A "virgin" TEX system that has no macros is like newborn baby that has an immense amount to learn about the real world; but it is capable of learning fast.
Donald Ervin Knuth: *The TEXbook.*

Kapitola třetí: Nejběžnější formáty a styly

Petr Olšák

Vraťme se nyní k našemu příteli a zkusme ho požádat o to, zda bychom si od něj nemohli „nakopírovat" TEX. Ačkoli jsme přesvědčeni, že zde nebudou žádné problémy, náš přítel nás vyvede z omylu. Postaví nás před hromadu pojmů a nutí nás, jako novopečené adepty na titul „uživatel TEXu", abychom se v nich orientovali a dokázali si vybrat, co potřebujeme. Například se bude hovořit o tom, že pro PCčka je asi nejvhodnější emTEX, že existují jakési komerční verze PCTEXu a případně další. Že je nutno se rozhodnout, zda budeme používat plainTEX, nebo se raději poohlédneme po LATEXu nebo AMS-TEXu, či rovnou nezkusíme jejich kombinace, jako je např. LAMSTEX, nebo AMS-LATEX. Pokud se budete bavit s člověkem, který do toho vidí, začne Vás přesvědčovat, že bez NFSS se neobejdete...

Začínající uživatel se nemá šanci samostatně rozhodnout a většinou jeho rozhodnutí vyplyne z toho, co používají jeho přátelé, kteří mu systém nabízejí nebo doporučují. Bohužel se setkávám s případy, že existují uživatelé TEXu, kteří o sobě tvrdí, že používají např. LATEX (mají pravdu), ale nejsou schopni odpovědět na otázku, jaký je rozdíl a jaká je souvislost mezi TEXem a LATEXem a proč se na jejich terminálu objevuje hlášení:

```
This is emTeX, Version 3.0 [3a] (no format preloaded) .
```

Pokusím se v této kapitole vnést do těchto pojmů trochu světla.

Programátor verze TEXu pro počítače řady PC výjde z Knuthova druhého dílu *TEX – the program*, v němž je podrobně popsán zdrojový text TEXu, ale to mu nestačí. Aby tak velký program „vměstnal" do tak malého počítače, k tomu musí vynaložit dost velké úsilí a kus svého vlastního programátorského umění. Proto se nedivím, že se nakonec do verze TEXu pro počítače řady PC podepíše tak, jak to udělal Eberhard Mattes, student techniky v Německu, který vytvořil public domain verzi TEXu pro DOS a OS/2. Tato verze se jmenuje emTEX a musela projít Knuthovými testy, aby vůbec mohla nést označení TEX. Tím je zaručena kompatibilita s jinými výpočetními systémy.

Mattesova práce nekončí vytvořením TEXu pro naše PCčka. Vytvořil také velice kvalitní obslužné programy, bez nichž by byl TEX samotný k ničemu. Jedná se o prohlížeč, který pracuje pod nejrůznějšími grafickými kartami a o tiskové programy pro výstup na všechny možné tiskárny. Navíc, aby balík programů byl úplný, nesmí chybět METAFONT pro výrobu fontů k TEXu a různé konverzní programy, které s tím souvisí. Často se pak emTEXem myslí nejen program `tex.exe`, ale celý balík programů.

Pokud tedy budete chtít instalovat kvalitní public domain verzi TEXu pro počítače řady PC, nelze v současné době doporučit nic lepšího, než je emTEX. Získáte ho například od CSTUGu, který jednoznačně podporuje jeho šíření. V další kapitole si povíme něco víc o vnitřní struktuře instalovaného balíku emTEX.

V tomto článku se nebudu zmiňovat o verzích TEXu pod jinými systémy. Přiznám se, že jsem sám neměl možnost blíž se s takovými verzemi seznámit. Nicméně lišit se může pouze způsob ovládání prohlížečů a možnosti tisku na různých výstupních zařízeních, případně způsob spouštění TEXu. Přitom TEX samotný musí pracovat ve všech verzích naprosto stejně přesně tak, jak je psáno v Knuthových knihách.

Mnohem problematičtější je pro nového uživatele výběr formátu, pod kterým bude za pomocí TEXu zpracovávat své dokumenty. Připomeňme si, že TEX je vybaven jistým „programovacím jazykem", tzv. jazykem maker. Tato makra po načtení TEXem mohou velmi podstatně ovlivnit jeho další činnost. TEX sám v okamžiku startu je vybaven zhruba třemi sty tzv. TEX-primitivními příkazy, kterými sice lze vyjádřit všechno, ale je třeba nejprve napsat plno věcí, které jsou uživatelsky naprosto nezajímavé. V této souvislosti si povšimněme citátu, který je uveden v záhlaví k této kapitole.

Aby se nemuselo vždy vypisovat množství (pro uživatele většinou naprosto nepochopitelných) příkazů na začátek každého dokumentu, byly vytvořeny textové soubory maker (tzv. formáty), které se po startu TEXu nabídnou ke čtení jako první. Po jejich přečtení je TEX informován o konkrétních fontech a zná množství příkazů, kterými se tyto fonty vyvolají. Navíc umí plno dalších uživatelsky jednoduchých příkazů (například na vytváření poznámek pod čarou, nebo na sestavovaní rejstříků), které byly definovány v těchto formátech většinou velmi komplikovaným způsobem. Uživatel nyní nemusí vědět, jak jsou tyto příkazy definovány, ale stačí, aby věděl, jak tyto příkazy má používat.

Tento přístup má jednu obrovskou výhodu. Tvorba formátů je svěřena do rukou odborníků v oboru typografie a TEXu. Například uživatel ví, že pokud chce napsat poznámku pod čarou, použije příkaz `\footnote`. V tom místě taky napíše text poznámky. V té době ještě není známo, kde TEX zalomí novou stránku a proto v okamžiku, kdy se při zpracování narazí na příkaz `\footnote`, musí si TEX „zapamatovat" text poznámky a do místa výskytu příkazu musí vložit odkaz na poznámku (například hvězdičku). Dále si musí zkrátit rozměr tiskového zrcadla tak, aby po zalomení stránky mohl vložit dolů text poznámky a přitom celkový vertikální rozměr tiskového zrcadla včetně poznámky zůstal nezměněn. Nakonec je třeba definovat přístup k nějakým (většinou menším) typům písma pro vlastní text poznámky pod čarou. Toto je jen velmi zjednodušený pokus seznámit čtenáře s tím, na co vše musí myslet „programátor“ příkazu `\footnote`, než jej napíše do formátu a odladí. Nakonec k němu musí napsat dokumentaci, aby jej uživatelé TEXu mohli používat.

Prvním „programátorem“ formátu pro TEX byl sám autor Donald Knuth. Formát nazval plainTEX a s podrobnými komentáři jej publikoval ve své knize *The TEXbook*. Znamená to, že po důkladném prostudování této knihy se vlastně můžete i vy stát programátory nových formátů. Máte tam napsaný velký vzor. To ovšem vyžaduje nejen programátorské umění psát makra v TEXu, ale také typografické vzdělání a cit k typografickým zásadám a tradicím. Je proto moc dobře, že je tvorba formátů relativně dosti složitá a že to nemůže dělat každý, kdo si letmo přečte *The TEXbook*. Když někdy vidím některé textové produkty výpočetní techniky, je mi smutno, že se typografické zásady neprobírají na školách jako povinný předmět. V současné době totiž počítače umí ovládat skoro každý a tím se skoro každému dostává do rukou mocný nástroj na psaní rozmanitých textů. Pokud je tento nástroj použit lidmi, kteří nemají nejmenší typografický cit, pak to dopadá tak, jak to dopadá. Kde jsou ty časy, kdy byla tvorba tiskovin pouze v rukách odborníků! Nejvíce si tuto skutečnost uvědomoval sám autor Donald Knuth, který se nestyděl napsat, že na vzhledu své knihy spolupracoval s odborníkem na typografii.

PlainTEX je základním formátem a většina ostatních mezinárodně užívaných formátů pro psaní technických publikací je z tohoto formátu více či méně odvozena. PlainTEX definuje přístup ke Computer Modern fontům. Dále vymezuje jakési základní postupy, použité i v jiných formátech (například přístup k vnitřním registrům TEXu). S tímto formátem lze napsat jednoduchý text, který nevyžaduje příliš mnoho složitých typografických operací. Nečekejte, že tam najdete například příkaz pro automatické generování obsahu. Není tam ani automatické číslování rovnic, kapitol a podobně. Lidé, kteří píší v plainTEXu se dělí na dvě skupiny. První z nich nepotřebují psát složité dokumenty a druzí jsou ti, kteří podrobně přečetli *The TEXbook* a tento formát považují za výchozí pro své další tvůrčí úsilí. Tito lidé si už dokážou například příkaz na automatické generování obsahu sami definovat. Přitom použití vyššího formátu je pro ně matoucí, protože tyto formáty jsou z jejich pohledu těžko zevnitř rozluštitelné a není snadné vždy zjistit, co vlastně v daný okamžik TEX po přečtení těchto složitých formátů dělá. Přitom se nespokojí s uživatelskou dokumentací k těmto formátům.

PlainTEX rozšiřuje počet příkazů, které může uživatel použít asi na devět stovek. Použití plainTEXu je ilustrováno ukázkou o telefonních ústřednících z první kapitoly. Na této jednoduché ukázce si pro zajímavost ukážeme, které příkazy jsou definované ve formátu a které jsou TEX-primitivní. Nahlédneme tak do zákulisí tvorby formátů. Při výkladu jednotlivých příkazů nebudu vždy zcela přesný, protože bych se zamotal do podrobností a mohlo by to snížit čitelnost tohoto textu.

TEX-primitivní příkazy přímo použité v naší ukázce jsou dva: `\input` pro načtení souboru a `\vskip` pro vložení vertikální mezery. Příkaz `\rightline` je v plainTEXu definován takto:

```
\def\rightline#1{\line{\hss#1}}
```

což znamená, že se zde definuje příkaz `\rightline` s jedním formálním parametrem `#1`. Za tento parametr se v okamžiku použití příkazu dosadí obsah složené závorky za příkazem.

Podívejme se například na příkaz `\rightline{V Praze dne \today}`. Ten je podle definice „rozvinut“ (expandován) do tvaru:

```
\line{\hss V Praze dne \today}
```

Příkaz `\line` je v plainTEXu definován pomocí definice

```
\def\line{\hbox to \hsize}
```

takže náš příkaz `\rightline` se ještě dále expanduje až do tvaru:

```
\hbox to \hsize{\hss V Praze dne \today}
```

a to už jsou TEX-primitivní příkazy (až na příkaz `\today`, o němž jsme si řekli, že je definován ve stylovém souboru `czech.sty` a expanduje se například na text `23. září 1992`). Příkaz `\hbox to \hsize` praví, že se vytvoří box s textem uvedeným v závorce a že šířka tohoto boxu bude rovna parametru `\hsize`, který představuje šířku tiskového zrcadla. Formát plainTEXu definuje hodnotu tohoto parametru takto:

```
\hsize=6.5in
```

tedy 6.5 palce, což je zhruba 16.5 cm. Uživatel může samozřejmě tuto hodnotu změnit. Příkaz `\hss` znamená vložení „natahovací“ mezery, přesněji text v boxu se vypíše bez deformací a do místa příkazu `\hss` se vloží taková mezera, aby celková šířka boxu odpovídala hodnotě parametru `\hsize`.

Podívejme se ještě na příkaz `\bye`, který je v naší ukázce úplně na konci. Tento příkaz je definován v plainTEXu jako skupina příkazů:

```
\def\bye{\par\vfill\supereject\end}
```

které po řadě znamenají: ukonči odstavec, vlož vertikální natahovací mezeru až do spodního okraje stránky (prázdné místo do konce stránky), vyvolej rutinu na zalomení stránky a ukonči činnost. Všechny tyto příkazy až na `\supereject` jsou už TEX-primitivní. Pokud chcete vědět, jak je definován `\supereject`, to si rovnou pořiďte *The TEXbook*!

Příkazy `\bf` a `\it` definují přístup k určitým fontům. Vysvětlení jejich definic by překročilo rámec tohoto článku.

Z tohoto velmi jednoduchého příkladu je zřejmé, že pro uživatele je lepší nevědět, jak jsou jednotlivé příkazy, které používá, definovány. Na druhé straně zájemce o programování v TEXu jsem asi touto jednoduchou ukázkou neuspokojil. K některým dalším vlastnostem „programovacího jazyka“ TEXu se vrátíme možná někdy příště.

Věnujme nyní pozornost některým dalším mezinárodně uznávaným formátům. Mezi ně patří LATEX a $\mathcal{AMS}$-TEX. Začněme LATEXem. Jedná se o formát TEXu, vytvořený panem Leslie Lamportem. Uživatelským manuálem k tomuto formátu je kniha:

Leslie Lamport: LATEX – *A Document Preparation System*. Addison Wesley, 1986.

Pro novopečeného uživatele je však i tato kniha dosti velkým soustem a rád by se poohlédnul nejdříve po nějakém stručném přehledu toho, co LATEX umí. Odborníci z oboru LATEX mi snad prominou, že si nevzpomenu na nic jiného, než na brožuru

Karel Pala: *Úvod do systému* LATEX. Ediční středisko ČVUT, 1990.

Jedná se o útlou knížečku pro studium na jeden večer. Pro složitější práci ovšem nezbyde, než si přečíst Lamportův manuál.

LATEX umí automatické číslování všech možných objektů, které se v technických dokumentech vyskytují. Jedná se o číslování rovnic, tabulek, obrázků, kapitol, odstavců apod. Dokáže automaticky sestavit obsah pouze napsáním příkazu `\tableofcontents`. Velikost nadpisů kapitol, odstavců apod. jsou v takovém poměru, aby to bylo typograficky v pořádku. Umí sestavovat rejstříky, psát text ve dvou sloupcích, sestavovat „plovoucí záhlaví“ podle názvu kapitol a odstavců apod. Dále je vybaven příkazy pro tvorbu jednoduchých obrázků (skládajících se z úseček a malých oblouků).

Text psaný ve formátu LATEX je poznat na dálku svými příkazy `\begin` a `\end`, která vymezují podobně jako v Pascalu jistá prostředí, uvnitř nichž mohou být další prostředí. Když jsme se seznámili s plainTEXem, může se nám zdát text v LATEXu příliš „ukecaný“. Autor formátu totiž nutí uživatele, aby důsledně dodržovali stanovená prostředí opatřená příkazy `\begin` a `\end`. Náš příklad s telefonními ústředníky vypadá v LATEXu takto:

```
\documentstyle[czech]{article}
\csprime
\pagestyle{empty}

\begin{document}
  \begin{flushright}
      V Praze dne \today
  \end{flushright}

  {\bf Vážení telefonní ústředníci!}

  \vspace{0.9cm}
  Dovolte mi, abych Vás upozornil, ...
  ... v hlavním městě státu.

  \vspace{0.6cm}
  Příjemnou zábavu při čtení urgencí jednotlivých žádostí o telefon přeje

  \begin{flushright}
     \it Ferdinand Mravenec
  \end{flushright}
\end{document}
```

Vidíme, že například požadavek vysázet text na pravou zarážku vyjádříme v LATEXu pomocí prostředí s názvem `flushright`. Příkaz `\rightline` LATEX nezná. Pokud bychom ho chtěli přepsat pomocí příkazu

```
\line{\hss V Praze dne \today}
```

asi bychom se velmi podivili. LATEX sice zná příkaz `\line`, ale používá ho v naprosto jiném významu, než plainTEX. LATEXovský příkaz `\line` se používá k nakreslení úsečky v prostředí pro tvorbu obrázků. Vidíme tedy, že LATEX není kompatibilní se svým předchůdcem plainTEXem. Nicméně TEX-primitivní příkazy lze samozřejmě v LATEXu použít.

Je zřejmé, že viditelná „ukecanost“ LaTeXu a jeho nekompatibilita se základním formátem může někoho odradit. Na druhé straně ale musíme zvážit výhody LaTeXu. Představme si, že v našem příkladě chceme pro oslovení telefonních ústředníků použít poněkud větší písmena. Netroufal jsem si ukázat v první kapitole, jak by se takový problém řešil v plainTeXu, protože to vyžaduje větší zkušenosti s tím, jak TeX pracuje s fonty a jak se definuje přístup k novému fontu pomocí TeX-primitivního příkazu `\font`. Na druhé straně v LaTeXu je to snadné. Stačí například napsat

```
{\Large \bf Vážení telefonní ústředníci!}
```

a máme oslovení větší! Přitom jako uživatelé vůbec nemusíme vědět, jak je „zařízen“ přístup k novému fontu.

Zatímco tedy plainTeX definuje přístup pouze k základní velikosti fontů (tzv. desetibodové písmo – velikost písma, kterou je napsán tento článek), umožňuje LaTeX pomocí příkazů `\tiny`, `\small`, `\large`, `\Large`, `\huge` a dalších snadný přístup k fontům různě zmenšeným a zvětšeným. Skoro ve všech velikostech lze použít přepínače `\rm`, `\it`, `\bf`, `\sl`, `\sf`, `\sc`, které po řadě znamenají románské písmo, kurzíva, tučné, skloněné, bezpatkové a malé kapitálky. Když to novopečený uživatel zjistí, zajásá a jeho první dokument se bude hemžit všemi možnými druhy písma ve všech možných velikostech. Právě to je proti všem zásadám typografické kultury. Naštěstí toto opojení s možností použít různé typy písma časem přejde a u většiny uživatelů LaTeXu začne rozumná práce.

Ještě si povšimněme hned prvního příkazu `\documentstyle`. Zde se definuje, jaký stylový soubor se má načíst. Kupodivu formát LaTeXu sám o sobě nestačí k tomu, aby po jeho načtení byl TeX ochoten číst náš dokument. Je třeba ještě načíst některý ze základních stylových souborů LaTeXu: `article.sty`, `book.sty`, `report.sty`, `letter.sty`, nebo nějaký jiný. Tyto soubory doplňují definice příkazů LaTeXu o plno parametrů, které jsou v LaTeXu samotném definovány jen obecně. Jde například o velikost tiskového zrcadla, velikosti písma pro nadpisy, způsob umístění a číslování těchto nadpisů a mnoho dalších. Z uvedených souborů si musíme jeden vybrat a uvést ho do příkazu `\documentstyle` do složené závorky.

Smysl použití stylů můžeme ukázat i na našem jednoduchém příkladě. Představme si, že nám nějaký odborník připraví styl speciálně určený pro psaní dopisů s názvem `myletter.sty`. Potom by zdrojový text našeho dopisu mohl vypadat takto:

```
\documentstyle{myletter}
\osloveni{Vážení telefonní ústředníci!}
\begin{dopis}
  Dovolte mi, abych Vás upozornil, ...
  ... v hlavním městě státu.
\end{dopis}
\rozlouceni{Příjemnou zábavu při čtení urgencí jednotlivých žádostí o telefon přeje}
\podpis{Ferdinand Mravenec}
```

Styl se sám postará o tisk hlavičky firmy, data vytištění dopisu, vhodně rozvrhne vertikální mezery mezi jednotlivými částmi dopisu apod. Kromě příkazů `\osloveni`, `\rozlouceni` a `\podpis` styl může definovat ještě třeba příkaz `\adresa`, za nějž by se dala napsat adresa příjemce, přičemž tato adresa se vysází ve vhodném tvaru na vhodném místě na stránce. Pokud byste požadovali, aby byla adresa vytištěna ještě jednou, tentokrát na obálku dopisu, i to by šlo zařídit ve stylu `myletter`. Jediné, co k tomu potřebujete, je TeX a odborníka na LaTeXovské styly. Samozřejmě, že pokud Váš odborník dává přednost plainTeXu, napíše styl pravděpodobně v něm. Vše pak může vypadat úplně jinak.

Používání stylů není tedy pouze specialitou LaTeXu, nicméně LaTeX požaduje v úvodním příkazu `\documentstyle` povinné uvedení aspoň jednoho stylu. Takový styl se píše do složených závorek, zatímco ostatní (nepovinné) styly se zapíší do hranatých závorek.

Za zmínku stojí ještě schopnost LaTeXu vyhodnocovat křížové odkazy. Například text

```
... viz tabulku 3.2 na straně 28 ...
```

nemůžeme do dokumentu napsat „natvrdo“, protože v okamžiku, kdy pořizujeme zdrojový text, nemáme tušení, na které stránce se objeví naše tabulka. V LaTeXu můžeme napsat například

```
... viz tabulku \ref{seznam} na straně \pageref{seznam} ...
```

přičemž slovo `seznam` je pouze naše pracovní značka, kterou uvedeme v místě, kde je napsána tabulka. LaTeX všechna čísla automaticky doplní. Pozorní a zkušení uživatelé TeXu mi jistě neodpustí, že jsem nepotlačil řádkový zlom mezi slovem `tabulku` a číslem tabulky. Správně tedy náš příklad má být zapsán takto:

```
... viz tabulku~\ref{seznam} na straně~\pageref{seznam} ...
```

Nyní si povíme trochu stručněji něco o $\mathcal{AMS}$-TeXu. Formát $\mathcal{AMS}$-TeX a zvláště jeho nadstavbový styl `amsppt.sty` byl původně vyvinut pro psaní preprintů (vědeckých článků) pro American Mathematical Society. Tato společnost přijímala (a přijímá) články k otištění v časopisech v tomto formátu. Brzy se tento formát stal mezinárodně uznávaným standardem pro psaní vědeckých publikací (zvláště v matematice).

Chcete-li snadno a rychle začít psát v $\mathcal{AMS}$-TEXu, stačí si pořídit knížečku

Michaela Lichá, Oldřich Ulrych: $\mathcal{AMS}$-TEX – *verze 2.1*. Československé sdružení uživatelů TEXu, Praha 1992.

Úplný manuál k $\mathcal{AMS}$-TEXu najdete v knize autora tohoto formátu:

Michal Spivak: *The Joy of TEX*. Addison Wesley 1982.

$\mathcal{AMS}$-TEX na rozdíl od LATEXu není tak „ukecaný“. Navíc definuje přístup k obrovské skupině nových fontů, které byly vytvořeny pro použití v TEXu později. Jedná se o nepřeberné množství matematických značek, zdvojená písmena, gotická fraktura a mnoho dalších. Dále umožňuje poměrně snadno psát velmi komplikované matematické formule a obrazce. Další jeho výhodou ve srovnání s LATEXem je jeho podstatně lepší kompatibilita s výchozím formátem plainTEX. Takže například příkaz `\line` funguje v $\mathcal{AMS}$-TEXu stejně jako v plainTEXu.

Na druhé straně však $\mathcal{AMS}$-TEX nemá schopnost automatického číslování a neumí ani křížové reference.

Dále jsem se setkal s jakýmisi kombinacemi uvedených formátů. Například L$\mathcal{AMS}$-TEX je určen pro uživatele $\mathcal{AMS}$-TEXu, kterým chyběly výhody LATEXu. Je zde doplněno automatické číslování a odkazy. Manuál k L$\mathcal{AMS}$-TEXu je samostatná kniha, nicméně je doporučena především uživatelům $\mathcal{AMS}$-TEXu. Podobně se LATEX inspiroval výhodami $\mathcal{AMS}$-TEXu a vzniknul $\mathcal{AMS}$-LATEX. Manuál k němu předpokládá znalost LATEXu.

Jistým „doplňkem“ k formátům je soubor velmi chytře vyřešených definic přístupu k velkému množství fontů, přičemž volba fontu je z uživatelského hlediska celkem snadná. Jedná se o tzv. NFSS (New Font Selection Scheme). Tento soubor definic lze zaměnit za příslušné původní definice přístupu k fontům v jednotlivých formátech. NFSS pracuje s mnoha méně běžnými fonty, jako je například tučná kurzíva nebo skloněný strojopis. Příkazy `\it`, `\bf` a další, které za normálních okolností fungují jako přepínače mezi jednotlivými fonty, zde mají jinou, poněkud komplikovanější, funkci. Například

```
{\tt strojopis \it kurzívou} a {\it kurzíva \bf tučně}
```

se vysází jako `strojopis` *`kurzívou`* a *kurzíva* ***tučně***. Přitom bez použití NFSS fungují příkazy `\bf`, `\it` a `\tt` jako přepínače mezi fonty, takže výsledek vypadá takto: `strojopis` *kurzívou* a *kurzíva* **tučně**.

Nakonec se ještě věnujme vztahu mezi formáty a styly. Obojí jsou textové soubory, obsahující definice nových příkazů a nastavení vnitřních parametrů TEXu. Tvorba formátů i stylů patří do rukou odborníkům. Na obojí se díváme jako na doplňkový software k TEXu napsaný v jazyce TEXu. Většinou je i tento software public domain. Ani z hlediska TEXu samotného není mezi formáty a styly žádný rozdíl. Nicméně formáty jsou většinou psány pro obecnější použití, zatímco styly upřesňují, jak bude celkový vzhled dokumentu vypadat. Dále formáty bývají velmi rozsáhlé (například formát LATEXu obsahuje přes 10 000 řádků textu), zatímco styly jsou již podstatně stručnější.

Často bývá rozdíl i při technické realizaci běhu TEXu. Nedovedu si představit, kam až by sahala trpělivost uživatele, který píše například v LATEXu krátkou desetiřádkovou zprávu, kdyby musel při každé korektuře čekat, až TEX načte přes 10 000 řádků textu. To naštěstí není potřeba. TEX umí uložit veškeré paměťové struktury, které při načtení textového souboru průběžně vytvořil, do binárního souboru. Většinou se tedy rozsáhlé formáty (jakým je například LATEX) načtou v textovém tvaru TEXem pouze jednou (tak zvaná inicializace) a vytvoří se příslušný binární soubor. Při dalším opakovaném spouštění TEXu se předloží formát ke čtení v této binární podobě. To podstatně urychlí běh TEXu. Na druhé straně styly se vždy předkládají TEXu v textové podobě.

Pokud se jedná o vytváření binárních souborů TEXem, jedná se o specialitu verze pro PC. Autor TEXu to myslel ještě poněkud jinak. Předpokládal, že se spustí jakýsi jiný (ale velmi podobný) program, zvaný iniTEX, který načte formát v textové podobě. Výstupem iniTEXu bude spustitelný program TEX, který v sobě obsahuje datové struktury načteného formátu. O takovém programu říkáme, že má již formát v sobě zavedený (format preloaded). V tuto chvíli už tedy rozumíme hlášení

```
This is emTeX, Version 3.0 [3a] (no format preloaded) .
```

Vraťme se k rozhodování nad tím, který formát zvolit pro svou práci. Pokud zajdete za někým, kdo používá LATEX, řekne Vám, že nejlepší je LATEX. Pokud budete mluvit s uživatelem $\mathcal{AMS}$-TEXu, bude zase tvrdit, že není nad $\mathcal{AMS}$-TEX. Ti, kteří používají NFSS jej také budou prosazovat. Tento článek je například napsán v plainTEXu bez použití NFSS. Nakonec záleží hodně na tom, jaký dokument se chystáte psát. Budete-li přispívat do Americké matematické společnosti, případně Váš text obsahuje plno složitých matematických formulí, použijete $\mathcal{AMS}$-TEX. Budete-li psát nějaký technický dokument (například diplomku), může být vhodný LATEX. Chystáte-li se vydat několikadílný cizojazyčný slovník s „plovoucím záhlavím“ podle hesel, či encyklopedii, budete potřebovat odborníka na TEX, který Vám pro takovou práci napíše formát přímo „na míru“.