První setkání s TeXem

Petr Olšák

Autor programu TeX je profesor Donald Knuth.

TeX je ochranná známka American Mathematical Society.

Ostatní v manuálu použité názvy programových produktů, firem apod. mohou být ochrannými známkami nebo registrovanými ochrannými známkami příslušných vlastníků.

Úvod

Tento manuál je koncipován jako „první seznámení s programem TEX[1] na jeden večer". Měl by umožnit začátečníkovi porozumět základním principům TEXu. Manuál obsahuje ukázku jednoduchého dokumentu, který by si měl čtenář sám přepsat do svého počítače a na něm TEX vyzkoušet. Je to dobrý první krok do pestrého světa tohoto programu plného zajímavých možností. Předvedená ukázka mimo jiné ilustruje základní principy psaní dokumentů v TEXu. Jsou zde předvedeny dvě úvodní ukázky: pro plainTEX a pro LATEX. Takže si uživatel může hned na prvním dokumentu rozhodnout, které TEXové rozšíření je bližší jeho srdci a podle toho vybrat další literaturu pro doplňující studium.

Předpokládáme, že čtenář má určité důvody proč použít TEX, takže se zde nebudeme zdržovat výčtem jeho výhod, rozepisovat obšírně jeho historii a nebudeme polemizovat o užitečnosti či neužitečnosti dávkového či interaktivního systému na přípravu sazby.

TEX a jeho okolí

TEX je formátor. Je to program, kterému předložíme vstupní text dokumentu v „holé" textové podobě doplněný textovými značkami, které vymezují strukturu dokumentu nebo dávají TEXu pokyny o způsobu formátování dokumentu. Bývá obvyklé (ale není to nutné) pojmenovat tento soubor s použitím přípony `.tex`, například `dokument.tex`. Na výstupu pak po zpracování TEXem dostaneme PDF soubor (`dokument.pdf`). Dříve se též hojně používal výstup do formátu DVI[2].

TEX tedy čte na svém vstupu textový soubor s dobře definovanou syntaxí jazyka značek a na výstupu je soubor s definitivním popisem sazby. TEX jako takový je zcela nezávislý na operačním systému. Vývoj samotného TEXu je zastaven, takže pro uživatele nehrozí nebezpečí vzniku dalších nekompatibilních verzí. Další programy „okolo TEXu" tvoří společně s TEXem distribuci. Dnes se nejčastěji používají volně přístupné distribuce TEXlive nebo MikTEX.

Začínající uživatel se samozřejmě hlavně ptá po způsobu, jak může v konkrétním operačním systému s konkrétní TEXovou distribucí s tímto programem pracovat, jak jej spustit, jakými tlačítky se ovládá textový editor, jaké nabídky jsou k dispozici, co nad kterým obrázkem udělá myš. Ptá se tedy po uživatelském rozhraní. Jednotlivé manuály o TEXu tradičně odkazují na tzv. „místní příručku" (Local Guide), která by měla toto rozhraní popisovat. Tato příručka je závislá na použitém operačním systému, na čase jejího vzniku, na použité distribuci TEXu, na vybraném textovém editoru a někdy též na administrátorovi systému, který konfiguruje některé věci specificky pro větší pohodlí uživatelů. Texty o TEXu uživatelské prosředí většinou neuvádějí (je totiž závislé na okolnostech) a popisují pouze na systému nezávislé vlastnosti TEXu jako formátoru. Ani tento manuál není v tomto ohledu výjimkou.

Při práci s TEXem je obvyklé mít otevřen v jednom okénku textový editor, ve kterém uživatel píše nebo modifikuje vstupní text, a ve vedlejším okénku prohlížeč výstupního souboru. Po modifikaci vstupního textu uživatel spustí TEX na pozadí klávesovou zkratkou a ve vedlejším okénku vidí během pár sekund výslednou změnu v sazbě.

[1] Název TEX se čte „tech", nikoli „teks".

[2] DVI: Odvozeno z anglického „device independent" – na zařízení nezávislý. Soubor lze prostřednictvím vhodného programu prohlédnout na obrazovce, nebo jej vytisknout na tiskárně.

Textový editor, ve kterém připravujeme nebo modifikujeme vstupní texty dokumentů, nesmí ukládat na disk žádné skryté formátovací informace implementované jen pro tento editor (jako například změna fontu, měkké konce řádku apod.). To dělají tzv. textové procesory, které v případě práce s TEXem nepoužíváme.

Zvyklosti ve značkování dokumentu jsou vesměs závislé na použitém *formátu* TEXu, který modifikuje jeho chování. Říkáme, že je dokument napsán ve formátu *LATEX*, pokud je někde na začátku vstupního textu dokumentu uvedena značka `\documentclass` (nebo dříve `\documentstyle`). Pokud tam tuto značku nenajdeme, můžeme předpokládat, že je dokument napsán ve formátu *plain*. Ten umožňuje psát jen anglické texty. V čechách nebo na slovensku se místo formátu plain používá *csplain*. Tento manuál je například napsán ve formátu csplain a je uložen v souboru `prvni.tex`. Může se stát, že nějaký dokument je napsán ještě v jiném formátu méně používaném formátu. Tím se ale zde nebudeme zabývat. Rozdíl mezi plainem a LATEXem a smysl použití formátů vyplyne až z dalšího textu.

Následující tabulka ukazuje způsoby spuštění TEXu. Předpokládáme, že je k dispozici operační systém, který umožňuje uživateli zadávat pokyny z příkazového řádku. Tím nevylučujeme, že nelze některé popisované činnosti implementovat do nějaké uživatelské nabídky konkrétního uživatelského rozhraní. Předpokládejme, že je vstupní text dokumentu připraven v souboru `dokument.tex`.

příkazový řádek	komentář
`tex dokument`	anglický dokument, formát plain
`pdfcsplain dokument`	formát csplain, výstup do PDF
`csplain dokument`	formát csplain, výstup do DVI
`pdflatex dokument`	formát LATEX, výstup do PDF
`latex dokument`	formát LATEX, výstup do DVI

Všimneme si, že v příkazovém řádku píšeme za jméno formátu název vstupního souboru a že příponu `.tex` nemusíme psát. Dobře instalovaná distribuce TEXu by měla podle jména formátu spustit TEX modifikovaný právě tímto formátem.

Pokud nemáme ve své distribuci TEXu formát csplain, je to špatné znamení. Nebudeme totiž schopni zpracovat ani tento manuál ani ukázku, která je v něm obsažena. V takovém případě lze doporučit poohlédnout se www stránce csplainu, kterou najdeme například na `http://petr.olsak.net/csplain.html`.

Jdeme na to

Nejprve zkusíme v nějakém textovém editoru vytvořit soubor `pokus.tex`[3], který obsahuje zkušební větu:

```
Ahoj světe!
\bye
```

Pokud zpracujeme tento soubor TEXem s formátem csplain (připomínáme, že je možné použít povel `pdfcsplain pokus`), dostaneme výstupní soubor `pokus.pdf`. Navíc TEX uloží informaci o zpracování do souboru `pokus.log`. Výsledný PDF soubor si můžeme prohlédnout vhodným prohlížečem. Dostaneme očekávaný výsledek:

[3] Soubor musí být kódovaný při použití cslainu z roku 2013 a novějšího v UTF-8.

```
Ahoj světe!
```

Přitom dole na stránce je ještě vytištěno číslo strany: 1.

Pokud zkusíme tentýž soubor zpracovat TEXem s formátem plain (příkazový řádek `tex pokus`), výstup bude zmršený: „Ahoj svte!“. Vidíme, že textový soubor s akcenty nelze jednoduše vnutit originálnímu americkému TEXu, ale místo formátu plain je potřeba použít modifikovaný csplain.

Kdybychom chtěli tentýž soubor zpracovat TEXem s formátem LATEX (krátce říkáme, že soubor zpracováváme LATEXem), obdržíme chybové hlášení:

```
! LaTeX Error: Missing \begin{document}.

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

l.1 A
     hoj světe!
?
```

Vidíme tedy, že soubor není vhodně připraven ke zpracování LATEXem. Chybí mu `\begin{document}`. Později ukážeme, že mu chybí více věcí, ale v tuto chvíli raději zůstaneme u csplainu. Komunikaci s TEXem při chybovém hlášení ukončíme odesláním znaku `x`. Chceme-li, aby TEX chybu ignoroval a pokračoval ve zpracování dokumentu, stačí na otazník odpovědět „Enter“ (v tomto příkladě, kdy dokument vhodný pro csplain chceme zpracovat LATEXem, se pouze dočkáme další zavlečené chyby).

Zkusíme si nyní přepsat do počítače následující poněkud rozsáhlejší dokument. Soubor nazveme třeba `mujprvni.tex` a vytvoříme jej libovolným textovým editorem. Pokud je čtenář od přírody „lenivý“, může se místo zdlouhavého přepisování pokusit najít text ukázky ve vstupním souboru `prvni.tex` tohoto manuálu a přenést jej do svého souboru `mujprvni.tex` jako blok v textovém editoru.

Možná nám může připadat část označená jako „oblast definic“ hodně nepochopitelná, skoro jako porucha na lince. Přesto se zatím pokusíme překonat odpor k této poruše a důsledně všechny znaky přepíšeme. TEX se nám za to odvděčí silnými možnostmi, které budeme postupně odhalovat.

Nemusíme se obtěžovat přepisováním textů, schovaných za znakem %, protože tímto znakem je zahájen komentář, který končí koncem řádku a který je při zpracování TEXem ignorován. V každém případě ale nevynechávejme prázdné řádky v ukázce a věnujme pozornost obsahu části označené jako „vlastní text“, kde jsou vyjmenovány základní jevy, se kterými se při pořizování textů pro TEX budeme často setkávat.

```
%%%%%%% Zde začíná "oblast definic" pro tento dokument %%%%%%%%%

\chyph       % inicializace českého dělení slov v csplainu
\font\titulfont=\fontname\tenbf\space scaled \magstep2  % větší font
\def\bod{\item{$\bullet$}}  % definice zkratky \bod pro výčet
\def\nadpis#1\par{          % definice nadpisu:
   \removelastskip\bigskip  %   odmaže poslední vert. mezeru a přidá vlastní
   \indent{\titulfont #1}   %   odsazený text nadpisu větším fontem
   \par\nobreak\medskip}    %   konec řádku, zakázaný zlom, menší mezera
\let\itemskip=\medskip      % kolem výčtu prvků bude menší mezera \medskip

%%%%%%% Zde začíná "vlastní text" dokumentu %%%%%%%%%%%%%%%%%%%%

\nadpis Můj první dokument

Zkouším napsat první text v~\TeX u. Tento odstavec musí být
tak dlouhý, aby bylo vidět, že se rozlomil aspoň na dva řádky.

Jednotlivé odstavce oddělujeme od sebe prázdným řádkem. Prázdnými řádky
vůbec nešetříme, protože zvyšují přehlednost zdrojového textu.
Vyzkoušíme si nyní několik věcí.

\itemskip
\bod Budeme používat české \uv{uvozovky}, které se liší od ‘‘anglických’’.
     Uvědomíme si, že použití "těchto znaků" je úplně špatně!
\bod Rozlišujme mezi spojovníkem (je-li), pomlčkou ve větě~--
     a dlouhou pomlčkou---ta se používá v~anglických dokumentech.
\bod Předpokládáme, že každý dokáže rozeznat 1 (jedničku) od l
     (písmene el) a 0 (nulu) od O (písmene~O).
\bod Zkusíme přepnout do {\bf polotučného písma}, nebo do
     {\it kurzívy}. Také vyzkoušíme {\tt strojopis}.
\bod Všimneme si, že ve slovech grafika, firma, apod. se písmena
     f a i automaticky proměnila v~jediný znak fi (srovnáme to
     s~nesprávným f\/i).
\bod Mezery mezi písmeny jsou automaticky vyrovnávány podle tvaru písmen.
     Ve slově \uv{Tento} je například písmeno~e těsněji přisazeno
     k~písmenu~T, aby se mezery mezi písmeny opticky jevily stejnoměrné.
\bod Vypravíme se na malou exkurzi do matematiky: $a^2 + b^2 = c^2$.
     Zjistíme, že číslo -1 je zde napsáno špatně (prokletý spojovník),
     zatímco správně má být $-1$.
\bod Protože \% uvozuje komentář a \$ přepíná do matematické sazby,
     musíme před ně napsat zpětné lomítko, chceme-li je dostat do dokumentu.
\itemskip

\nadpis Závěr

To by pro začátek stačilo. Příkazem {\tt\char‘\\bye} ukončíme své pokusy.
\bye
```

Po zpracování tohoto dokumentu formátem csplain si můžeme prohlédnout prohlížečem dvi souboru následující výsledek:

Můj první dokument

Zkouším napsat první text v TEXu. Tento odstavec musí být tak dlouhý, aby bylo vidět, že se rozlomil aspoň na dva řádky.

Jednotlivé odstavce oddělujeme od sebe prázdným řádkem. Prázdnými řádky vůbec nešetříme, protože zvyšují přehlednost zdrojového textu. Vyzkoušíme si nyní několik věcí.

- Budeme používat české „uvozovky“, které se liší od “anglických”. Uvědomíme si, že použití ”těchto znaků” je úplně špatně!
- Rozlišujme mezi spojovníkem (je-li), pomlčkou ve větě – a dlouhou pomlčkou—ta se používá v anglických dokumentech.
- Předpokládáme, že každý dokáže rozeznat 1 (jedničku) od l (písmene el) a 0 (nulu) od O (písmene O).
- Zkusíme přepnout do **polotučného písma**, nebo do *kurzívy*. Také vyzkoušíme `strojopis`.
- Všimneme si, že ve slovech grafika, firma, apod. se písmena f a i automaticky proměnila v jediný znak fi (srovnáme to s nesprávným fi).
- Mezery mezi písmeny jsou automaticky vyrovnávány podle tvaru písmen. Ve slově „Tento“ je například písmeno e těsněji přisazeno k písmenu T, aby se mezery mezi písmeny opticky jevily stejnoměrné.
- Vypravíme se na malou exkurzi do matematiky: $a^2 + b^2 = c^2$. Zjistíme, že číslo -1 je zde napsáno špatně (prokletý spojovník), zatímco správně má být -1.
- Protože % uvozuje komentář a $ přepíná do matematické sazby, musíme před ně napsat zpětné lomítko, chceme-li je dostat do dokumentu.

Závěr

To by pro začátek stačilo. Příkazem `\bye` ukončíme své pokusy.

Všimněme si, že v příkladu je důsledně oddělena forma od obsahu dokumentu. V části označené „vlastní text“ jsou použity značky `\nadpis`, `\bod` a `\itemskip`, které ohraničují logické části dokumentu (vymezení nadpisu, uvedení další položky ve výčtu prvků, obklopení skupiny výčtu prvků) a nepopisují žádné konkrétní formátovací informace (volba fontu, velikost fontu, velikost mezer nad a pod nadpisem, tvar puntíku ve výčtu prvků apod.).

Značky vymezující strukturu dokumentu jsou definovány v části „oblast definic“. Zde je řečeno, jaký bude mít nadpis font, jak bude v textu umístěn a jak bude vypadat formátování výčtu prvků. Podrobnější rozbor těchto definic uvedeme za chvíli.

Toto oddělení formy od obsahu se v mnoha případech začátečníkům nedaří. Přímo v textu jejich dokumentů se vyskytují značky jako `\vskip12mm` (vertikální mezera 12 mm), `\vfill\break` (vynucené ukončení strany) a mnoho dalších nešvarů. My se pokusíme hned z počátku se takovým věcem pokud možno vyhnout. V sekci „změna vzhledu dokumentu“ uvidíme, že se nám to bohatě vyplatí.

Vysvětlení použitých značek v příkladu

Jednotlivé značky, které řídí formátování a vymezují strukturu dokumentu jsou vesměs ve tvaru `\slovo`. Tyto značky se nazývají *řídicí sekvence* a někdy též budeme hovořit o *příkazech*, protože jimi přikazujeme, aby TEX něco vykonal.

Řídicí sekvence v příkladu rozdělíme na dvě skupiny. 1. ty, co jsou definovány v samotném TEXu nebo v použitém formátu (v našem případě ve formátu `csplain`). 2. řídicí sekvence, které jsme definovali sami.

Začneme rozborem druhé skupiny řídicích sekvencí:

- ▹ `\nadpis` je sekvence, která za sebou očekává text nadpisu a pak prázdný řádek.

- `\titulfont` je přepínač pro větší velikost fontu pro nadpis. Je definován na řádku začínajícím příkazem `\font` a použit v definici řídicí sekvence `\nadpis`.
- `\bod` je sekvence, která uvozuje položku ve výčtu prvků. Promění se v nějakou grafickou realizaci zarážky (zde puntík) a způsobí odsazení textu položky.
- `\itemskip` vytvoří vertikální mezeru, která oddělí výčet prvků od ostatního textu. Použije se na začátku i na konci výčtu.

Ostatní řídicí sekvence jsou definovány v použitém formátu nebo přímo zabudovány v TEXu. Uživatel se s nimi bude postupně seznamovat studiem vhodné literatury. Zde uvedeme velmi stručně jen ty nejdůležitější řídicí sekvence, abychom usnadnili pochopení příkladu.

- `\chyph`. Tato řídicí sekvence inicializuje české vzory dělení slov a je definována pouze ve formátu csplain. V případě českých textů bychom ji nikdy neměli vynechat! Analogicky `\shyph` inicializuje slovenské vzory dělení slov. Bez těchto příkazů TEX pracuje implicitně s anglickými vzory dělení.
- `\font` zavede z instalace TEXu do dokumentu další font. Struktura parametrů příkazu bude vysvětlena v sekci o fontech.
- `\def` definuje novou řídicí sekvenci (zde `\bod` a `\nadpis`). Za řídicí sekvencí může následovat formální popis parametrů nové sekvence a pak následuje ve složených závorkách tělo definice. V ní je popsáno, co se při použití nové řídicí sekvence má vykonat.
- `\item` zahájí výčtovou položku (odsazením textu) a převezme za sebou ve složených závorkách parametr, který popisuje vzhled puntíku.
- `\bullet` vytvoří v matematickém módu černý puntík: $\bullet$.
- `\bigskip` vytvoří vertikální mezeru velikosti jednoho řádku a `\medskip` velikosti poloviny řádku. `\indent` odsadí další text o velikost odstavcové zarážky.
- Řídicí sekvence `\par` je explicitní ukončení odstavce. TEX ji interně vytváří v místě každého prázdného řádku. Ve formálním popisu parametru za `\def\nadpis` má ale `\par` pouze vymezovací účinek. Formální popis parametru v našem příkladě čteme takto: nově definovaná řídicí sekvence `\nadpis` převezme za sebou text až po první výskyt `\par` (tedy až po první výskyt prázdného řádku) a uloží jej do „proměnné“ s označením `#1`.
- `\let` čteme česky nechť. Tento příkaz ztotožní význam nové řídicí sekvence (v našem příkladě `\itemskip`) s předlohou (v tomto příkladě `\medskip`).

V naší ukázce jsme použili též některé speciální TEXovské znaky. Vysvětlíme si nyní stručně jejich význam.

- Znak `~` znamená nedělitelnou mezeru. Je zde použita za neslabičnými předložkami a před jednopísmennými ukázkami, aby se v těchto místech nerozdělil řádek. Při pořizování textu nemusíme psát za neslabičnými předložkami vlnku „ručně“. Vlnky tam lze doplnit později jednoduchými programy, které bývají součástí TEXovských instalací.
- Znaky `{ }` mají v TEXu tři mírně odlišné významy.
  1. Obklopují těla definic za příkazem `\def`, jak již bylo řečeno.
  2. Obklopují parametry některých řídicích sekvencí (viz například text `uvozovky`, který je parametrem řídicí sekvence `\uv`, nebo text `$\bullet$`, který je parametrem řídicí sekvence `\item`.

3. Samotné znaky `{ }` vymezují jisté skupiny, ve kterých je veškeré přiřazení a nastavení lokální. Skupiny se často používají pro vymezení platnosti přepínačů písma (viz `\bf`, `\it`, `\tt` a `\titulfont`).

Závorky `{ }` musí vzájemně párovat, což je důležité zejména ve vymezovacích významech (ad 1 a 2). Proto třeba tělo definice `\bod` obsahuje text `\item{$\bullet$}` a je ukončeno až druhou závorkou `}`.

- Znak `%` uvozuje komentář až do konce řádku.
- Znak `$` přepíná do *matematického módu* a zpět. V matematickém módu TeX sestavuje sazbu poněkud odlišným způsobem (všimneme si, že například proměnné a, b jsou v matematickém módu automaticky sázeny kurzívou).
- Znak `^` v matematickém módu uvozuje horní index (exponent).

Změna vzhledu dokumentu

Předvedeme, v čem spočívá výhoda oddělení obsahu dokumentu od formy. Předpokládejme, že nám nějaký zkušenější kolega pomůže s přípravou definic pro náš dokument. Předpokládejme dále, že onen kolega má na věc poněkud jiný typografický názor a začne věci předělávat. V editoru modifikuje definice a ve vedlejším okénku v prohlížeči se průběžně mění náš první dokument skoro k nepoznání. Přitom kolega *vůbec nemusí* zasáhnout do vlastního textu dokumentu.

Především se mu nelíbí rodina fontů Computer Modern, která je v TeXu implicitně nastavena. Napíše tedy na začátek dokumentu třeba `\input cbookman` a celý dokument je nyní v rodině Bookman. Příkaz `\input` zavádí do dokumentu externí soubor definic, zde soubor s názvem `cbookman.tex`. Tento soubor obsahuje příkazy `\font` na zavedení skupiny fontů Bookman a nastaví je jako implicitní.

Kolega se dále rozhodl vkládat mezi každý odstavec drobnou vertikální mezeru a místo puntíků pro výčty chce použít čtverečky, které ve větší velikosti zařadí i do nadpisů. Konečně velikost fontu pro nadpis se mu zdá příliš velká (místo `\magstep2` v řádku `\font` použije „menší“ `\magstep1`). Výsledek jeho snažení v „oblasti definic“ dopadne třeba takto:

```
\chyph             % inicializace českého dělení slov v csplainu
\magnification\magstep1     % celý dokument bude 1,2 krát větší
\input cbookman \setsimplemath  % použité písmo: Bookman i v matematice
\font\titulfont=\fontname\tenbf\space scaled \magstep1  % větší font
\newdimen\indskip \indskip=15pt % výčty budou odsazeny 15pt
\def\ctverecek#1{\noindent  % čtvereček proměnné velikosti v místě \indskip
   \hbox to\indskip{\vrule height#1pt depth0pt width#1pt\hss}}
\def\bod{\par\hangindent=\indskip \ctverecek{4}}  % definice zkratky \bod
\def\nadpis#1\par{          % definice nadpisu:
   \removelastskip\bigskip  %   odmaže poslední vert.mezeru a přidá vlastní
   \ctverecek{7}{\titulfont #1} % nadpis odsazený čtverečkem
   \par\nobreak}            %   konec řádku, zakázaný zlom, žádná mezera
\parskip=\medskipamount     % mezi odstavci bude mezera jako \medskip
\parindent=0pt              % odstavce nebudou odsazeny zarážkou
\let\itemskip=\relax        % žádné další mezery mezi výčty
```

Náš dokument vypadá pak následovně:

■ Můj první dokument

Zkouším napsat první text v TEXu. Tento odstavec musí být tak dlouhý, aby bylo vidět, že se rozlomil aspoň na dva řádky.

Jednotlivé odstavce oddělujeme od sebe prázdným řádkem. Prázdnými řádky vůbec nešetříme, protože zvyšují přehlednost zdrojového textu. Vyzkoušíme si nyní několik věcí.

- Budeme používat české „uvozovky“, které se liší od “anglických”. Uvědomíme si, že použití ”těchto znaků” je úplně špatně!
- Rozlišujme mezi spojovníkem (je-li), pomlčkou ve větě – a dlouhou pomlčkou—ta se používá v anglických dokumentech.
- Předpokládáme, že každý dokáže rozeznat 1 (jedničku) od l (písmene el) a 0 (nulu) od O (písmene O).
- Zkusíme přepnout do **polotučného písma**, nebo do *kurzívy*. Také vyzkoušíme `strojopis`.
- Všimneme si, že ve slovech grafika, firma, apod. se písmena f a i automaticky proměnila v jediný znak fi (srovnáme to s nesprávným fi).
- Mezery mezi písmeny jsou automaticky vyrovnávány podle tvaru písmen. Ve slově „Tento“ je například písmeno e těsněji přisazeno k písmenu T, aby se mezery mezi písmeny opticky jevily stejnoměrné.
- Vypravíme se na malou exkurzi do matematiky: $a^2 + b^2 = c^2$. Zjistíme, že číslo -1 je zde napsáno špatně (prokletý spojovník), zatímco správně má být -1.
- Protože % uvozuje komentář a $ přepíná do matematické sazby, musíme před ně napsat zpětné lomítko, chceme-li je dostat do dokumentu.

■ Závěr

To by pro začátek stačilo. Příkazem `\bye` ukončíme své pokusy.

Kdyby náš kolega chtěl, implementoval by třeba automatické číslování položek, automatické číslování nadpisů, generování obsahu a další věci. Vysvětlení nových řídicích sekvencí, které kolega použil, bohužel překračuje rámec tohoto úvodního dokumentu.

Definice lze umístit do jiného souboru než vlastní text dokumentu. Na začátku dokumentu pak soubor definic načteme příkazem `\input`. Nebo naopak, hlavní bude soubor definic, ze kterého se příkazem `\input` postupně načítají jednotlivé kapitoly rozsáhlejšího díla.

Stojíme na křižovatce

V předchozím příkladě jsme ilustrovali důležitou vlastnost TEXu – schopnost měnit vzhled dokumentu jen výměnou některých definic. Kromě toho ale tyto definice také musejí navazovat na úmluvu, jakými značkami bude autor vymezovat strukturu svého dokumentu. Kdyby autor použil místo značky `\nadpis` značku `\section`, TEX by nám při zpracování dokumentu vynadal:

```
! Undefined control sequence.
l.14 \section
              Můj první dokument
?
```

tedy: nedefinovaná řídicí sekvence. Odpovíme-li na otazník pouhým stiskem klávesy Enter, TEX tuto sekvenci zcela ignoruje a pracuje dál. Žádného zvýraznění nadpisu bychom se nedočkali. Je tedy vidět, že je podstatné ujasnit si, jaké značkování struktury dokumentu použijeme.

V této souvislosti si musíme odpovědět jednu důležitou otázku. Chceme se naučit jazyk definic TEXu na takové úrovni, jako náš imaginární kolega z předchozího příkladu? Budeme raději sami kontrolovat každý detail vzhledu dokumentu prostřednictvím vlastních definic, než abychom přebírali hotová řešení odjinud? Pokud na tyto otázky odpovíme „ano“, pak je pro nás výhodné použít formát plain (pro české a slovenské dokumenty jen mírně modifikovaný formát csplain), který definuje jen minimum základních řídicích sekvencí. O další řídicí sekvence stejně jako o modifikaci vzhledu dokumentu podle našich představ se musíme postarat sami.[4] V takovém případě si můžeme sami rozhodnout, jaké značky pro vymezení struktury dokumentu použijeme, protože si pro ně nakonec uděláme vlastní definice.

Na druhé straně, pokud rádi přebíráme hotová řešení, pokud nechceme zbytečně pronikat do problematiky jazyka definic TEXu, pokud se spokojíme s už připravenými šablonami vzhledu dokumentu (tzv. styly), pokud jsme ochotni se místo tří set základních příkazů TEXu učit zhruba tisíc uživatelských značek pro LATEX, bude pro nás výhodné použít raději formát LATEX.

LATEX doporučuje určité značkování struktury dokumentu. Například se předpokládá členění na kapitoly (značka `\chapter`) a na sekce (značka `\section`). Každý dokument by měl začít záhlavím uvozeným sekvencí `\documentclass`. Parametrem této sekvence by měl být název základního stylového souboru (souboru definic upravujících vzhled dokumentu). Nejčastěji bývá tímto parametrem `book` (formát knihy) nebo `article` (formát článku). Doplňkové stylové soubory se načítají pomocí sekvence `\usepackage`. Vlastní text dokumentu musí být uzavřen mezi značkami `\begin{document}` a `\end{document}`. Často se vyskytují další značky `\begin` a `\end` vymezující v LATEXu jistá prostředí (například prostředí pro výčtové položky).

Vraťme se k našemu příkladu a přepišme jej do značkování podle LATEXu.

[4] Po zveřejnění makra OPmac (`http://petr.olsak.net/opmac.html`) uvedená věta o nutnosti plainTEXisty vše si programovat vlastními silami není zcela pravdivá. I tito uživatelé mohou použít hotové řešení OPmac. Navíc mají možnost snadno do jeho maker nahlédnout a upravit si je k obrazu svému.

```
\documentclass{article}       % základní styl bude "odborný článek"
\usepackage[utf8]{inputenc}   % je nutno specifikovat kódování dokumentu
\usepackage[T1]{fontenc}      % požadujeme LaTeXovské fonty s akcenty
\usepackage[czech]{babel}     % z Babylónu jazyků volíme češtinu
\begin{document}

\section{Můj první dokument}

Zkouším napsat první text v~\TeX u. Tento odstavec musí být
tak dlouhý, aby bylo vidět, že se rozlomil aspoň na dva řádky.

Jednotlivé odstavce oddělujeme od sebe prázdným řádkem. Prázdnými řádky
vůbec nešetříme, protože zvyšují přehlednost zdrojového textu.
Vyzkoušíme si nyní několik věcí.

\begin{itemize}
\item Budeme používat české \uv{uvozovky}, které se liší od ‘‘anglických’’.
      Uvědomíme si, že použití "těchto znaků" je úplně špatně!
\item Rozlišujme mezi spojovníkem (je-li), pomlčkou ve větě~--
      a dlouhou pomlčkou---ta se používá v~anglických dokumentech.
\item Předpokládáme, že každý dokáže rozeznat 1 (jedničku) od l
      (písmene el) a 0 (nulu) od O~(písmene~O).
\item Zkusíme přepnout do {\bf polotučného písma}, nebo do
      {\it kurzívy}. Také vyzkoušíme {\tt strojopis}.
\item Všimneme si, že ve slovech grafika, firma, apod. se písmena
      f a i automaticky proměnila v~jediný znak fi (srovnáme to
      s~nesprávným f\/i).
\item Mezery mezi písmeny jsou automaticky vyrovnávány podle tvaru písmen.
      Ve slově \uv{Tento} je například písmeno~e těsněji přisazeno
      k~písmenu~T, aby se mezery mezi písmeny opticky jevily stejnoměrné.
\item Vypravíme se na malou exkurzi do matematiky: $a^2 + b^2 = c^2$.
      Zjistíme, že číslo -1 je zde napsáno špatně (prokletý spojovník),
      zatímco správně má být $-1$.
\item Protože \% uvozuje komentář a \$ přepíná do matematické sazby,
      musíme před ně napsat zpětné lomítko, chceme-li je dostat do dokumentu.
\end{itemize}

\section{Závěr}

To by pro začátek stačilo. Příkazem \verb|\bye| ukončíme své pokusy.
\end{document}
```

Upozorňujeme, že v současné době je bohužel LaTeXů několik druhů. „Starý LaTeX“ (verze 2.09), ve kterém se struktura záhlaví dokumentu mírně lišila (byl použit příkaz `\documentstyle` místo `\documentclass`). Dále se pro potřeby českých dokumentů dlouho používal místo LaTeXu tzv. CSLaTeX, který se vyhýbal balíčku `babel`. Ovšem CSLaTeX je dnes také považován zastaralý. Uvedená ukázka zahrnuje klasický LaTeX, často označovaný jako verze 2e.

Na druhé straně na novějších modifikacích TeXu (LuaTeX a XeTeX) je vybudována novější verze LaTeXu zvaná LuaLaTeX a XeLaTeX. V této novější verzi se nepoužívají balíčky `inputenc`, `fontenc` ani `babel`, ale místo toho balíčky `fontspec` a `polyglossia`. O tom ale tento úvodní text nepojednává.

Výsledek po zpracování našeho dokumentu LaTeXem v tomto manuálu pro stručnost neuvádíme. Kdo chce, může si sám LaTeX vyzkoušet. Nelíbí se nám, že za čísly v nadpisech nejsou tečky? Zavedeme do dokumentu prostřednictvím `\usepackage` další doplňkový styl, který toto výchozí chování základního stylu `article` upraví. Nelíbí se nám, že jsou mezi jednotlivými položkami ve výčtu velké mezery a položky jsou až příliš odsazeny? Použijeme v dokumentu další doplňkový styl. Nelíbí se nám, že je použito písmo Computer Modern? Napišme třeba `\usepackage{times}`.

Otázka ale je, kdo pro nás tyto doplňkové styly (neboli doplňující sady definic pro TeX) bude připravovat. Velké množství stylů na všechno možné lze nalézt ve veřejných archivech TeXovského softwaru. LaTeX nám tedy při jednoduchých šablonovitých požadavcích na vzhled dokumentu umožňuje zůstat v roli autora, který pořizuje text. Nemusíme umět poměrně složitý jazyk definic TeXu.

Pokud nám žádná z možností nabízených stylů nevyhovuje, musíme se pokusit tyto styly modifikovat podle své potřeby. To ale může být už hodně komplikované. Záleží znovu na nás, zda rádi modifikujeme zdrojové kódy cizích programů nebo si raději napíšeme programy vlastní. Pokud rádi píšeme programy vlastní, asi nám bude spíše vyhovovat jednodušší výchozí formát plain (csplain).

Jestliže jsme se rozhodli pracovat raději v plainu, pak lze k dalšímu studiu doporučit následující literaturu:

[1] Petr Olšák. *TeX pro pragmatiky (TeX – plainTeX – CSplain – OPmac)*. Pracovní verze textu je volně k dispozici na `http://petr.olsak.net/tpp.html`.

[2] Petr Olšák. *TeXbook naruby*. Konvoj 1997. Celý text knihy je volně k dispozici ve formátu `pdf` na `http://math.feld.cvut.cz/olsak/tbn.html`.

Vyhovuje-li nám více LaTeX, pak je možné sáhnout po těchto manuálech:

[3] Pavel Satrapa. *LaTeX pro pragmatiky*. Text je volně dostupný na `http://www.nti.tul.cz/~satrapa/docs/latex/`.

[4] Leslie Lamport. *LaTeX—A Document Preparation System—User's Guide and Reference Manual*. Addison-Wesley, Reading, MA, USA, 2nd ed. 1994.

[5] Michel Goossens, Frank Mittelbach, Alexander Samarin. *The LaTeX Companion*. Druhé vydání, Addisson Wesley 1994.

[6] Michel Goossens, Sebastian Rahtz, Frank Mittelbach. *The LaTeX Graphics Companion: Illustrating Documents with TeX and PostScript*. Addisson Wesley 1997.

[7] Jiří Rybička. *LaTeX pro začátečníky*. Druhé, upravené vydání, Konvoj 1999.

Pokud budeme používat LaTeX a budeme chtít rozumět použitým stylovým souborům, můžeme použít knihu [2]. Popisuje totiž vnitřní algoritmy TeXu, což jsou informace, které využijeme jak v plainu tak v LaTeXu.

Pro úplnost ještě citujme dva tituly. První z nich je základní biblí k TeXu od samotného autora TeXu (česká alternativa [2] ji poměrně dobře nahrazuje) a druhý titul obsahuje informace o TeXových souvislostech, tj. popis spolupracujících programů v běžných distribucích, implementace fontů, vkládání obrázků apod.

[8] Donald E. Knuth. *The TeXbook*. Mnohonásobné vydání. Addison Wesley, 1986–*. Díl A z pětidílné monografie k TeXu a METAFONTu „Computers & Typesetting“.

[9] Petr Olšák. *Typografický systém TeX*. CSTUG 1995.

Další text v tomto manuálu se věnuje základům TeXu, které bývají shodné při použití většiny formátů. Ukázky budeme pro jednoduchost nadále zkoušet ve formátu csplain, protože jinak bychom museli kolem vlastního textu ukázky přidat zmíněné „obkladové řádky“ závislé na použitém „druhu“ LaTeXu.

Technické pozadí formátů

V předchozím textu jsme na mnoha místech hovořili o formátech TeXu (plain, LaTeX, csplain), ale zatím jsme pořádně neřekli, co to je.

Formát je binární soubor (v TeXové distribuci má příponu `.fmt`), který zahrnuje:

- ▹ výchozí sadu definic, která rozšiřuje vestavěné řídicí sekvence o další, pro uživatele většinou snadněji použitelné,
- ▹ výchozí nastavení vnitřních parametrů TeXu (například šířka odstavce nebo velikost odstavcové zarážky),
- ▹ výchozí fonty, které budou v dokumentu použity, pokud uživatel nespecifikuje jiné,
- ▹ vzory dělení vybraných jazyků, podle kterých TeX dělí slova při zalamování odstavce.

Až na vzory dělení lze vše ostatní ve vlastním dokumentu pomocí TeXových definic dodatečně měnit. Pro načtení vzorů dělení jednotlivých jazyků má TeX speciální řídicí sekvenci `\patterns`, která funguje jen při vytváření formátu. Existuje ještě jedna řídicí sekvence, která má smysl pouze při generování formátu: `\dump`. Tento příkaz způsobí uložení „nabytých vědomostí“ TeXu z jeho vnitřní paměti do binárního formátového souboru `*.fmt` a ukončí činnost TeXu. Tento soubor se může později při startu TeXu načíst, a TeX tím začíná se svými znalostmi z místa, kde naposledy načítání definic skončil v době příkazu `\dump`.

Příkazy `\patterns` a `\dump` umí speciální varianta TeXu zvaná iniTeX. V nových distribucích TeXu není tato varianta reprezentována samostatným programem, ale vyvolá se prostřednictvím přepínače `-i` nebo `-ini`. Například k vygenerování formátu plain lze postupovat takto:

```
> tex -ini plain
* \dump
```

TeX zde ve variantě iniTeX načetl soubor definic plain.tex a uložil nabyté vědomosti do souboru `plain.fmt`. Nyní lze formátový soubor použít:

```
> tex -fmt plain document
```

V běžných TeXových distribucích je implementována nějakým způsobem zkratka, která uživateli umožní místo příkazu `tex -fmt plain` psát pouze `tex` a třeba místo `tex -fmt latex` psát pouze `latex`.

Speciální znaky

V této sekci popíšeme chování speciálních vstupních znaků, které nejsou TeXem většinou slepě přepisovány do výstupu, ale TeX na ně určitým způsobem zareaguje.

Jedním takovým speciálním znakem je \ (zpětné lomítko). Pokud za ním následuje písmeno, TeX přečte řídicí sekvenci typu `\slovo` ukončenou prvním znakem, který není písmeno (separátorem). Je-li tímto separátorem mezera, pak se na výstupu neobjeví. Ostatní separátory nejsou na rozdíl od mezery ignorovány. Vyzkoušejte si:

```
Zkouším \TeX.  % Tečka je separátor sekvence \TeX, který se vytiskne
Píšu v \TeX u  % Separátorem je mezera před u, která se netiskne
\TeX   je formátor.
```

Z posledního řádku ukázky dostáváme nesprávný výsledek: TEXje formátor. Projevila se totiž další vlastnost TEXu: jednu mezeru i více mezer za sebou považuje za mezeru jedinou a ta v našem příkladě funguje jako separátor, který mizí. Proto se často používají „zbytečné skupiny“, jejichž závorky mají funkci separátoru řídicí sekvence:

```
Píšu v \TeX{}u. \TeX{} je formátor.
```

Pokud za zpětným lomítkem následuje něco jiného, než písmeno (například `\$`), je řídicí sekvence tvořena jen tímto znakem. Mezery za takovými jednoznakovými sekvencemi zůstávají zachovány:

```
Pracuji se 100\% nasazením.
```

Vidíme, že dalším speciálním znakem v TEXu je mezera. Jak jsme před chvílí uvedli, více mezer za sebou se chová jako mezera jediná.

Všechny mezery na začátku řádku jsou ignorovány až po první znak, který není mezera. Toho můžeme využít pro zlepšení přehlednosti našich vstupních textů (viz naše ukázka s výčtem prvků).

Konec řádku je v TEXu interpretován jako mezera, která se vytiskne, pokud není separátorem řídicí sekvence. Pokud si mezeru z konce řádku nepřejeme, můžeme ji „zamaskovat“ komentářovým znakem:

```
Toto            je
      zvrácený
pří%
klad v~\TeX
u.
```

Prázdný řádek vytvoří interní příkaz `\par`, který ukončuje odstavec. Není-li co ukončovat, `\par` nedělá nic. Proto více prázdných řádků pod sebou se chovají stejně jako jeden prázdný řádek. I toho lze využít pro zvýšení přehlednosti zdrojových textů.

V následující tabulce je přehled všech znaků, které bývají nastaveny jako speciální:

`\`	uvozuje řídicí sekvenci
`{`	zahájení skupiny, parametru nebo definice
`}`	konec skupiny, parametru nebo definice
`$`	přepínač matematického módu
`&`	separátor používaný v tabulkách
`#`	označení parametru v definicích
`^`	konstruktor mocniny v matematickém módu
`_`	konstruktor indexu v matematickém módu
`~`	nedělitelná mezera
`%`	zahajuje na řádku komentář

Speciální význam každého znaku lze v TEXu nastavit pomocí určitých definic. Výše uvedená tabulka tedy není v ničem definitivní. Uvedený seznam speciálních znaků bývá takto nastaven ve formátech plain, csplain i LATEX.

Pokud chceme vytisknout souvislejší část textu bez speciální interpretace, musíme těmto znakům jejich speciální funkce odebrat. V LATEXu se pro tyto účely používá LATEXové prostředí vymezené příkazy `\begin{verbatim}` a `\end{verbatim}`. Vše mezi těmito příkazy[5] se vytiskne tak, jak je napsáno ve vstupním textu. Uvedené prostředí vždy ukončí odstavec a zahájí tisk textu bez speciální interpretace. Pokud chceme mít bez speciální interpretace jen část textu uvnitř odstavce, použijeme v LATEXu příkaz `\verb|text bez svislé čáry|` nebo třeba `\verb+text bez znaku plus+`.

V plainu ani v csplainu hotové řešení na vypnutí speciální interpretace znaků nenajdeme. OPmac nabízí dvojici `\begtt` a `\endtt` .

Pokud chceme vytisknout jen jednotlivé speciální znaky, měli bychom vědět, jakou sekvencí to zařídit. Pro znaky používané v běžném textu (`$`, `&`, `#` a `_`) jsou ve všech formátech připraveny řídicí sekvence `\$`, `\&`, `\#` a `\_`. Tím požadovaný znak vytiskneme v libovolném fontu. Výjimkou je znak $, který se v kurzívě Computer Modern fontu mění v libru: *£*. Je to taková malá kuriozita TEXu.

S ostatními speciálními znaky to tak jednoduché není. Zaručeně je vytiskneme pomocí `{\tt\char‘\`⟨*znak*⟩`}`. Zde `\tt` přepíná do strojopisu (v tomto fontu jsou znaky dle ASCII zaručeně přítomny) a příkazem `\char` je možné vytisknout znak s libovolnám kódem. Například `{\tt\char‘\\}` vytiskne backslash strojopisem. Je třeba upozornit na to, že chlup za příkazem `\char` v této ukázce je *zpětný apostrof*, který najdeme na klávesnici vlevo nahoře.

Znaky `< > | \ { }` nejsou v implicitním fontu Computer Modern (s výjimkou strojopisu) bohužel zastoupeny, protože se v běžném textu nevyskytují. Jsou určeny pro sazbu matematických vzorečků. V matematickém módu (mezi `$...$`) znaky `< > |` fungují přímo a pro `\ { }` má TEX rezervovány speciální řídicí sekvence: `\setminus`, `\{` a `\}`.

V LATEXu lze místo konstrukcí `{\tt\char‘\znak}` použít jednodušší `\verb|znak|`. Ovšem příkaz `\verb`, který odebírá znaku jeho speciální funkci a zapíná tisk ve strojopisu, nemusí fungovat všude. Například jej nelze použít jako argument jiného příkazu (`\section`, `\uv`, apod.).

Rozměrové jednotky používané v TEXu a typografii

Z historických důvodů v typografii stále přežívají měrné jednoty rozdílné od soustavy SI. Základní měrnou jednotkou, která se používá v anglosaských zemích, je jednotka point (`pt`), která má rozměr asi třetinu milimetru. Dvanáctinásobek je pica (čteme pajka, `pc`). Jednotkou, která se používala v Evropě, je „Didotův bod“ (`dd`), který je větší než point, ale zhruba taky měří třetinu milimetru. Dvanáctinásobek tohoto bodu je cicero (`cc`). V počítačových programech pro sazbu se používá počítačový bod (`bp`), který je jen velmi nepatrně větší. 72 počítačových bodů se přesně vejde do jednoho palce (`in`, inch používaný především v Americe).

Všechny tyto jednotky je možné použít v TEXu jako dvoupísmenové zkratky, jak ukazuje následující tabulka. Navíc lze použít jednotky odvozené z metru.

[5] s výjimkou sekvence čtrnácti znaků „`\end{verbatim}`“

`pt`	monotypový bod	1 pt = 1/72,27 in ≐ 0,35146 mm
`pc`	pica	1 pc = 12 pt
`bp`	počítačový bod	1 bp = 1/72 in
`dd`	Didotův bod	1 dd = 1238/1157 pt
`cc`	cicero	1 cc = 12 dd
`in`	palec (inch, coul)	1 in = 25,4 mm
`cm`	centimetr	1 cm = 10 mm
`mm`	milimetr	1 mm ≐ 2,84528 pt
`sp`	jednotka TEXu	1 sp = 1/65536 pt
`em`	velikost písma	závislé na aktuálním písmu
`ex`	výška malého x	závislé na aktuálním písmu

Velikost písma se měří zhruba jako celková výška řádku, který obsahuje všechny znaky písma (mimo akcentované verzálky, tj. neuvažujeme v takovém řádku háčky a čárky nad velkými písmeny). Zhruba to také odpovídá šířce velkého písmene M (odtud jednotka `em`). Bohužel, na jednotlivých písmech není nic společného, co by se dalo vždy jednoduše změřit a přesně říci, že právě to je ona velikost písma.

Práce s fonty

Implicitní fonty, které jsou v každé distribuci TEXu k dispozici a které jsou nezávislé na použitém operačním systému, jsou fonty rodiny Computer Modern. Běžná antikva v této rodině má název `cmr10`. To je zkratka pro „Computer Modern Roman ve velikosti 10pt". Fonty Computer Modern bývají už načteny ve formátu a pro jednotlivé varianty (antikva, kurzíva, polotučné, strojopis) bývají připraveny přepínače `\rm`, `\it`, `\bf`, `\tt`.

Fonty Computer Modern neobsahují akcentovaná písmena (s háčky a čárkami). Proto jsou ve formátu csplain místo nich implicitně načteny tak zvané CSfonty, které rozšiřují Computer Modern fonty o písmena s akcenty z české a slovenské abecedy. Běžná antikva v této rodině má název `csr10`. LATEX také implicitně pracuje s rodinou Computer Modern, pomocí dodatečných stylů se dá přinutit k zavedení dalších fontů. Používá k tomu zabudovaný balík makrer NFSS, který uživatele totálně odstíní od primitivního příkazu `\font`.

Ve všech formátech (v LATEXu navzdory jeho NFSS konceptu) můžete zavést nový přepínač pro nové písmo pomocí příkazu `\font`. Ten má následující syntaxi:

```
\font\přepínač=název-fontu nepovinné parametry zvětšení
```

Například

```
\font\titulfont=csr10 scaled \magstep2
```

zavede do TEXu font csr10 (tedy běžnou počeštěnou antikvu odvozenou z Computer Modern) ve zvětšení 1,44 krát normální velikost, která je 10 bodů. Tento font se pak v textu aktivuje přepínačem `\titulfont`. Proč zrovna koeficient 1,44? To je koeficient, pro který byla v TEXu vytvořena zkratka `\magstep2`. Následující tabulka shrnuje všechny zkratky typu `\magstep`, které jsou definovány ve všech běžně používaných formátech.

sekvence	koeficient	implementováno jako
`\magstep0`	1:1 (žádné zvětšení)	1000
`\magstep1`	1,2	1200
`\magstep2`	$1{,}2^2 = 1{,}44$	1440
`\magstep3`	$1{,}2^3 = 1{,}728$	1728
`\magstep4`	$1{,}2^4 = 2{,}0736$	2074
`\magstep5`	$1{,}2^5 = 2{,}48832$	2488
`\magstephalf`	$\sqrt{1{,}2} \doteq 1{,}095445$	1095

Odstupňování jednotlivých velikostí písma pomocí mocnin čísla 1,2 bývá v typografii dobrým zvykem.

Ve sloupci „implementováno jako" vidíme, že koeficient se za slovem `magstep` (stejně jako na mnoha dalších místech v TEXu) zadává jako celé číslo odpovídající tisícinásobku uvažované hodnoty. Chceme-li tedy použít font dvojnásobně velký, použijeme `scaled 2000` a při požadavku na poloviční velikost píšeme `scaled 500`.

Kromě koeficientu zvětšení (slovo `scaled`) můžeme chtít zvětšit font bez ohledu na jeho původní velikost do námi požadované velikosti. K tomu se používá slovo `at`, například:

```
\font\prvni=csr10 at 20pt
\font\druhy=csr10 scaled 2000
```

Oba řádky této ukázky zavádějí stejný font ve stejném zvětšení.

Rodina písma Computer Modern (a jeho odvozeniny, například CSfonty) obsahuje různé velikosti stejné varianty písma, přitom tyto alternativy nejsou jen stejnoměrným násobením všech rozměrů. Doporučuje se, zvláště v menších velikostech písma, používat implicitní velikost písma a dále ji nezmenšovat ani nezvětšovat. Implicitní velikost písma je označena číslem v názvu fontu, tj. například `csr10` má implicitní velikost 10 pt a `csr5` má velikost 5 pt. Srovnáme výsledek tohoto příkladu:

```
\font\zvetseny=csr5 at10pt
\font\normalni=csr10
\normalni Tady je přirozená velikost písma 10 bodů
\zvetseny a tady je písmo navržené pro pět bodů zvětšeno na 10 bodů.
```

Na výstupu dostaneme:

> Tady je přirozená velikost 10 bodů a tady je písmo navržené pro pět bodů zvětšeno na 10 bodů.

Přepnout písmo dokumentu do jiné rodiny fontů znamená postarat se o změnu významu všech přepínačů jednotlivých variant písma (`\rm`, `\bf`, `\it` a `\tt`) a nezapomenout na vhodnou změnu fontu též v nadpisech a v dalších velikostech písma, které jsou v dokumentu použity. Jednoduché definice alternativních přepínačů najdeme pro csplain v následujících souborech:

soubor	Rodina fontů
`cavantga.tex`	Avantgarde Book
`cbookman.tex`	Bookman
`chelvet.tex`	Helvetica
`cncent.tex`	New Century
`cpalatin.tex`	Palatino
`ctimes.tex`	Times Roman

Tyto rodiny fontů jsou instalovány v každé TEXové distribuci (přesněji jsou instalovány jejich volně přístupné alternativy). Chceme-li například přepnout do písma Bookman, stačí napsat do dokumentu `\input bookman`. Takovou věc jsme už ilustrovali na našem příkladě v předchozím textu.

Podíváme-li se do souborů `cbookman.tex` a dalších, které najdeme v instalaci TEXu někde v adresáři `csplain`, zjistíme, že zde nejsou předefinovány přímo přepínače `\rm`, `\bf`, `\it` a `\tt`, ale že se zde místo nich pracuje s přepínači `\tenrm`, `\tenbf`, `\tenit` a `\tentt`. To jsou totiž v plainu (i csplainu) skutečné přepínače „nejnižší úrovně". Pro uživatele se pak definují značky `\rm`, `\bf`, `\it` a `\tt` s dalším přihlédnutím na chování těchto značek v matematickém módu. Například `\bf` je definováno takto:

```
\def\bf{\tenbf \fam\bffam}
```

Je-li tedy předefinován přepínač `\tenbf`, bude se od této chvíle chovat jinak i značka `\bf`. Kód `\fam\bffam` zde nebudeme rozebírat, protože překračuje rámec tohoto úvodního textu. Spokojíme se s tím, že v textovém módu nemá tento kód žádný vliv a v matematickém módu cosi udělá.

Příkaz `\fontname\přepínač\space` se promění zpět v původní název fontu ukončený mezerou. Tato vlastnost byla použita v našem příkladu, kde jsme zaváděli větší font pomocí této konstrukce:

```
\font\titulfont=\fontname\tenbf\space scaled \magstep2  % větší font
```

Výhodou tohoto zápisu je fakt, že nemusíme znát název fontu, stačí si zapamatovat základní přepínače `\tenrm`, `\tenbf`, `\tenit` a `\tentt`. Názvy fontů se nejenom těžko pamatují, ale také se mohou změnit, pokud před takovou konstrukcí použijeme `\input cbookman` nebo něco podobného.

V LATEXu asi takové obraty nebudeme potřebovat, protože o zavedení potřebných fontů pro různé velikosti se LATEX stará sám. Pro přepínání mezi rodinami fontů používáme v LATEXu příkaz `\usepackage` a následuje ve složených závorkách jedno ze slov `avantgar`, `bookman`, `helvet`, `newcent`, `palatino`, `times`. Písmeno `c` na začátku názvu rodiny fontů (na rozdíl od csplainu) nepíšeme.

Možná nás začne zajímat, jaké fonty máme v TEXové instalaci připraveny k použití. Stačí udělat menší průzkum v adresáři `tfm` (odvozeno od zkratky TEX font metrics) a podívat se do jednotlivých podadresářů na názvy přítomných souborů. To jsou současně názvy fontů, které jsou použitelné v příkaze `\font`. Chceme vědět, jak který font vypadá? Napišme na příkazový řádek

```
tex testfont
```

TeX se nás vyptá na název fontu, který zadáme bez přípony `.tfm`. Pak nás požádá o instrukci, co s načteným fontem má dělat. Nejlépe je odpovědět `\table\end` a podívat se na tabulku znaků testovaného fontu třeba pomocí `xdvi testfont`.

Umístění sazby na papíře

Při poznávání TeXu si jistě velmi brzo položíme otázku, jak je možné změnit velikost okrajů, neboli jak umístit sazbu na papíře.

V plainu jsou implicitně nastaveny velikosti okrajů jeden palec z každé strany papíru amerického formátu Letter. Takové formáty papíru u nás většinou nerostou, takže plain nám na papíru A4 udělá jen levý a horní okraj velikosti jeden palec a pravý okraj bude menší a spodní větší.

V csplainu jsou implicitně nastaveny velikosti okrajů jeden palec z každé strany pro formát A4. Sazba je tedy v csplainu mírně užší a vyšší, než v plainu. Sazbu přitom měříme bez případného záhlaví a bez stránkových číslic.

Po zavedení makra OPmac v plainu nebo csplainu je možné okraje pohodlně nastavit makrem `\margins`. Jak to udělat je popsáno v dokumentaci k OPmac. Níže je uveden postup nastavení okrajů v TeXu na úrovni TeXu samostného bez použití maker.

Umístění sazby měříme vzhledem k počátku, který se nalézá na papíře 1 palec od levého okraje a 1 palec od horního okraje. Levý horní roh sazby se kryje s tímto počátkem, pokud jsou nastaveny registry `\hoffset=0pt` a `\voffset=0pt`. Levý horní roh sazby se posune doprava o hodnotu `\hoffset` a dolů o hodnotu `\voffset`. Při záporných hodnotách těchto registrů se sazba posunuje samozřejmě doleva respektive nahoru.

Šířka sazby (přesněji šířka zpracovávaného odstavce) se nastaví pomocí registru `\hsize`. Výška sazby na stránce se nastaví pomocí `\vsize`. V následující ukázce jsou uvedeny hodnoty, které nastavuje plain.

```
\voffset=0in       % velikost horního okraje = \voffset + 1 palec
\hoffset=0in       % velikost levého okraje  = \hoffset + 1 palec
\hsize=6.5in       % šířka řádku, 165.1mm
\vsize=8.9in       % výška sazby, 266mm
```

Formát csplain má registry `\hoffset` a `\voffset` také nulové, ale šířku a výšku sazby nastavuje odlišně:

```
\hsize= 159.2 mm   % šířka řádku v csplainu (šířka A4 - 2in)
\vsize= 239.2 mm   % výška sazby (výška A4 - 2in)
```

Pokud chceme nastavit vlastní velikosti, doporučujeme nejprve registry `\hoffset` a `\voffset` nastavit na hodnotu −1 in a pak k nim přičíst hodnoty požadovaných okrajů pomocí příkazu `\advance`. Dále doporučujeme výšku sazby přesně rozměřit na počet řádků. K tomu potřebujeme vědět, že vzdálenost dvou řádků se určí pomocí registru `\baselineskip` (pozor: při větším písmu se toto řádkování může rozhodit). Plain i csplain nastavují `\baselineskip` na 12 pt. Kromě toho je účaří prvního řádku od pomyslného horního okraje sazby vzdáleno o `\topskip`, který má v plainu i v csplainu hodnotu 10 pt. Protože se výška sazby `\vsize` měří od horního pomyslného okraje po účaří posledního řádku na stránce, vychází `\vsize` jako `\topskip` + $(n-1)\times$ `\baselineskip`, kde n je počet řádků na stránce. Nastavení velikosti sazby tedy můžeme udělat například takto:

```
\voffset=-1in
\advance\voffset by 2cm   % velikost horního okraje bude 2cm
\hoffset=-1in
\advance\hoffset by 1.5cm % velikost pravého okraje bude 1.5cm
\hsize=10cm               % šířka řádku bude 10cm
\vsize=\topskip
\advance\vsize by 15\baselineskip % sazba bude mít 16 řádků na stránce
```

V LaTeXu se při nastavování rozměrů sazby používají registry speciálně deklarované v tomto formátu. Jedná se o `\textheight` (výška sazby), `\textwidth` (šířka sazby), `\oddsidemargin` (levý okraj na lichých stránkách), `\evensidemargin` (levý okraj na sudých stránkách) a `\topmargin` (horní okraj). LaTeX pak sám podle hodnot těchto registrů nastaví vnitřní registry TeXu `\hoffset`, `\voffset`, `\hsize` a `\vsize`. Uživatel LaTeXu by k nim neměl přistupovat přímo a navíc by měl s registry zacházet „LaTeXovsky", což prakticky znamená, že místo jednoduchého přiřazení nebo příkazu `\advance` by měl zapisovat své požadavky zhruba takto:

```
\setlength\topmargin{-1in}
\addtolength\topmargin{2cm}       % velikost horního okraje bude 2cm
\setlength\oddsidemargin{-1in}
\addtolength\oddsidemargin{1.5cm} % velikost pravého okraje bude 1.5cm
\setlength\evensidemargin{\oddsidemargin}
\setlength\textwidth{10cm}        % šířka sazby bude 10cm
\setlength\textheight{\topskip}
\addtolength\textheight{15\baselineskip} % 16 řádků
```

TeXem většinou nenastavujeme parametry pro archovou montáž sazby, takže nám výše uvedené příklady pro nastavení velikosti sazby bohatě stačí. Pokud bychom chtěli se sazbou dále manipulovat a umisťovat ji na jednotlivé archy podle určitých požadavků, použijeme většinou pomocné programy, které manipulují s PostScriptovým výstupem. Představme si, že chceme stránky tohoto manuálu zmenšit tak, aby se vešly dvě vedle sebe na stranu A4. Dále chceme tyto stránky uspořádat tak, abychom po oboustranném vytištění manuálu na šest archů A4 dostali svazeček, který přeložíme v půli a máme knížečku s 24 na sebe navazujícími stránkami. Pro takový úkol se asi nejlépe hodí programy z volně šířeného balíčku psutils. Na příkazový řádek můžeme postupně napsat tyto instrukce:

```
> csplain prvni
> dvips prvni
> psbook prvni.ps p0.ps
> pstops "4:0L@.7(21cm,.5cm)+1L@.7(21cm,14.4cm)" p0.ps p1.ps
> pstops "4:2L@.7(21cm,.5cm)+3L@.7(21cm,14.4cm)" p0.ps p2.ps
> lpr -Ptiskarna p1.ps
> lpr -Ptiskarna p2.ps
> rm prvni.ps p0.ps p1.ps p2.ps
```

Příkaz `dvips` převede dokument do PostScriptu a `psbook` uspořádá stránky pro použití do „svazečku". První volání příkazu `pstops` vybere vždy dvě ze čtyř stránek, zmenší je na 0,7 násobek původní velikosti (`@.7`) a umístí je do archu podle uvedených parametrů. Tím vzniká podklad pro tisk lícových stran archů A4 (`p1.ps`). Podobně druhé volání příkazu `pstops` vytvoří podklad pro rubovou stranu archů. Vlastní tisk (`lpr`) pak můžeme provést na tiskárně, která neumí oboustranný tisk, ve dvou průchodech. Před druhým průchodem obrátíme vytištěné papíry a vložíme je do zásobníku tiskárny znovu.

Overfull/Underfull box

Při práci s TEXem narazíme postupně na celou řadu chybových hlášení, při kterých se program většinou zastaví a vyzve nás k nějaké akci. Ačkoli třeba jen stiskneme klávesu Enter, uvědomíme si, že je něco špatně a pokusíme se chybu řešit.

Kromě toho TEX vypisuje varování o přetečených (overfull) a nedoplněných (underfull) boxech. Protože se při těchto výpisech nezastavuje, považují to mnozí začátečníci za menší zlo, kterého si není nutné všímat. Není to tak docela pravda.

Přetečené boxy (overfull) bychom měli rovněž zařadit do kategorie chyb. TEXu se totiž nepodařilo vměstnat sazbu do předepsané šířky `\hsize`. Prakticky to znamená, že sazba v daném místě „vyčnívá" na pravé straně ven směrem do okraje. Hlášení obsahuje údaj, o kolik bodů sazba vyčnívá, číslo řádku ve zdrojovém kódu a kus textu, který určuje problémové místo. Plain a csplain dále nastavuje registr `\overfullrule` na 5 pt, takže se v sazbě na problémovém místě objeví těžko přehlédnutelný černý obdélník. LATEX tento registr nuluje, takže černé obdélníky nejsou vidět, což dává uživateli pocit, že je všechno v pořádku.

Objeví-li se přetečený box v odstavci, většinou stačí „rozvolnit" mezery. Mezery mezi slovy mají totiž pružnost (mohou se smršťovat nebo natahovat). Tato pružnost není neomezená, ale je daná jistými parametry podle použitého fontu. Chceme-li dát mezerám větší volnost v roztahování, než si přál autor fontu, pišme například `\emergencystretch=2cm`. TEX má plno dalších vnitřních registrů, jejichž nastavením ovlivníme algoritmy na sestavování odstavce. Jejich popis ovšem překračuje rámec tohoto úvodního textu.

Nedoplněné boxy (underfull) můžeme na rozdíl od přetečených boxů považovat pouze za varování. TEX nás informuje, že byl nucen v některém místě natáhnout mezery víc, než je esteticky zdrávo. Hodnota `badness`, která hlášení doprovází, udává zhruba stupeň estetické vady v takovém místě (nebo také velikost násilí provedené na mezerách). Čím vyšší badness, tím horší výsledek. Maximální hodnota badness je 1000, což značí, že některý řádek je úplně špatně. Stojí zato se podívat do sazby na takto označená místa a zamyslet se, co by se dalo změnit, aby se zlepšila estetická úroveň výsledku.

LATEXoví uživatelé dosti často neopatrně pracují s příkazem `\\`, který se v různých LATEXových prostředích chová jako ukončení řádku. Někdy se dostane takový příkaz i na konec odstavce, což samo o sobě nemá logiku, protože na konci odstavce se samo sebou ukončí řádek. Pokud se tak stane, TEX na konci odstavce vytvoří ještě další prázdný řádek, ve kterém nemá žádnou mezeru k natažení na šířku `\hsize` a oprávněně se rozčílí: `Underfull hbox badness 1000`. V tomto případě vlastně TEX křičí na uživatele, který nebyl schopen opustit principy starodávného psacího stroje a potřebuje mít k ruce tu velikou páku, do které je občas potřeba praštit, aby se přešlo na nový řádek.