používejte příkaz `\centering`.

Někteří autoři si vytvoří preambuli, která je vhodná pro jeden konkrétní dokument, a poté tuto preambuli používají pro další dokumenty, přičemž přidávají další balíčky a tak dále. A potom přijde nějaký nováček, který si vezme tuto preambuli jako základní šablonu a bez jejího pochopení ji používá. Nedělejte to!

Začněte použitím vhodné třídy dokumentu a další balíčky, volby a definice přidávejte, až když je budete potřebovat. Načítání balíčků si uspořádejte do logických bloků (například všechny fontové balíčky dejte k sobě) a pečlivě si hlídejte, abyste žádný balíček nenačetli více než jednou. Jestliže některý balíček načítáte s volbami, pak jsou při jeho dalším načtení nové volby ignorovány. Některé balíčky interně načítají další balíčky. Například balíček mathtools načítá amsmath a balíček amssymb načítá amsfonts. Důležité je také pořadí balíčků. Balíček hyperref musí být načítaný téměř poslední – těch několik balíčků, které se mohou načítat až po hyperref, je dobře zdokumentováno.

Přečtěte si dokumentaci.

Zpracování dokumentu

Jakmile máme vytvořen zdrojový soubor, je na čase jej zpracovat (LA)TEXem a vygenerovat soubor výstupní. Je několik programů, ze kterých si podle potřeby můžeme vybrat, například pdfLATEX, XELATEXnebo LuaLATEX. Tyto programy můžeme spustit z příkazového řádku nebo prostřednictvím textového editoru. Někdy musíme zdrojový soubor zpracovat (LA)TEXem vícekrát. Kolikrát, to závisí na tom, jak je nutné přesouvat různé informace a jak se tyto informace mění.

Informace o křížových odkazech se zapisují do souboru .aux, informace o sekcích se zapisují do souboru .toc. Dokument může obsahovat i další seznamy. Seznam literatury se někdy zpracovává dalším programem (v takovém případě je potřebné zkontrolovat, jestli externí program nenahlásil chybu) a přeformátovaná data o jednotlivých záznamech jsou uložena do dalšího souboru. Potom je nutné LATEX spustit ještě minimálně dvakrát – jednou pro načtení souboru .aux a dalších vedlejších souborů (pro správné nastavení křížových odkazů) a podruhé pro správné nastavení čísel stran (jelikož například vložení obsahu nebo seznamu literatury v předchozím spuštění LATEXu mohlo text dokumentu posunout).

Předchozí odstavce předpokládají, že ve zdrojovém souboru nejsou chyby. Případné chyby jsou zaznamenány v souboru .log. Zjistěte si, kde se tento soubor nachází, a zvykněte si jej kontrolovat. Například varování o znacích chybějících v nějakém fontu se za běhu (LA)TEXu nezobrazí a jsou pouze v souboru .log.

`Missing character: There is no` ⟨*znak*⟩ `in font` ⟨*název*⟩`!`

Někdy soubor .log obsahuje skupinu chyb majících čísla řádku blízko sebe. V takovém případě první číslo řádku je to, na kterém (LA)TEX narazil na problém, zatímco na dalších uvedených řádcích ve skutečnosti chyba vůbec nemusí být.

Opravte první chybu a spusťte (LA)TEX znovu. Často se stane, že původní další chyby už se znovu neobjeví.

Přeji hodně štěstí. Časem se to naučíte.

A ještě... Nezapomeňte si přečíst dokumentaci.

Poděkování

Autorka děkuje samcarter, Mikaelovi Sundquistovi a (jako vždy) Karlu Berrymu za nápady a za nalezení a odstranění jejích překlepů. Dokážu překlepy najít u cizích článků, ale ne u svých.

Odkazy

1. BEETON, Barbara. What every (LA)TEX newbie should know. *TUGboat*. 2023, **44**(2), 164–169.
2. *TEX StackExchange* [online]. [cit. 2023-11-13]. Dostupné z: `https://tex.stackexchange.com`.
3. SCHARRER, Martin. *Often referenced questions* [online]. 2023-05-09. [cit. 2023-11-13]. Dostupné z: `https://tex.meta.stackexchange.com/q/2419`.
4. BEETON, Barbara. Debugging LATEX files – Illegitimi non carborundum. *TUGboat*. 2017, **38**(2), 159–164.
5. BEETON, Barbara. Ladění LATEXových souborů. *Zpravodaj CSTUGu*. 2021, **31**(1), 63–75.
6. OLŠÁK, Petr. *TEXbook naruby*. 2. vyd. Konvoj, 2001. Dostupné také z: `http://petr.olsak.net/ftp/olsak/tbn/tbn.pdf`.
7. MITTELBACH, Frank; FISCHER, Ulrike. *The LATEX Companion*. 3. vyd. Addison-Wesley Professional, 2023.
8. *CTAN: Search* [online]. [cit. 2023-11-13]. Dostupné z: `https://ctan.org/search`.

Summary: What Every (LA)TEX Newbie Should Know

LATEX has a reputation for producing excellent results, but at the cost of a steep learning curve. That's true, but by understanding a few basic principles, and learning how to avoid some techniques that may seem obvious but often lead one into the weeds, it's possible to avoid some of that pain. Our goal here is to encourage good habits before bad habits have had a chance to develop.

Keywords: LATEX, newbie, errors, log file

Barbara Beeton, TUGboat, Providence, RI, USA, `bnb@tug.org`