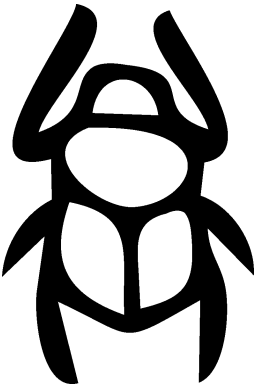
TEX METAFONT



1–4

ZPRAVODAJ 95

ČESKOSLOVENSKÉHO SDRUŽENÍ UŽIVATELŮ TEXU

OBSAH

LaTeX a hieroglyfické písmo starověkého Egypta

Luděk Berec

Hieroglyfické písmo starověkého Egypta

Hieroglyfické písmo je výsledkem dlouhého vývoje písma používaného obyvateli starověkého Egypta.[1] Jeho název pochází z řečtiny a znamená *posvátné písmo tesané*; Řekové, kteří mu dali toto jméno, se s ním totiž seznámili z nápisů vytesaných na staroegyptských chrámech. Znalost tohoto dnes stále více uchvacujícího písma však byla omezena jen na úzký okruh vyvolených (učenců a kněží), kteří jediní tak dokázali záhadné obrázky přečíst. Přesto však jeho užívání pokrývá období přibližně od roku 3 000 př.n.l. až do křesťanské éry Egypta. Poslední známý hieroglyfický nápis je datován do roku 394 n.l. Zpočátku se hieroglyfické písmo používalo pro všechny účely (kterých však v té době nebylo příliš mnoho), později ale, kdy grafická podoba staroegyptského písma směřovala ke zjednodušování používaných hieroglyfických znaků (hieroglyfů) a k vzniku *hieratického* a později také *démotického písma* (viz např. [4]), zejména k monumentálním nápisům na zdech chrámů egyptských bohů či v hrobkách vyšších vrstev obyvatelstva. Za jeho bájného tvůrce byl považován staroegyptský bůh moudrosti Thovt, vlastním bohem písma byla bohyně Sešat.

Po svém zániku představovalo hieroglyfické písmo dlouho neproniknutelnou záhadu a jeho tajemství po celá staletí fascinovalo a přitahovalo nejen vědce celého světa. V této době je však neuměl žádný žijící člověk přečíst. Teprve v minulém století se takový člověk našel. Byl jím Francouz Jean F. Champollion, který dokázal poselství egyptských bohů přečíst a jejich tajemství tak otevřít komukoliv, kdo o něj projeví zájem. Datum 14. září 1822 se tak zapsalo do historie lidského úsilí poznat svou minulost velkým písmem. Samotný Champollionův objev podstaty tohoto písma však ještě neznamenal odhalení všech souvislostí. Navázali však na něj další badatelé, zejména filologové, a jen díky jim je toho dnes o staroegyptském hieroglyfickém písmu tolik známo. Na základech, které

[1] Výsledkem, co se týče jeho struktury – grafický vývoj totiž pokračoval ještě dále.

tento Francouz položil, byla vytvořena nová věda *egyptologie*, zabývající se dějinami Egypta.

LaTeX a hieroglyfické písmo: programový balík Serge Rosmorduca

Zdá se mi, že lidé, kteří dnes používají TeX na velmi pokročilé úrovni, se předhánějí v tom, co ještě nového by do něj mohli přinést. Vznikají tak stále nové fonty a balíky maker. Je téměř jisté, že se staroegyptské hieroglyfy nemohly tomuto trendu dlouho vyhýbat. Tím šílencem (samozřejmě v tom dobrém slova smyslu), který tuto myšlenku uvedl v život, je Francouz Serge Rosmorduc[2] (připomeňme, že ten, kdo hieroglyfické písmo rozluštil, byl také Francouz – snad je práce s egyptskými hieroglyfy součástí jakési francouzské kulturní tradice).

TeXovský balík Serge Rosmorduca, nazvaný *Sesh Nesout*, umožňuje komukoliv sázet v LaTeXu samostatné hieroglyfy, celé hieroglyfické texty, jejich transliteraci[3] i následné překlady.

Tento balík je free software (jeho distribuce a modifikace se řídí podmínkami GNU General Public Licence publikované Free Software Foundation, která je součástí balíku) a lze jej, mimo jiné, nalézt ve všech CTAN archívech. Ty jsou dostupné přes anonymní ftp službu; např. na `ftp.tex.ac.uk` je v adresáři `/CTAN/tex-archive/fonts/hieroglyph`.

Konečným cílem Rosmorducovy snahy je vytvořit kompletní, snadno použitelný a dostatečně obecný (programový) nástroj, který by umožňoval vystavět databázi staroegyptských hieroglyfických textů včetně uchování jejich obsahu. Zatím se však, jak sám tvrdí, nachází poměrně daleko od tohoto vytčeného cíle. I v této fázi však mohou být jeho výsledky užitečné. Co je tedy zatím k dispozici?

Sázení hieroglyfických textů

K tomu, abychom mohli sázet hieroglyfické texty, potřebujeme tři věci: hieroglyfické fonty, soubor `hiero.sty` a konečně program `sesh`, fungující jako jakýsi preprocesor vytvořených zdrojových souborů.

Hieroglyfické fonty

Hieroglyfické fonty jsou k dispozici ve formě *fontname*`.mf`. Egyptologové rozdělují hieroglyfické znaky do celkem 26 skupin podle jejich typu (mužské postavy, ptáci, lodě a jejich části apod.). Stejné dělení je

[2] Je však možné, že existují i jiní.

[3] Zápis hieroglyfických textů pomocí poněkud rozšířené latinky je podrobněji vysvětlen v části věnované stylu `hiero`.

zachováno i zde. Každému fontu odpovídá jeden typ hieroglyfů, který si můžeme zobrazit například pomocí souboru `testfont.tex` (v daném fontu ovšem nejsou naprosto všechny objevené znaky příslušející dané skupině, neboť zdaleka ne všechny jsou jasné egyptologům, natož nám, laikům).

Hieroglyfickým písmem se psalo a četlo v řádcích nebo sloupcích, převážně zprava doleva, jen zřídka zleva doprava, a shora dolů. Proto obsahuje každý font příslušné hieroglyfické znaky v obou směrech. Směr čtení poznáme tak, že znaky mající přední a zadní část (např. )[4] se dívají k začátku textu, ve kterém se vyskytují (výjimka může být při zápisu do sloupců; týká se však jen jejich pořadí, nikoli čtení znaků uvnitř jednotlivých sloupců). Tato pravidla zůstávají v platnosti i tehdy, pokud se vyskytují znaky nad sebou v rámci jednoho řádku či vedle sebe v rámci jednoho sloupce. Egypťané se totiž snažili psát bez nevzhledných děr v textu, a tak se tímto způsobem psalo dost často. Některé znaky se pak kvůli úpravě pochopitelně psaly poněkud menší. Zejména z tohoto důvodu jsou zde stejné fonty instalovány v už zmíněných několika velikostech.

Pro základní rozlišení 300 dpi je možno přímo na CTANu získat už hotové fonty *fontname*`.pk` a příslušné metriky *fontname*`.tfm` v používaných velikostech, pro ostatní rozlišení je nutné tyto soubory pomocí METAFONTu vygenerovat. Kromě už zmíněných 26 fontů jsou zde ještě diakritické fonty pro transliteraci a font pro sázení speciálních znaků (např. *kartuší* – oválných rámečků, do kterých se zapisovala některá králova jména[5] –).

Styl `hiero.sty` *a program* `sesh`

Nejdříve si vysvětleme, co rozumíme pod pojmem *transliterace*. K tomu, abychom mohli posoudit původní zvukovou hodnotu staroegyptského jazyka, potřebujeme hieroglyfický text přepsat do podoby, kterou můžeme foneticky přečíst. Egyptologové proto přepisují staroegyptské texty do poněkud rozšířené latinky, která tuto zvukovou podobu vyjadřuje (třetí sloupec tabulky v příloze A zobrazující „abecední“ hieroglyfické znaky a jejich vlastnosti). Tento přepis se nazývá transliterace

[4] Velikost znaků můžeme řídit několika příkazy (viz [2]).

[5] Král měl v nejstarších dobách egyptského státu tři jména, později pak pět. Z těch se pak čtvrté a páté psalo do těchto kartuší.

a takto rozšířená latinka je fyzicky uložena v několika velikostech ve zmíněných diakritických fontech.

K transformaci vstupního kódovaného tvaru sázeného hieroglyfického textu do jeho LaTeXovské podoby v podstatě slouží program `sesh`. Text, který má být na výstupu hieroglyfický, vložíme ve speciálním formátu (vysvětleno v dalším odstavci) do prostředí `hieroglyph`. Vně tohoto prostředí se řídíme obvyklými pravidly LaTeXu. Pak použijeme na náš dokument program `sesh` jako filtr. Ten vše ponechá původní, jen obsah prostředí `hieroglyph`, kterých může být v textu i více, transformuje do LaTeXovské podoby čitelné stylem `hiero.sty`, a tento výsledný zdrojový text je už připraven k překladu a následnému zobrazení či tisku. `Sesh` je program napsaný v jazyku C a běžící v systému Unix[6]. Čte ze standardního vstupu – obvykle klávesnice a zapisuje na standardní výstup – obvykle monitor. Pro čtení ze souboru a pro zápis do souboru je tedy třeba vstup a výstup přesměrovat. Máme-li např. původní text dokumentu v souboru `test.ltx`, program `sesh` jej načte a LaTeXovský výstup přesměruje do `test.tex` pomocí příkazu `sesh < test.ltx > test.tex`.

Jak vlastně vypadá obsah prostředí `hieroglyph`? Hieroglyfy jsou zapisovány buď pomocí kódované transliterace (kombinace znaků pátého sloupce tabulky v příloze A, nebo Gardinerova kódu `XY`, kde `X` znamená typ hieroglyfické skupiny, který odpovídá názvu fontu obsahujícího daný znak (`A-I`, `K-Z`, `Aa`), `Y` kód znaku v rámci této skupiny. Kompletní seznam hieroglyfických znaků spolu s jejich transliterací (pokud ji mají) a Gardinerovým kódem je např. v [1] a [4]. Podobný seznam pro znaky používané v tomto balíku je v [2].

Kromě těchto kódů se v prostředí `hieroglyph` používají také speciální znaky, jež podporují práci s hieroglyfy v rámci hieroglyfického textu či kódují speciální symboly používané kdysi Egypťany nebo dnes egyptology k zdůraznění některých aspektů textu. Např. hieroglyfický text umístěný mezi znaky `<-,->` je zapsán v kartuši, text mezi znaky `-#-,-#-` je zašrafován (zdůrazňuje nečitelnost v původním textu), mezi jednotlivými slovy je možno psát mezeru, kde se pak TeX pokouší lámat řádky, „`-`“ slouží jako oddělovač dvou znaků na stejné úrovni apod. Seznam všech používaných speciálních znaků je uveden v [2].

[6] K dispozici je ve formě zdrojových souborů a souboru `Makefile`. Je však možné, že po drobné úpravě poběží i pod DOSem. Nejsem však C-expert, a tak jsem se tímto problémem nezabýval.

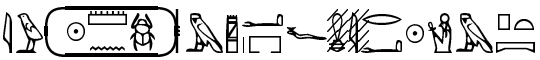
Kódování hieroglyfů, transliterace a naprosté většiny speciálních symbolů není autorův výmysl. Je to mezinárodní standard vyvinutý pro kódování staroegyptských textů do počítače (viz [1]). Snahou Serge Rosmorduca je tento standard plně implementovat, popřípadě přidat některé další myšlenky pro usnadnění práce s hieroglyfickými texty.
Příklad:

```
\begin{hieroglyph}
i-w- <-ra-mn:n-xpr-> -m-aH-a:Z1*pr-=f- -#-mi-i-#-
r:a-ra-C2 m p*t:pt
\end{hieroglyph}
```

Po `sesh`:[7]

```
\begin{hieroglyph}%
{%
\leavevmode\makeatletter%
\Hunh{\Aca M/17/}\Hrp \Hunh{\Aca G/46/}\Hitmts%
\cartouche{\Hunh{\Aca N/5/}\Hrp \Hbt{\HhbtI{\Aca Y/5/}
\Hhbt{\Aca N/35/}}\Hrp \Hunh{\Aca L/1/}}\Hitmts%
\Hunh{\Aca G/19/}\Hrp \Hunh{\Aca O/11/}\Hrp
\Hbt{\HhbtI{\Aca D/38/}\Hhbt{\Aca Z/1/\hfil\Aca O/1/}}\Hrp
\Hunh{\Aca I/10/}\Hitmts%
\hachure{\Hunh{\Aca W/21/}\Hrp \Hunh{\Aca M/17/}}\Hitmts%
\Hbt{\HhbtI{\Aca D/21/}\Hhbt{\Aca D/38/}}\Hrp
\Hunh{\Aca N/5/}\Hrp \Hunh{\Aca C/2/}\Hitmts%
\Hunh{\Aca G/19/}\Hitmts%
\Hbt{\HhbtI{\Aca Q/3/\hfil\Aca X/1/}\Hhbt{\Aca N/1/}}}%
\end{hieroglyph}
```

Výsledek:



Směr psaní

Standardní sázení hieroglyfů je pochopitelně zleva doprava. Použijeme-li TEX–XETT (verze umožňující sázet v obou směrech včetně výhody

[7] Řádkování je upraveno, aby se tento zdrojový text vešel na textový řádek dokumentu – skutečný řádek končí tam, kde je znak `%`.

lámání řádků), je možné správně obracet celé části hieroglyfického textu pomocí přepínačů `+dg, +gd` použitých v prostředí `hieroglyph`, a simulovat tak psaní zprava doleva. V normálním LaTeXu je obrácení možné pomocí přepínačů `+rl, +lr`, ovšem bez výhody lámání řádků. Je-li však konečný výstup PostScript, můžeme např. využít balík `pstricks` k zrcadlovému obrácení textu psaného zleva doprava.

K psaní do sloupců slouží příkaz

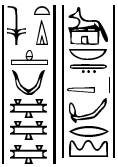
`\EnColonne[šířka sloupce]{prostředí hieroglyph}`

Sloupce se pak čtou zleva doprava a znaky v něm zleva doprava a shora dolů.

Příklad:

```
\EnColonne[1.2\Htm]{
\begin{hieroglyph}
#def wAt N31#
sw*(t:di) -Htp -wp-wAt-wAt-wAt-!
E16-nb-tA:idb\s2*Z1-Dsr-r-xAst-!
\end{hieroglyph}
}
```

Výsledek (centrováno):



Psaní maker

V rámci prostředí `hieroglyph` můžeme příkazem `#def MACRONAME Body#` definovat vlastní makra pro jednu nebo určitou skupinu hieroglyfů.

Příklad:

```
\begin{hieroglyph}
#def king A42#
#def nn M22-M22-n:n#
king nn
\end{hieroglyph}
```

Výsledek:

Může to být užitečné, vyskytuje-li se daná sekvence znaků v textu příliš často, nebo chceme-li některému znaku nebo celé skupině (obvykle slovu) dát své vlastní, více říkající jméno (obvykle jeho význam). Tato makra pak zůstávají v platnosti i ve všech následujících prostředích `hieroglyph` bez nutnosti jejich opětovné definice.

Styl `hierLtx.sty`

V podstatě existuje ještě jeden alternativní styl pro práci s hieroglyfy – `hierLtx.sty` Tento styl definuje jediný příkaz `\hieroglyphe`. Používá se takto:

```
\hieroglyphe{A/1}
```

Zde písmeno udává název hieroglyfického fontu, číslo *pořadí daného znaku v tomto fontu*. Abychom pomocí něj mohli psát i komplikovanější texty, je možné použít oddělovačů „`-`“ k horizontálnímu oddělení znaků nebo jejich skupin (označovaných pomocí složených závorek) a „`:`“ k vertikálnímu oddělení. Například:

```
\hieroglyphe{{Q/3-X/1}:N/1}
```

Více práce s hieroglyfy tento styl nepodporuje. Na jedné straně je sice pravda, že k napsání znaků tímto způsobem nepotřebujeme mít po ruce UNIXovský stroj, ale potřebujeme-li více než jen vložit do textu (ne však do textového řádku) jeden nebo pár znaků, které více než tyto dvě operace nepotřebují, narazíme.

Práce s transliterací – `egypto.sty`

V rámci tohoto programového balíku je také možno pracovat s texty zapsanými pomocí transliterace (viz předchozí části). K tomu slouží jednak transliterační diakritické fonty, jednak styl `egypto.sty`. Transliterační fonty podporují jak NFSS2, tak starý systém (nepodporují NFSS). Je několik možností jak takové texty sázet a pracovat s nimi:

1. Příkaz `\eg` – přepíná přímo do transliteračního fontu podobně jako například `\bf` do tučného písma.

2. Prostředí `translit` – vytváří minidokument obsahující transliterovaný text (automaticky přepíná do `\eg`), nadepsaný zadanou hlavičkou, jejíž prvky se vkládají jako argumenty tohoto prostředí:[8]

```
\begin{translit}{zdroj egyptského textu}
                {číslo strany nebo recto/verso}
                {čísla řádků}
                {alternativní reference nebo komentář}
```

LaTeX těmto číslům řádků a stran rozumí a má k dispozici příkazy umožňující s těmito čísly uvnitř prostředí pracovat (zvyšovat je, měnit, zobrazovat), což je velmi užitečné pro případné křížové reference. Jejich popis je opět v [2].
Uvnitř prostředí můžeme zapsaný text i přímo překládat – příkaz `\traduction` přepne do italiky.
Příklad:

```
\begin{translit}{Svitek č. XXI}{verso}{10-15}{Pohádka}
\affpage \\
\affligne\ iw.i rx.kw mdw nTr \\
\| xpr XSH
\traduction
  Zde může následovat překlad.
\end{translit}
```

Výsledek: SVITEK Č. XXI, verso 10–15 POHÁDKA

(verso)

10 | *iw.i rḫ.kw mdw nṯr*

11 | *ḫpr ḫšḫ*

Zde může následovat překlad.

3. Prostředí `exemple` – vše provádí a sází stejně jako v prostředí `translit` a navíc přidá na začátek hlavičky prostředí slovo „Exemple“ a přiřadí mu pořadové číslo v rámci dokumentu. Kdybychom chtěli tisknout „Příklad“, musíme v preambuli užít definici `\def\EXEMPLE{Příklad}`.

[8] `Recto` je *lícová strana* papyrového svitku (tj. ta strana, kde je vrstva plátků z papyrové dřeně uspořádána *vodorovně*), `verso` pak jeho rub (*svislé* uspořádání vláken).

4. Příkaz `\traduction{}{}` – slouží k on-line překladu. První argument je transliterace, druhý pak překlad. Můžeme přepínat (viz [2]), zda chceme mít text a překlad ve dvou sloupcích či pod sebou. V případě dvou sloupců můžeme přejít na další stranu a případné poznámky pod čarou jsou odděleny pro každý sloupec zvlášť. Jako default je text sázen do dvou sloupců, přičemž překlad (pravý sloupec) je sázen v prostředí `verse`. Tento způsob je tedy velmi výhodný pro zápis a následný překlad staroegyptské poezie.

Soubor `*.dic`

Soubor `egypto.sty` vytváří během překladu zdrojového textu *jobname*`.tex` externí soubor *jobname*`.dic`, do kterého zapisuje tři druhy informací:

1. Použijeme-li ve zdrojovém souboru prostředí `translit`, zapíše se do souboru *jobname*`.dic` řádek

```
\Citation{zdroj egyptského textu}
         {reference (číslo strany,řádku)}
         {alternativní reference nebo komentář}
         {číslo strany dokumentu}
```

   (všechny parametry na jeden řádek). Pro výše uvedený příklad zapíše

```
\Citation{Svitek č. XXI}{verso,10}{Pohádka}{5}
```

2. Použijeme-li ve zdrojovém souboru prostředí `exemple`, zapíše se do souboru *jobname*`.dic` řádek

```
\Exemple{zdroj egyptského textu}{reference}
        {alternativní reference nebo komentář}
        {číslo příkladu v dokumentu}
        {číslo strany dokumentu}
```

3. K použití v prostředích `translit` a `exemple` je navíc definován příkaz

```
\dico{kód transliterace}{překlad}{komentář}
```

Ten je vhodný pro vytvoření indexu staroegyptských výrazů. Je-li tento příkaz použit, připisuje se do souboru *jobname*`.dic` řádek

```
\DicoIndex{kód transliterace}{překlad}{komentář}
          {zdroj egyptského textu}{reference}
          {název dokumentu}
          [{číslo příkladu v dokumentu}]
          {číslo strany dokumentu}
```

tedy parametry příkazu `\dico` a navíc reference použité v příkazech `\Citation` a `\Exemple`, jen šestý parametr zde není čtvrtým parametrem příslušného prostředí, definovaným v jejich záhlaví, nýbrž parametrem definovaným vně těchto prostředí příkazem `\Montitre{název dokumentu}`. První argument příkazu `\dico` může obsahovat prostředí `hieroglyph`.

Gramatická pravidla

Můžeme také sázet přehledná gramatická pravidla a vytvořit si tak svou vlastní minimluvnici egyptského jazyka s hieroglyfickými příklady. K tomu slouží prostředí `gramrule`. Slova jsou v něm zobrazována v `\sl` fontu a sekvence `~X␣` zapisuje `X` v transliteraci (mezera je nutná). V tomto prostředí můžeme užít prostředí `possib` (k sázení několika možností se složenou závorkou před nimi – `\\` odděluje jednotlivé případy) či `pile` (k psaní textu do sloupce – `\\` odděluje jednotlivé řádky ve sloupci). Prázdný objekt $\emptyset$můžeme zapsat pomocí příkazu `\zero`.
Příklad:

```
\begin{gramrule}
~ir + \begin{possib}
       infinitiv\\
       prospektiv ~sDm.f \\
       ~mrr.f \\
       \end{possib}
    + \pile{cokoli\\chcete\\}
 \end{gramrule}
```

Výsledek:

$$\textit{ir} + \begin{cases} \textit{infinitiv} \\ \textit{prospektiv } \textit{s}\underline{\textit{d}}\textit{m.f} \\ \textit{mrr.f} \end{cases} + \begin{matrix} \textit{cokoli} \\ \textit{chcete} \end{matrix}$$

Závěr

Tento článek si nekladl za cíl podrobně popsat stávající verzi programového balíku Serge Rosmorduca pro sázení hieroglyfických textů v LaTeXu. Jeho detailní popis je možno najít v manuálu [2] dodávaném spolu s tímto balíkem jako LaTeXovský soubor. Chtěl jen upozornit na zajímavou možnost využití LaTeXu a seznámit s tímto balíkem na pozadí základních informací o staroegyptském hieroglyfickém písmu.

Zajisté lze na tomto balíku ještě spoustu věcí vylepšit a přidat další myšlenky. Ten je také teprve základním krokem k vytvoření kompletního, snadno použitelného a dostatečně obecného programu, který by sloužil k sestavování databází staroegyptských hieroglyfických textů včetně uchování jejich obsahu. Tento cíl si totiž Serge Rosmorduc vytkl. V jeho dalším, jistě nelehkém úsilí mu také touto cestou přeji hodně štěstí a trpělivosti.

Nicméně i v této fázi jsou jeho výsledky zajímavé a k některým účelům určitě použitelné. Já osobně jeho balík používám například k sestavování egyptsko-českého slovníčku při studiu staroegyptského jazyka. Přeji proto i vám, všem čtenářům, hodně zábavy při hře s hieroglyfickými znaky.

Literatura

[1] Jan Buurman, Nicolas Grimal, Michael Hainsworth, Jochen Hallof, and Dirk Van Der Plas. *Inventaire des signes hieroglyphiques en vue de leur saisie informatique.* Mémoires de l'Académie des Inscriptions et Belles Lettres. Institut de France, Paris, 1988.

[2] Serge Rosmorduc: *LaTeXperiment of hieroglyphic typesetting.* Manuál pro TeXovský balík *Sesh Nesout*, který je s ním dodáván jako LaTeXovský soubor.

[3] Viktor Dračuk: *Svědkové tisíciletí.* Lidové nakladatelství, Praha, 1985.

[4] Alan Gardiner: *Egyptian Grammar.*

Luděk Berec
berec@utia.cas.cz

„Abeceda“, transliterace, kódování

znak	Gardinerův kód	kód podle fontu	transliterace	kód transliterace	česká výslovnost
	G1	G/1	*ȝ*	A	*a (alif)*
	M17	M/17	*i̯*	i	*í*
	D36	D/38	*ʿ*	a	*a (ajn)*
	G43	G/46	*w*	w	*u (anglické w)*
	Z7	Z/9	*w*	W	*u – pozdější alternativní forma* 
	D58	D/61	*b*	b	*b*
	Q3	Q/3	*p*	p	*p*
	I9	I/10	*f*	f	*f*
	G17	G/19	*m*	m	*m*
	Aa15	Aa/14	*m*	M	*m – pozdější alternativní forma* 
	N35	N/35	*n*	n	*n*
	S3	S/3	*n*	N	*n – pozdější alternativní forma* 
	D21	D/21	*r*	r	*r*
	E23	E/24	*l*	l	*l – vznikl později*
	O4	O/4	*h*	h	*h*
	V28	V/28	*ḥ*	H	*h (emfatické h)*
	Aa1	Aa/1	*ḫ*	x	*ch*
	F32	F/32	*ẖ*	X	*ch,š*
	O34	O/34	*s*	s	*s (původně z)*
	S29	S/32	*s*	s	*s*
	N37	N/38	*š*	S	*š*
	N29	N/29	*ḳ*	q	*k (zadní k)*
	V31	V/31	*k*	k	*k*
	W11	W/13	*g*	g	*g*
	X1	X/1	*t*	t	*t*
	V13	V/13	*ṯ*	T	*c,č*
	D46	D/48	*d*	d	*d*
	I10	I/11	*ḏ*	D	*dž*

Drobná typografická poznámka ke spojovníku

Jiří Kosek ml.

Spojovací čárka neboli spojovník je grafický znak v podobě vodorovné čárky kladené bez mezer mezi slova nebo jejich části. Užívá se, chceme-li vyjádřit, že jím spojené výrazy tvoří těsný (slovní nebo souslovný) celek. Spojovník se graficky i funkčně odlišuje od pomlčky.

Toliko praví na téma spojovníku diskutovaná Pravidla českého pravopisu z roku 1993. Každý uživatel TEXu jistě ví, jaký je rozdíl mezi spojovníkem a pomlčkou. K sazbě spojovníku používáme ‚`-`', kdežto k sazbě pomlčky ‚`--`' nebo ‚`---`'. Rád bych se však zmínil o jedné drobnosti, kterou „Pravidla" nezmiňují. Je to dělení slov v místě, kde obsahují spojovník. Dle ustálených českých typografických zvyklostí se v tomto případě spojovník opakuje i na začátku následující řádky — odliší se tak sousloví obsahující spojovník od běžně rozdělených slov. Ukázku správného dělení v takovýchto případech přináší následující odstavec, formátovaný pro různé šířky tiskových zrcadel:

Drahý příteli, víte-li, jak se mají správně dělit slova, jež obsahují spojovník. Např. pověstná ASCII-tabulka či onen výbušný propan-butan.

Drahý příteli, víte-li, jak se mají správně dělit slova, jež obsahují spojovník. Např. pověstná ASCII-tabulka či onen výbušný propan-butan.

Jak vidíte, v oprávněných případech se spojovník objevil i na začátku řádky. Bohužel, touto vlastností sám o sobě TEX nedisponuje. Napíšete-li do textu `víte-li`, můžete jenom snít o tom, že se vám spojovník objeví i na začátku řádky.

Naštěstí díky flexibilitě TEXu existuje poměrně jednoduché a elegantní řešení tohoto problému. Zkušenější TEXista jistě tuší, že existuje primitiv:

`\discretionary{`⟨pre-break text⟩`}{`⟨post-break text⟩`}{`⟨no-break text⟩`}`

V místě kam tento primitiv vložíte do vstupního souboru obvykle TEX vloží poslední variantu textu (⟨no-break text⟩). Pokud je však v místě, kde je uvedeno `\discretionary` potřeba provést řádkový zlom, je na konec řádky umístěn ⟨pre-break text⟩ a na začátek následující řádky je umístěn ⟨post-break text⟩. Standardní TEXovský spojovník se tedy chová tak, jako kdyby na jeho místo bylo vkládáno `\discretionary{-}{}{-}`. Aby dělení proběhlo dle našich národních zvyklostí, bylo by třeba, aby se spojovník choval jako `\discretionary{-}{-}{-}`.

První řešení, které se nabízí, je změnit spojovník na aktivní znak (category-code 13) a předefinovat jej výše zmíněným způsobem. Toto řešení je však neschůdné, neboť znak používaný pro spojovník ‚`-`‘ je používán i mnohde jinde (v číselných konstantách `\vskip-3pt`, nebo jako ligatura ‚`--`‘ a ‚`---`‘).

Pokud se tedy rozloučíme s myšlenkou, že půjde předefinovat samotný znak ‚`-`‘, musíme najít nějaký jiný vhodný znak, jenž budeme používat pro zápis spojovníku. (Definování makra s názvem `\spojovnik` asi totiž nebude tím pravým ořechovým.) Na onen znak jsou kladeny v podstatě tři požadavky:

1. Znak by neměl být již v TEXu používán pro nějaký jiný účel z důvodu zachování kompatibility.
2. Znak by měl co nejvíce opticky připomínat ‚`-`‘, aby byl zdrojový text čitelný a srozumitelný.
3. Znak by mělo jít nějak snadno vykouzlit na klávesnici. (Řešení využívající *Mapu znaků* a clipboard ve Windows budeme považovat za nevyhovující.)

Když se po takovém znaku porozhlédneme v první polovině ASCII-tabulky, budeme zklamáni. Jediný znak, který nalezneme, je ‚`_`‘ a ten je navíc používán v makru `\_` pro sazbu podtržítka. Naštěstí umožňuje TEX od verze 3.0 práci s 256 znaky a můžeme tedy naše pátrání posunout i na znaky s kódy 128–255. Tato oblast je však poněkud problematická, existuje pro ní několik naprosto rozdílných kódování. Následující úvahy se budou opírat o následující dva druhy: kódování bratří Kamenických a Latin 2. Omlouvám se všem uživatelům UNIXu, kteří používají ISO Latin 2. Toto kódování bohužel detailně neznám, ale přesto doufám, že v něm vhodný znak naleznou a provedou obdobné úpravy jako my.

Snad jediné, co mají Kameničtí společné s Latin 2, jsou znaky pro jednoduché rámečky. Když je podrobněji prostudujeme zjistíme, že znak pro vodorovnou čáru je velmi podobný normálnímu spojovníku. Tento znak má ASCII-kód 196 a graficky jej budeme znázorňovat takto: ‚—‘.

Prvním krokem bude přinutit editor, aby v něm šlo tento znak snadno napsat. Samozřejmě existuje možnost stisknout `Alt+196`, ale jistě není tou optimální. Využijeme schopnosti většiny editorů — možnost tvorby maker. Makro bude obsahovat jediný znak — náš znak s kódem 196. Makro by mělo být přiřazeno nějaké klávese, která by odpovídala původnímu spojovníku. Jako nejvhodnější se jeví kombinace `Ctrl+-` nebo `Alt+-`.

Pokud používáte editor QEdit, můžete tuto úpravu provést tak, že do souboru s definicí kláves `qconfig.dat`, přidáte následující řádek:

```
^-   MacroBegin '—'
```

případně (pro kombinaci s klávesou `Alt`):

```
@-   MacroBegin '—'
```

Znak mezi apostrofy je samozřejmě znak s ASCII-kódem 196. Pro korektní nakonfigurování QEditu je ještě potřeba nově upravený soubor s definicí kláves zakompilovat do editoru. To provedeme programem `qconfig.exe` a postupným zvolením `K` a `S`.

Poslední, co zbývá, je přinutit TEX, aby na místa obsahující „—“ vložil správné `\discretionary`. Toho dosáhneme následující definicí (je vhodné ji umístit na začátek souboru s dokumentem nebo do souboru s makry, který pravidelně inputujeme):

```
\catcode`\—=13 \def—{\discretionary{-}{-}{-}}
```

Nyní je již jen na vás, abyste si zvykli na správných místech místo „-“ zadávat „—“. Doufám, že tento skromný příspěvek pozvedne alespoň o 1 mm úroveň dokumentů připravovaných v TEXu. Rád bych poděkoval Štěpánu Kasalovi, s kterým jsme na tento problém narazili na zastávce tramvají číslo 6, 9 a 12. Už jen díky tomu, že jsem se pak donutil nastudovat něco o `\catcode`, bez jehož použití by šlo opravdu nadefinovat jen makro s názvem `\spojovnik` nebo jemu podobným.

Jiří Kosek ml.
XKOSJ06@VSE.CZ

Po tom, co jsem článek dopsal, jsem při používání zmíněných technik narazil na jeden drobný problém — znak „-“ se netiskl tak, jak by měl (resp. se netiskl vůbec). Jak jsem zjistil, bylo to způsobeno drobnou nesrovnalostí v tcp-tabulce, která je součástí CS-TEXu. Na znak s kódem 196 (v CS-fontech přehlasované A) se mapuje jak přehlasované A, tak i znak „—“. Asi nejlepší způsob, který pro řešení tohoto problému existuje, navrhl pan Olšák. Mapovat znak „—“ na pozici 156 (9ch). Na této

pozici je v CS-fontech alternativní hyphenchar. Použijete-li ve vstupním souboru ‚—‘ bez definice, dostanete obyčejný spojovník (nedělitelný). Po použití výše zmíněné definice, dosáhnete kýženého efektu opakování spojovníku na začátku řádky.

Přemapování znaků v tcp-tabulce můžete nejlépe provést pomocí programu `maketcp`. Pokud jste na tom jako já a tento program nemáte momentálně po ruce, můžete provést změny přímo v souboru `kamenic.tcp` nebo `pclatin2.tcp` (fuj, to je ale ošklivý způsob). Obsah bytu na offsetu c8h (počítáno od nuly) změňte na hodnotu c4h; obsah bytu na offsetu 1c8h změňte na hodnotu 9ch. Nezapomeňte po provedení změn v tcp-tabulce znovu vygenerovat formát, aby začaly změny účinkovat. (V CS-TEXu slouží ke generování formátu dávka `initex` uložená v adresáři `BATS`.)

Lunisolární výpočty v TEXu

Bc. Stanislav Brabec

Tento článek patří svým zaměřením spíš k TEXovým kuriozitám. Chce ukázat, co vše lze v TEXu dělat a jak neuvěřitelná makra lze psát. K napsání mě inspirovala kniha [1], zvláště pak její kapitola o „bílých trpaslících“ a „introvertech“ mezi programy, kde byl jako příklad popsán Gaussův algoritmus pro výpočet data velikonoc. A protože jsem též přečetl známý článek [2] a z valné části i knihu [3] s krásnou ukázkou Erastothenova síta, došel jsem k závěru, že nejkrásnější ukázku tohoto typu programu je možné připravit v TEXu. Po jistém úsilí vznikl program, jenž vám předkládám. Na první pohled připomíná spíš poruchy na lince, nežli makro v TEXu, ale skutečně funguje, jak jistě každý nahlédne (tj. vyzkouší a uvěří výsledku). Komentáře nejsou vloženy (viz [2]: Opravdový programátor nepotřebuje poznámky – vlastní kód je zřejmý.). Navíc se domnívám, že komentář by stejně příliš nepomohl. Tak alespoň přikládám podobný program v jazyce C (ten však umí pouze česky, neboť standardní ošetření formátovaného výstupu nezná přehození parametrů).

Velikonoce patří k starobylým svátkům. Již pohané slavili příchod jara o prvojarní úplňkové noci (z těchto dob pochází i název вєлико ноцє). Křesťanská kultura umístila do těchto pohyblivých svátků oslavu Kristova zmrtvýchvstání. Tím došlo k posunu tohoto svátku na následující neděli. A tak nám zde naši předkové připravili matematický rébus – co nejjednodušeji vypočíst datum velikonoc. Jeden z největších matematických géniů – Carl Friedrich Gauß nám zde zanechal pravděpodobně nepřekonatelně jednoduchý algoritmus, díky němuž mohl vzniknout i tento článek.

Co se týče vlastního Gaussova algoritmu, žel neznám nikoho, kdo by ho pochopil a uměl rozumně vysvětlit. Kromě toho se domnívám, že důkaz jeho správnosti by vyžadoval (vzhledem k rozsahu tohoto bulletinu) několik jeho ročníků.

Lunisolární algoritmy lze napsat v TeXu samozřejmě i přehledněji. Příklad nalezneme v souboru `HEBCAL.STY` od Michaila Rozmana a Ramy Porratové, jenž je součástí hebrejského TeXu. Rozdíl délky souborů je však zřejmý (stonásobný).

Ještě malé upozornění: program by (alespoň dle mého úsudku) měl správně fungovat pouze pro data mezi roky 1900 a 2099. Implementace pro ostatní léta je též možná. Pro zájemce mám připravený věčný kalendář pro TeX, jenž náležitě ošetřuje i juliánský kalendář. Bude-li někdo v TeXu sázet třeba efemeridy, nic mu nebrání je v TeXu i počítat. Nakonec, jaký jiný typografický systém by něco podobného dokázal?

Použití je triviální: Do konzole napíšete „`csplain \year=`rok `\language=`jazyk `\input Easter`“ a program vytvoří `DVI` soubor s datem Velikonoc. Vynecháte-li zadání roku, počítá se letos, vynecháte-li zadání jazyka, bude to anglicky.
Příklad: „`csplain \year=1968 \language=\czech \input Easter`“.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                               %
%          TeX algorithm to compute an Easter date              %
%         ========================================              %
%                                                               %
% File: Easter.tex v 1.5 95-04-18 12:20:00                      %
% Author: (c) 1995 Stanislav Brabec utx@k332.feld.cvut.cz       %
%                                                               %
% Copyright: Easter.tex is freely distributable as a sample of  %
%            compact astronomic computations in plain TeX       %
%                                                               %
% Usage:                                                        %
```

```
% virtex [&fmt] [\year[=]year] [\language[=]\xxx] \input Easter %
%  (writes result to Easter.dvi, not console, sorry)            %
%                                                               %
% Years available: 1900-2099                                    %
% Languages: czech, slovak, german, none specified = english    %
%       You can simply add any other.                           %
% Requires language token defined in format (from hyphen.lan)   %
%                                                               %
% Based on Easter algorithm of Carl Freidrich Gauss.            %
% Warning: Trying to understand the algorithm can take a very   %
%       long time and can cause a headache! Don't try it.       %
% ... (And if you find a shorter algorithm, let me know!)       %
% Note: Single-language algorithm can be even much shorter.     %
%                                                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

\def~#1{\catcode`#113}~:\let:\let~+:+\advance~`:`\year~$
:$\newcount~^:^\def~[:[\ifnum~?:?\the~]:]\fi~*:*\multiply
~/~'~(~)~;$($'$)$;^/#1#2{(#1\divide(#2*(-#2+#1(}'`/'{19}*'19
+'24/'{30})5;`/;4*;2+);;`/;7*;4+);;'*;6+);/)7+)'+)21
[)>55+)-7](31^:{; }^'{+)1 [)>(+)-(^:{/ }]}'^*{}[)=(^*{:}]^~{?)}
^$#1#2#3#4{\ifx#1\undefined\else[\language=#1^;{#2}^/{#3}#4]]}
$0{March}{April}{Easter falls on :~.\ and '*~., ?`.}
$\czech{března}{dubna}{Velikonoce připadají na ~.\ *'a ~.\ :?`.}
$\slovak{marca}{apríla}{Veľkonočné sviatky pripadajú na ~.\ *'a ~.\ :?`.}
$\german{März}{April}{Die Ostern fiel sein am ~.\ *'und ~.\ :?`.}
\end
```

Program v jazyce C:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

char x[32],y[32];
struct tm *curtime;
char *day(char *x,const d,const short om)
{
sprintf(x,"%i. %s",d>31?d-31:d,(d>31&&!om?"dubna "
           :(d==31)||!om?"března ":""));
return(x);
}
int main(int argc,char *argv[])
{
int d,r;
time_t t;
if (argc==2) r=atoi(argv[1]); else {
  time(&t);
  curtime=gmtime(&t);
  r=1900+curtime->tm_year;
```

```
}
if ((d=22+(d=(19*(r%19)+24)%30)+(5+2*(r%4)+4*(r%7)+6*d)%7)>56) d=d-7;
printf("Velikonoce připadají na %sa %s%i.\n",day(x,d,1),day(y,d+1,0),r);
return(0);
}
```

Seznam literatury:

[1] Ivan Kopeček, Jan Kučera: *Programátorské poklesky*, Mladá Fronta, Praha 1989

[2] Ed Post: *Real Programmers Don't Use Pascal*, Datamation, July 1983, pp. 263–265 (Readers' Forum).

[3] D. E. Knuth: *The TEX Book*, Addison-Wesley Publishing, 1991

[4] Kalendáře na roky 1995, 1996, 1990 a 1991

Addendum:

Předminulý týden se mi dostal do ruky další geniální příklad z panoptika programů v [1] – program „želva“. Je jím program `wp2latex`. Na mém počítači (bez matematického koprocesoru) tráví přibližně 95 % veškerého času vypisováním toho, kolik procent je již převedeno (během převodu 100 kB souboru vyšle do konzole desítky až stovky tisíc znaků (polovina jsou znaky `BS`)). 3 % času tráví výpočtem toho, kolik procent je již převedeno (v aritmetice s dvojnásobnou přesností), zbylé 2 % času pak tráví převodem – to jest neustálým vyhledáváním znaků v naprosto neefektivně organizované tabulce. Takový dokonalý příklad „želvy“ nevymysleli ani autoři [1].

To je překrásné využití TEXu; pokud si ale pro začátek chcete TEXovskou aritmetiku vyzkoušet na něčem jednodušším, leč stejně praktickém, zkuste sestavit makro s jedním parametrem `#1`, které posune aktuální datum o `#1` dní, tedy např. `\advancedate{14}` udělá v roce 1995 z 26. února 12. března, ale následující rok správně 11. března (jedno z možných řešení najdete na konci tohoto čísla).

Karel Horák

Knižní úprava pro uživatele TEXu. Část první: Teorie

Philip Taylor

Abstrakt: Knižní úpravě nelze nikoho naučit; to musí udělat každý sám, nejlépe kritickým studiem co největšího množství knih. Ze všech prvků, které dávají dohromady knihu, je prázdný prostor tím nejčastěji opomíjeným a zároveň nejdůležitějším. V článku jsou srovnávány *avantgardní* přístupy k designu s těmi konzervativnějšími a tradičními. Definujeme tři klíčové prvky — *jednotnost*, *informace* a *struktura* — a v rámci těchto prvků je probírána „správná návrhářská praxe“.

Klíčová slova: Design, typografie, grafická úprava

Motto: Pod titulky nikdy nemůže být příliš málo místa, pouze příliš!

1 Úvod

Široké užívání TEXu a dalších sázecích a DTP systémů desetitisíci vědců, výzkumných pracovníků a dalších intelektuálů vyústilo ve dva poněkud znepokojující fenomény: (1) stále víc lidí se častěji věnuje tomu, aby jejich publikace dobře *vypadaly*, místo aby se starali, zda jsou tyto publikace správné po faktické stránce nebo zda jsou dobře napsány, a (2) stále méně a méně lidí myslí při prvním otevření knihy na její obsah a místo toho začínají knihu posuzovat podle formy nebo — přesněji řečeno — podle toho, jak je vysázena a jaký je její design. Vyrůstáme vlastně v generaci návrhářů a typografů samouků, ale mlčky přitom pomíjíme léta učení a absolvování oboru s výučním listem, což předchozí generace pokládaly za nezbytné.

Samo o sobě to není špatná věc — existuje velmi mnoho samozvaných „expertů“, vždy ochotných zasvětit nováčky do tajemných záhad svého řemesla za nezanedbatelnou finanční sumu — ale máme-li se efektivně učit z různých zdrojů, musí být začátečník vystaven dobrým i špatným příkladům ze svého oboru a kriticky promýšlet, v čem se tyto příklady od sebe liší. Na katedrách typografie a designu bývá hodně těchto příkladů k dispozici a profesoři denně porovnávají a zdůrazňují rozdíly mezi dobrým a špatným k nemalému užitku svých studentů; ale v incestním světě TEXu jsou dobré příklady vzácné, zatímco hojně je těch špatných.

Položme si otázku proč. Co způsobuje, že TEX, jenž ve zručných rukou dokáže produkovat výsledky srovnatelné s nejlepšími příklady horké sazby, zároveň podporuje druhořadou nebo dokonce podřadnou knižní úpravu? Řekl bych, že na tuto otázku existují dvě hlavní odpovědi: (1) v manuálu *TEXbook*, což bývá první (ne-li jediná) kniha, kterou uživatelé TEXu[1] otevřou, je věnováno neuvěřitelně málo místa designu dokumentu oproti jeho formátování, a (2) standardy zabudované v LATEXu poskytují výsledky, které by i ten nejvěrnější stoupenec LATEXu stěží mohl hájit jako „umělecký produkt“ knižního designu a o nichž by mohlo být střízlivými slovy po právu řečeno, že postrádají jakoukoli jemnost a finesu.[2]

Nedostatek přímého vedení uživatele spolu s dost ubohými příklady tvořenými standardními styly LATEXu ústí tudíž v geometrický nárůst knih s mizerným designem, z nichž ze všech čiší „TEX“ (nebo „LATEX“).[3] Nebylo by fér vůči autorům uvádět jednotlivé příklady tohoto vzývání prostřednosti, nicméně krátký pohled do jakékoli rozsáhlejší knihovny s knihami o TEXu (nebo dokonce s knihami v TEXu vysázenými) by ukázal, co mám na mysli.

Není však nic ztraceno: objevuje se nová generace sazečů v TEXu, kteří — zdá se — studují sazečské řemeslo, a několik posledních knih o TEXu podává důkaz, že měly skutečně navržený *design* a že nebyly předčasně vyrrvány z rukou svého autora.

Navrhuji tedy, abychom se v této přednášce zabývali tím, co odlišuje knihu s designem dobrým od knihy s designem špatným (nebo ještě hůř, od knihy zcela bez designu). A doufám, že budu tímto způsobem moci svým malým dílem přispět k univerzálnějšímu přijetí TEXu. Neboť dokud budou profesionální nakladatelské domy vidět pouze špatné příklady v TEXu vysázených knih, nepřijmou pravděpodobně TEX jako sazečský standard; pokud se ale podaří zvednout úroveň v TEXu vysázených knih tak, aby byly buď nerozlišitelné, nebo dokonce lepší než knihy vyrobené tradičními prostředky či komerčními sázecími programy, pak už pouhá ekonomická úvaha zaručuje, že nakladatelství budou věnovat TEXu pozornost, která mu náleží.

---

[1] na rozdíl od LATEXu

[2] Holanďané, tradičně k těmto otázkám citliví, vyrobili subsystém „Sober“ („Střídmý“), který se snaží nejhorší výstřelky předdefinovaných stylů LATEXu potlačit.

[3] Knuth v závěru své práce napsal: „Nyní jděte a vytvořte *mistrovská díla knižního umění*.“ Nikde, alespoň pokud vím, nenapsal: „a nechť z každé stránky těchto děl je okamžitě poznat ‚TEX‘“...

2 Kniha

Všichni víme, jak kniha vypadá, neboť všichni ji denně používáme; v západní kultuře kniha v podstatě představuje množinu listů papíru stejné velikosti, spojených nějakým způsobem při levém okraji a vložených mezi dva o maloučko větší listy z poněkud tvrdšího a silnějšího materiálu, který knihu obepíná i po levém okraji. Od časopisu se odlišuje především poctivostí svého obalu: časopisecký je pouze trošku pevnější (třebaže často i lesklejší) než stránky, které chrání, zatímco u knihy je téměř bez výjimky buď silnější nebo tvrdší nebo obojí; obálka časopisu má také většinou stejný rozměr, zatímco většina knižních obálek je nezávislá na ostatních. A ještě jeden aspekt odlišuje i tu nejtenčí knihu od nejtlustších časopisů: kniha je obvykle vázána po *blocích*, zatímco časopis bývá prošit ve hřbetě jako jeden celek.

Otevřme ale knihu a otevřme časopis a uvidíme, že tyto odlišnosti jsou pouze povrchní; neboť daleko podstatnější odlišnosti se objevují uvnitř. Časopis je charakterizován různorodostí — každá stránka je zřetelně odlišná od předcházející i následující; na rozdíl od toho je pro knihu charakteristický jednotný ráz — z dálky je prakticky jedna stránka nerozlišitelná od druhé (kromě několika zvláštních stran). A v této jednotnosti leží podstata úspěchu knižního designu; neboť čtenáři tuto jednotnost *očekávají* a cokoli, co se od ní odchyluje, čtenáře pouze ruší.

Nicméně jednotnost sama nestačí: té bychom mohli dosáhnout i tím, že ponecháme každou stranu prázdnou nebo prostě její okraje obtáhneme velkým černým rámečkem; to však našeho čtenáře, který hledá nejen jednotnost, ale též *informaci*, neuspokojí. A informace je skutečným *raison d'être* knihy; bez ní kniha neslouží vůbec žádnému účelu a v nejlepším případě je pouhým uměleckým dílem (a v nejhorším je naprosto bezcenná).

Kniha tedy existuje proto, aby nám poskytla informaci; a cokoli, co potlačuje nebo ruší tok informací z knihy ke čtenáři, knihu znehodnocuje. Je-li tok informací narušen příliš, čtenář knihu prostě odloží. (Kolik z nás při pokusu přečíst si stranu vysázenou inverzně písmem Bodoni v jinak konvenčním časopise to prostě vzdá a nechá článek nepřečtený? Já sám jsem to už udělal mnohokrát a nadával jsem typografovi za jeho stupiditu, že dal přednost formě před funkčností.)

Jednotnost, informace: a co ještě? Pokud je kniha v jakémkoli smyslu *technická* (čímž vylučuji román, ale jinak zahrnuji téměř vše ostatní), pak je rovněž *strukturována* (jak uvidíme, je ve skutečnosti v mnohém

smyslu strukturován i román, ale ne v tom, jaký mám na mysli nyní); díky této strukturovanosti je možné ke knize jako ke struktuře přistupovat. Bude mít přinejmenším nějaký obsah; *měla by mít* rejstřík (třebaže značná část knih, kterým by rejstřík velice prospěl, zůstává v tomto ohledu mnohé dlužna) a může mít rovněž svou vnitřní strukturu, v níž může být čtenář čas od času odkázán *ke kapitole třetí* nebo též *viz odstavec 2.4.2.* Tvrdím, že tyto tři prvky tvoří jádro úspěšné knižní úpravy: jednotnost, informace a struktura. Probereme postupně každý z nich, abychom si ukázali, jak jich nejlépe dosáhnout, jak je naplnit či uskutečnit.

3 Jednotnost

Vezměte si knihu (tradiční knihu, ne nějakou horkou novinku z líhně revolučního DTP) a prolistujte ji, jako by v ní měl být onen starodávný animovaný obrázek s postavičkami v rohu každé stránky. Co vidíte? Většina lidí získá dojem pravidelné šedivé mřížky: ne černobílé — tu vidíte jen, když se díváte na jednotlivou stránku — ale šedivé skvrny tam, kde je umístěn text, a bílé tam, kde žádný text (či něco jiného) není. To, co stojí za povšimnutí, je skutečnost, že bílá se objevuje na každé stránce na stejném místě: nad záhlavní a pod patní řádkou, mezi nimi a samotným textem a nalevo a napravo od textu, na okrajích stránky. Je-li kniha vysázena ve více sloupcích (obvykle ve dvou až na specializovaná díla), pak se další blok (nebo bloky) bílého místa objeví v oddělení sloupců.

A v mnohém smyslu jsou tato bílá místa nejdůležitějšími grafickými prvky, které budou tvořit každou stranu. Představují rámec či matrici, do níž je umístěn „tmavý materiál“ — text, kresby atd., které tvoří obsah *informace* na dané straně. Zřejmě však proto, že tento prostor v sobě žádnou informaci nenese, je mu často věnována menší pozornost, než vyžaduje, zvláště těmi, ktdož tvoří knižní úpravu bez jakéhokoli formálního vzdělání. A i když to *vypadá*, že v sobě žádnou informaci nenese, ve skutečnosti je v tomto prostoru informací mnoho: bez něj bychom nevěděli, kde končí záhlavní řádek a začíná samotný text; kde končí text a začíná pata; kde končí levý sloupec a začíná pravý a tak dále… Opravdu je tento bílý prostor pro naše vnímání obsahu stránky *naprosto nezbytný*, a je tudíž *přinejmenším* stejně důležitý jako všechny ostatní prvky na stránce, ne-li důležitější.

Neboť bílý prostor a tmavý text jsou od sebe navzájem neoddělitelné části — jedna začíná tam, kde druhá končí, až po fyzické hranice stránky — jakákoli diskuse o jednotnosti bílého místa musí být stejně

spojena s diskusí o jednotnosti tisku na stránce. Je tu však navíc vedle této jednotnosti ještě třetí prvek, který mnohem víc závisí na neoddělitelném propojení bílých míst a tištěného textu, a tím je pocit „šedosti“ každé jednotlivé strany. Lidské oko je pozoruhodně citlivé na drobné odchylky v odstínech šedi, a pokud se zdánlivá šedost mění ať už na jedné stránce, nebo mezi jednotlivými stranami (zvláště pak mezi stranami ležícími proti sobě a tvořícími *dvojstránku*), může tento efekt docela vést ke zmatku. Tyto odchylky ve zdánlivé šedi mohou mít nejrůznější příčiny, z nichž nejobvyklejší jsou: (a) rozpalování písma k vyplňování řádků na formát; (b) nerovnoměrný rozpal řádků mezi dvěma nebo více bloky textu vysázenými stejným typem písma; (c) nepřiměřené změny v rozpalu řádků nebo v písmu (nebo v obém) při záměrném vysázení bloku odlišným typem písma (například v několikařádkových citacích). Léky na každý z těchto neduhů jsou jednoduché: *nikdy* neužívejte rozpalu písma k doplňování řádků na formát; *nikdy* nedovolte, aby sazečský systém měnil řádkový rejstřík ve snaze po vyplnění vertikálního rozměru formátu (a nikdy nesázejte dva bloky textu stejným písmem, ale s různým řádkovým rejstříkem, aniž byste si byli vědomi efektu, kterého tím docílíte); a vždy, když sázíte bloky textu různými typy písma, mějte na paměti dosaženou úroveň šedi (poměr bílého a tmavého místa na stránce).

V ideálním případě by pozornost věnovaná předchozímu odstavci měla z velké části zajistit, aby viditelná šedivost jednotlivých stran byla jednotná: jenže je zde další problém, který pramení z našeho ne zcela dokonalého světa a který může rovněž výrazně ovlivnit dosažený stupeň šedi. Jedná se o problém „protisku“. Ideální papír představuje jednotnou neprůsvitnou bělobu, na níž je nanesena tiskařská čerň; na druhé straně skutečný papír, třebaže je rovnoměrně bílý (přinejmenším pokud jde o tiskárenský papír vysoké kvality), není zcela neprůsvitný; když jej dáte proti světlu, i ten nejlepší papír nějaké paprsky propustí. A horší papíry jsou tak transparentní, že vytištěný text je možné přečíst téměř bez námahy z rubu stejně jako z líce stránky (i když zrcadlově obráceně). To by samo o sobě nemuselo znamenat žádný problém, kdyby obě strany listu nebyly dva na sobě nezávislé celky: nejenže se tisknou každá zvlášť, ale text vytištěný na jedné straně má v sobě malou nebo vůbec žádnou korelaci k textu na straně druhé. Avšak při *návrhu* těchto stránek musíme mít efekt jejich potištění z obou stran na paměti a dobrá úprava se bude snažit zajistit, aby každý řádek textu na líci přesně zapadal do řádku na rubu. V praxi toho samozřejmě nelze dosáhnout; proud textu je přerušován odstavci stejně jako ilustracemi a dalšími grafickými prvky:

ale *záměrem* úpravce musí být, aby této vyváženosti řádků proti sobě dosáhl. Na této filosofii je založen celý koncept *řádkového rejstříku*. Rejstřík představuje abstraktní model každé strany; zvláštní stránky (například počátky kapitol) mohou mít svůj vlastní speciální rejstřík, ale normální, „obyčejné“ stránky budou mít každá rejstřík naprosto stejný a ten se vyplní různými prvky na té které straně. Rejstřík můžeme chápat hierarchickým způsobem: v nejpovrchnější rovině se jedná o fyzické hranice stránky, levé a pravé okraje textu (nebo sloupců při díle vysázeném ve více sloupcích), horní a dolní okraje stránky, záhlavní a patní řádek. V další úrovni jde o jemné propracování; stránku rozdělíme na řádky textu (z tohoto důvodu většina tradičních pouček o knihách vyjadřuje rozměry strany spíše v počtech řádků textu než v bodech, palcích nebo centimetrech). Přiloženy k sobě se musí tyto dva rejstříky dokonale krýt, každý řádek textu v jednom musí odpovídat řádku v druhém; levý okraj textu na líci musí přesně souhlasit s pravým okrajem na rubu a tak dále (což má další důsledky pro okraje stran, jak uvidíme později).

Rejstřík je pochopitelně jen ideálem, který je třeba čas od času porušit; kdyby tomu tak nebylo, existoval by kupříkladu jen konečný počet pozic, kam by bylo možné (například) umístit titulek nad text, který uvádí: jeden řádek, dva řádky, tři řádky, atd. Takové měřítko je však přespříliš hrubé pro požadavky na estetiku opravdového knižního designu a s nadpisy je tudíž třeba zacházet jako se zvláštními úkazy, kterým je dovoleno vybočit z jejich „přirozeného“ řádku v knize, zatímco odstavce nad a pod titulkem musejí v rejstříku zůstat. S ilustracemi a grafikou je rovněž nutno zacházet jako se speciálními případy, posazenými do bílého prostoru vyznačeného příslušným počtem řádků textu s tím, že ilustrace samotné budou na rejstříku nezávislé, zatímco okolní odstavce zůstanou pevně na místě.

Někdy však tyto požadavky na zalomení každé stránky zapříčiní, že se strana zcela nezaplní: odstavec například může skončit takovým způsobem, že zde není dost místa (například pouze jeden prázdný řádek), aby nový odstavec začal; nebo můžeme mít dost místa pro nadpis, ale ne už pro nadpis a prázdné místo pod ním s alespoň dvěma řádky textu. Jak v těchto případech řešit rozpor mezi požadavky na jednotnost a grafickou úpravu stránky? Je třeba říci, že v krajním případě neexistuje žádné obecné pravidlo, které by se vždy dalo použít, a skutečné knihy budou muset jedno či druhé omezení porušit; stejně často však lze nalézt řešení, které je jak elegantní, tak esteticky uspokojivé: porušit stejným způsobem jednotnost dvou proti sobě stojících stran (tzn. *celé dvojstrany*).

Například jestliže rub (tedy levá stránka) je o jeden řádek kratší, pak *uměle* zkraťte líc (pravou stránku) rovněž o jeden řádek; jestliže rubová stránka vyjde o jeden řádek delší, dovolte jí to, ale požadujte, aby i lícová strana byla o jeden řádek delší.

V konceptu vyváženosti *dvojstrany* — na rozdíl od jednotnosti *všech* stran — spočívá, jak věřím, podstata dobré úpravy. Neboť držíme-li v ruce otevřenou knihu nebo položíme-li ji otevřenou na stolek či lavici, neuvidíme nikdy jednu stránku, ale vždycky dvojstranu; a pokud obě stránky v tomto rozevření vypadají jednotně (jednotně co do šedivosti nebo vizuální hustoty; jednotně v umístění záhlaví i paty stránky; jednotně co do velikosti okrajů — vnější okraje oba stejně velké, vnitřní okraje rovněž stejně velké, nikoli však nezbytně stejně velké jako viditelná[4] šířka okrajů vnějších; a jednotně musí obě strany vypadat i co do posazení rejstříku, rubové řádky musí tedy dokonale odpovídat svým protějškům na lícové straně) a vyváženě (s rubovou i lícovou stránkou začínajícími ve stejné výšce od paty a pokračujícími do stejné vzdálenosti směrem dolů), pak už jsme jistého stupně dobré úpravy dosáhli.

Existuje však spousta praktických problémů spojených s návrhem rejstříku a vyvážených dvojstran; některé z nich jsou specialitou při používání TEXu, zatímco jiné mají obecnou povahu. Ty, které se týkají výlučně TEXu, probereme v druhé části této přednášky. Zde se zaměříme na obecné problémy.

Uvžujme nejprve problém s vyvážením dvojstránek: výše jsme uvedli, že pokud je rubová stránka o jeden řádek kratší nebo delší, má být protilehlá lícová strana *uměle* o jeden řádek zkrácena či prodloužena. Ale co když rubová stránka bude mít normální rozměr, a o jeden řádek míň nebo víc bude mít strana lícová? Pokud je grafická úprava dělána na základě definice jednotlivých stran, pak je už v téhle chvíli pozdě znova sázet rubovou stránku s tím, že buď bude mít lícová strana nepřirozený počet řádků (pokud má grafická úprava stránky dostatečnou volnost, aby to dovolila), nebo bude porušena vyváženost dvojstrany. A zde tedy musíme stanovit pravidlo, že kterýkoli sazečský systém, navržený k produkci knih s dobrým designem *musí* být schopen přinejmenším sázet *dvojstránky* jako jeden celek, a to spíš než jednotlivé stránky samy. To samozřejmě náš problém zase úplně neřeší: například pokud rubová stránka o jeden řádek „přeteče“, ale rubová strana končí odstavcem ve své přirozené výši,

[4] Termín *viditelná šířka* zde užívám zcela záměrně, neboť jak ještě uvidíme, viditelná šířka vnitřních okrajů je vždy menší než jejich skutečný rozměr, a to o míru, která je funkcí tloušťky knihy a použitého způsobu vazby.

pak asi nebude možné připojit k lícové straně další řádek, aniž bychom porušili jisté (mlčky předpokládané) omezení; za těchto okolností může být nezbytné vrátit se dokonce i na předchozí strany a začít si klást otázky typu: „Co kdybych vysadil předchozí dvojstránku o řádek kratší nebo o řádek delší?“ apod.; čím více různých rozhodnutí lze nakonec učinit, tím pravděpodobnější je, že se nám podaří dosáhnout lepšího výsledku. Při neustále klesajících cenách počítačové paměti není v žádném případě nemyslitelné uvažovat o optimalizaci celé kapitoly zároveň.

A dále k jednotnosti: co když se několikařádkový citát, vysázený drobnějším typem písma s obdobně upraveným záhlavím, *musí* objevit jako jeden celek na jedné straně, zatímco na (fyzicky) opačné straně žádný takový citát není? Pak se bezpochyby po celé délce citátu objeví protisk a v nejhorším případě dojde k interferenčímu efektu tam, kde se řádky citace budou dotýkat a překrývat s řádky normálního textu na druhé straně. Zde je třeba odkázat na grafickou úpravu obecně, bez ohledu na to, kolik materiálu se nám podaří sesbírat; nakonec jsme zde více závislí na zručnosti výrobce papíru (jaké neprůsvitnosti dokáže dosáhnout) než na naší dovednosti v užívání a programování příslušného sazečského systému.

Jednotnost však znamená mnohem víc než jen pravidelný řádkový rejstřík a vyvážené dvojstrany. Jednotnost je pojem, který je součástí každé dobrého knižní úpravy. Vezměme například zacházení s nadpisy kapitol a jednotlivých oddílů, s odstavci, citáty a tak dále: v jakém smyslu lze i tyto věci „sjednotit“? Je zcela zřejmé, že každá z nich musí mít svou osobitost, aby čtenář mohl okamžitě rozpoznat, na co se vlastně dívá; přesto pokud nejsou pouze osobité, ale navíc poskytují širokou škálu rozdílných typografických přístupů, pak je jakýkoli pocit logické souvislosti ztracen a kniha začíná připomínat zmatenou směsici návrhářských nápadů.

Můžeme začít tvrzením, že by v knize mělo být použito jen malé množství odlišných řezů písma — „čím míň, tím líp“ měl by znít axiom pro výběr písma! — a stejně tak by mělo být použito malé množství rozdílných umístění textu. Například mají-li odstavce zcela vyplněné řádky (což je pravděpodobné v případě knihy — výjimky z tohoto pravidla budeme probírat jinde) a jestliže nadpisy jednotlivých oddílů jsou odraženy od levého okraje, pak by se asi kniha jako celek měla omezit na dva možné styly sazby: bylo by nepřiměřené mít v takovéto knize s vyplněnými odstavci a doleva zarovnanými titulky nějaké centrované nadpisy. Jsou-li ale nadpisy oddílů odraženy od levého okraje (případně ve spo-

jení se zlomy řádku po smyslu textu[5]), zatímco normální odstavce jsou plně zarovnány, pak citáty je možno vysázet buď plně zarovnané (jako odstavce) nebo je odrazit od levého okraje (jako nadpisy oddílů), asi by však neměly být bez dobrého důvodu odraženy vpravo.

A co zarážky? Zde existují dva odlišné přístupy. Jeden praví, že požadavek jednotnosti zahrnuje i zarážku a že jednou zvolená zarážka by měla být aplikována na celou knihu: tudíž s okraji by se např. mělo zacházet stejným způsobem jako u odstavců; citáty by se měly sázet s dalším odsazením, stejným jako tato zarážka; a bibliografie by se mohla sázet s negativní zarážkou, rovněž stejné velikosti. Druhý přístup říká, že požadavky na jasnost a nedvojznačnost diktují nutnost použít *odlišnou* zarážku všude tam, kde jde o různé celky, a tak dát čtenáři maximum indikací o povaze celého textu již při pouhém letmém pohledu na stránku. Chovám sympatie k oběma pohledům na věc, ale můj v zásadě konzervativní přístup považuje první z nich za přitažlivější. Nemyslím, že bych kdy viděl příklad toho, jak může být čtenář spletený použitím jednotné zarážky v knize. Celá tato oblast přesahuje rámec *jednotnosti* (z níž plyne požadavek jednotné zarážky) a přerůstá do pojmu *informace* (z něhož vyplývá možnost použití různých zarážek k různým účelům), který nás přivádí k další kapitole.

4 Informace

Primární funkcí jakékoli knihy je sdělení informace; v předchozí části jsme se nicméně téměř výhradně zabývali estetikou knižního designu spíš než rolí knihy jako informačního a komunikačního média. Avšak za předpokladu, že mezi těmito dvěma idejemi nedojde ke konfliktu, jednotný a esteticky příjemný vzhled knize velice pomáhá v její komunikativní roli, neboť dovoluje čtenáři soustředit se na text (tzn. na *informační obsah* knihy), aniž by ho rušila její úprava (skutečnost, která je bohužel opomíjena mnoha dnešními *avantgardními* návrháři). Zde ale docházíme k bodu, kde další přílišné trvání na pravidlech jednotnosti by nás odvedlo od primární role knihy jako zdroje informací. Proto nyní obrátíme svou pozornost na tuto oblast.

Vyjděme napřed z funkce nadpisů jednotlivých oddílů: těch jednořádkových (občas víceřádkových) textů, které slouží čtenáři jako úvod

[5] Návrh, v němž je sazba na praporec používána společně s „vřele doporučovanými“ oddělovači řádek, zajišťujícími, že jednolité myšlenky (fráze, tvrzení, výroky atd.) nemusí být nutně rozděleny do dvou řádků.

k myšlenkám, jež následují. Tato přednáška například používá pouze jednoduché titulky pro nadpisy oddílů, neboť autor dal přednost v každém oddíle skočit rovnou do textu; jiní autoři, obzvláště ti s pevným vědeckým zakotvením, se cítí šťastnější, když mohou své myšlenky utřídit přísně hierarchickým způsobem, a často se uchylují nejen k *první* úrovni titulků (jako v této přednášce), ale též k druhé, třetí, čtvrté a výjimečně též k páté úrovni. První požadavek na tyto nadpisy zní, že musí — *jednoznačně* — odkazovat k textu, který následuje: nemělo by se v knize s dobrým designem stávat, aby se nadpisy oddílů vázaly k předchozímu textu. Prostředky, kterými se toho dosáhne, jsou jednoduchost sama, nicméně je v amatérsky navržených knihách i jiných dokumentech porušována tak často, že člověk váhá, jestli tato myšlenka jejich tvůrce vůbec kdy napadla: nadpis oddílu by měl být fyzicky blíž textu, který uvádí, než tomu, jenž předchází. Všimněte si, že se jedná zásadně o vztah „menší než“, nikoli „menší nebo rovno“: nadpis *nikdy* nesmí být uprostřed mezi předcházejícím a následujícím textem. Toto pravidlo má některé zajímavé vedlejší dopady: například ten, že nadpis se nesmí *nikdy* objevit izolovaně na konci strany, neboť jinak by podle definice byl předcházejícímu textu blíže než textu následujícímu.

Ale v přísně hierarchicky stavěné knize nebo přednášce je stejně tak důležité, aby byly i různé úrovně nadpisů (první, druhý, atd.) rozlišitelné na první pohled. Jak by tato hierarchie mohla být co nejlépe předána čtenáři? Máme po ruce několik možných přístupů: (1) Nadpisy vyšší úrovně mohou být odděleny od předchozího textu větším bílým místem než nadpisy úrovně nižší; (2) nadpisy vyšší úrovně mohou být odděleny od svého (následujícího) textu větším bílým místem než nadpisy úrovně nižší; (3) nadpisy vyšší úrovně mohou být vysázeny větším písmem; (4) nadpisy vyšší úrovně mohou být vysázeny tučnějším písmem; (5) pro jednu nebo i více úrovní nadpisů je možné použít další typografickou diferenciaci (např. *groteskové písmo* v knize nebo dokumentu vysázeném jinak *patkovým písmem*); (6) Pro nejnižší úroveň nadpisů je možno použít titulky včleněné do textu. Ve skutečnosti jsou toto jen některé z dostupných možností: například v některých dílech jsou oddíly nejvyšší úrovně sázeny vždy na novou stránku, i když je tento oddíl pouze jedním z mnoha v rámci kapitoly.

Škála možností je zřejmě obrovská a v této stručné přednášce nelze udělat víc, než předložit několik obvyklých konvencí, avšak jeden požadavek je zcela zásadní: jestliže v jediném dokumentu použijeme dvě nebo více pravidel, pak žádná z jejich kombinací nesmí vést k nejasnosti textu.

Například, jestliže nadpisy nejvyšší úrovně jsou vysázeny v šestnáctibodové antikvě, nadpisy druhé úrovně nesmíme vysázet čtrnáctibodovým tučným písmem, neboť tučnost nadpisů druhé úrovně by rušila efekt menšího písma a vedla by k nejasnosti v mysli čtenáře. Dokonce i když není užito přímo tučné písmo, může se stát (například výběrem špatně korespondujícího *groteskového písma* pro nadpisy druhé úrovně v dokumentu vysázeném jinak *patkovým písmem*), že omylem zvolíme *na pohled* tučnější písmo pro nadpisy nižší úrovně. Takovéto nejasnosti se musíme vyvarovat.

Jakými dalšími způsoby může návrhář zajistit, aby se informace co nejlépe přenesla ke čtenáři? Snad ze všeho nejdůležitější je přesvědčit se, že kniha se dá *číst*! Řekli byste možná, že to se rozumí samo sebou, ale existuje bohužel příliš mnoho již publikovaných protipříkladů, které zrovna tento požadavek nesplňují z žádného rozumného kritického hlediska. Možná je třeba, abychom napřed definovali, co míníme výrazem, že se kniha „dá číst“; má-li být četba efektivní a příjemná, pak tvrdím, že pro průměrně schopného dospělého člověka musí jít o téměř nevědomou činnost. Když si vezmu knihu s cílem získat z ní nějakou informaci, pak skutečnost, že je mi do chřtánu násilím cpána návrhářova osobnost (kromě případu, kdy jde o knihu o knižním designu, v kterézto situaci bych se mohl už podle designu rozhodnout, jestli stojí za to ji číst), je tou *poslední* věcí, kterou bych si přál; knižní úprava tedy musí být velmi „nenápadná“ a nesmí rušit; musí dovolit, aby obsah přirozeným způsobem vyplňoval formu, spíše než aby forma trčela ze stránky a odváděla čtenáře od obsahu. Pochopitelně z tohoto pravidla existují výjimky a do kategorie těchto výjimek spadají například zcela zřejmě knihy o designu, neboť jsou ve své podstatě samovztažné, ale obecně vzato chce čtenář vědět o návrháři jen málo, a o obsahu tolik, kolik jen lze.

Navíc musí být možno nerušeným způsobem v četbě pokračovat; je dobře známo, že jakákoli chyba ze strany autora ústí do nejasnosti v čtenářově mysli a zapříčiní tak, že ten se musí v textu vracet v naději, že vysleduje další klíče k dílu, a tudíž luští text až na druhé nebo další čtení. Klasičtí autoři knih o gramatice (Fowler, Weseen, Partridge, Onions, Gowers, Quiller-Couch, Sweet) věnují této skutečnosti velmi mnoho pozornosti. Existuje ale i značné množství typografických pastí, které rovněž mohou způsobit, že čtenář listuje zpátky, a je pro návrháře stejně důležité vyhnout se těmto nástrahám jako pro autora vyhnout se léčkám gramatiky.

Například ve třicátých letech bylo velkou módou používat v typografii *bezpatkového písma*: bylo to moderní, *avantgardní*, stylové, módní — vyberte si termín, jaký chcete. A obzvlášť v Severní Americe a trochu méně i v Evropě byl takový tlak na používání této typografie, že se na její *raison d'être* — jednoduchý, minimalistický styl kratších úseků textu, které neodvedou pozornost od hlavního tématu (často doplňující grafiku) — zcela zapomnělo a tato typografie byla chápána (a užívána) jako mustr v každém myslitelném případě. Ty se neomezily pouze na klasická užití v záhlavích, titulcích, na plakátech atd., nýbrž se místo toho rozšířily tak, že zatemňovaly i obyčejný text v knihách; každá stránka byla vysázena *groteskem*, s pramalým citem pro pohodlí a pohodu čtenáře. Efekt, který to mělo na čtenáře, se dal (při zpětném pohledu) naprosto předvídat: čtenáři zjišťovali, že se na takové knihy v libovolně dlouhém časovém úseku špatně soustředí, že je unavují a dokonce vyčerpávají; a důvod byl velice prostý, třebaže v tom čase nebyl dobře pochopen: třebaže *patky*, jež dnes charakterizují většinu naší klasické typografie, nejsou ve skutečnosti ničím víc než artefakty, vztahujícími se k dávným písmovým formám tesaným do kamene (obzvlášť v případě velkých písmen) a k pozdějším řezbářům písem, přesto tyto *patky* mají velice důležitou funkci, když se tato písmena objeví uprostřed textu: *patky* zde slouží k tomu, aby oko přirozeně plulo v lince textu, a značně snižují riziko, že oko bude kolísat mezi dvěma sousedními řádky textu. Rovněž napomáhají k minimalizování zpětného vyhledávání v rámci jednoho řádku. A tak se díky zpětnému pohledu a na základě psychologického a fyziologického výzkumu dospělo k přesvědčení, že typografie určená pro pasáže hladkého textu (na rozdíl od titulků apod., což se týká nejvýše pár řádků pohromadě) je takřka výlučně typografií *patkového* písma. Je smutné, že je dodnes tato skutečnost občas ignorována.

Je-li však výběr *patkového* písma téměř povinný, aby se zabránilo těkání mezi jednotlivými řádky textu a zpětného vyhledávání na řádku, jaké další psychologické a fyziologické faktory mohou též ovlivnit čitelnost textu? Snad nejdůležitější ze všeho — a důvod, proč normální TEX poskytuje velmi špatné vedení — je velikost písma vzhledem k *šířce sloupce* (tedy délce řádky) textu. PlainTEX je určen pro použití desetibodových řezů na šířku 6,5 palce (39 pica), což jednoduše řečeno znamená přespříliš znaků na řádek. Psychologové prokázali, že optimální počet znaků na řádku pro lidi s normálním zrakem se pohybuje někde v rozmezí mezi čtyřiceti až sedmdesáti, přičemž toto optimum se blíží spíš horní hranici tohoto intervalu; je-li znaků méně, jsou lidé frustrováni:

jedním pohledem obsáhnou velice malý výsek informací; a je-li znaků více, mají čtenáři sklon ztrácet se v textu a buď se v řádku vracejí, nebo skáčou opětovně znova a znova na začátek řádku následujícího, ztrácejí se vertikálně a přeskakují na začátek špatného řádku. Dokonce i LaTeX, který obecně v záležitostech typografického designu uživatele vede lépe než obyčejný TeX, umožňuje naprostou svobodu při výběru mezi desetibodovým, jedenáctibodovým nebo dvanáctibodovým písmem, nehledě na zvolený styl, a tudíž i šířku textu. Evropským čtenářům, zvyklým na velikosti papíru v normě DIN, mohu poskytnout takovéto nejlepší rady: sázíte-li na formát A4 (což je nepravděpodobné v případě knihy, ale docela možné pro zprávu nebo jiný podobný dokument) s „normálními“ okraji (přibližně 1 palec), pak si to žádá dvanáctibodové písmo; dá se to zvládnout i s jedenáctibodovým písmem, ale desetibodové je naprosto nepřípustné. Totéž platí pro severoamerické čtenáře při palcových okrajích na listu amerického „dopisního“ papíru ($8,5'' \times 11''$). A pokud jde o knihu? K otázce, „jak je kniha velká“, se ještě vrátím, ale obecně řečeno sluší knihám desetibodová typografie; nicméně při natahování šíře papíru se stávají nezbytnými dva sloupce, případně je třeba udělat patologicky široké okraje.[6] V nezvykle malých knihách lze použít i devítibodového písma, ale při čemkoli menším už vznikají pro lidi s normálním zrakem problémy s čitelností.

V předchozím odstavci jsem hovořil o „desetibodovém písmu“, jako by se jednalo o nějaký standard ISO; bohužel tomu tak není. Fonty se liší jak svou skutečnou velikostí (měřitelnou), tak velikostí vnímanou, a citovaná velikost je v lepším případě jakousi aproximací a v nejhorším sprostou lží! Neboť důležité je, že teoretická velikost písma je vzdáleností, která odděluje dva po sobě jdoucí řádky v textu odstavce tímto písmem vysázeném, aniž by se spodní dotažnice v jednom řádku stýkaly s horními dotažnicemi v řádku následujícím; je to rovněž přibližně výška + hloubka oblých závorek. Ve skutečnosti však desetibodové písmo jednoho typografa může být jedenáctibodovým typografa jiného; a používáte-li dva nebo více druhů písma v jednom dokumentu, pak je na vaší odpovědnosti návrháře zajistit, aby používané velikosti vizuálně korespondovaly, i když to třeba znamená vybrat u jednoho typu desetibodové a u jiného jedenáctibodové písmo (nebo třeba i 10,634 7 bodu, pokud toto číslo vyjadřuje ten správný poměr mezi jejich vnímanými velikostmi).

[6] Jeden severoamerický student mi sdělil, že v Severní Americe mají studenti zvyk dělat si do knih poznámky; z tohoto důvodu *očekávají* v knize daleko širší okraje než evropští čtenáři, což může vysvětlovat něco z předdefinovaných stylů LaTeXu.

A nyní k rozpalu řádků: některé autority tvrdí, že by mělo jít o „1,2násobek navržené velikosti fontů“; jiní doporučují „2 body nad tuto navrženou velikost“; a jiní by přišli s dalšími pravidly. Odpověď samozřejmě zní, že žádná definice nemůže jednoznačně platit pro každý typ písma nebo pro každou jeho velikost, a dokud vás zkušenost nenaučí podívat se na vzorek písma a *vědět*, jaký správný rozpal máte zvolit, budete muset používat nejmocnější prostředek, jaký máte k dispozici: své oči. Jinými slovy, budete si muset vytisknout ukázku textu s různými rozpaly mezi řádky (kupříkladu v pořadí velikostí uvedených ve vzorcích výše) a upravovat to tak dlouho, dokud se vám to nebude zdát dobré. Když ale budete tyto vzorky tisknout, přijdete k další, velice subtilní psychologické zvláštnosti: předpokládejme, že pracujete jako většina lidí a tisknete své obtahy na laserové tiskárně; váš výstup se pak objeví buď na listu formátu A4 nebo na listu velikosti „dopisního“ papíru, a velice zřídka na něčem jiném. A ač se budete snažit sebevíc, nebudete se schopni rozhodnout mezi velikostí písma a rozpalů mezi řádky, jak budou vypadat v konečné podobě knihy, i když si kolem ukázky textu namalujete rámeček v rozměrech konečné podoby oříznuté stránky; vaše oko a mozek odmítnou věřit tomu, že papír kolem vašeho rámečku nepatří k textu, a rozhodnout se mezi velikostí textu a rozpalem řádků na neoříznutém listu A4 nebo „dopisním“ papíru. Řešením je samozřejmě oříznout list do výsledné podoby stránky a pak k sobě dvě takto oříznuté stránky přiložit (nebo vytisknout rovnou vysázenou dvojstránku) a prohlédnout si výslednou dvojstranu knihy v „životní velikosti“; a potom — ale opravdu až potom — budete schopni správně se rozhodnout mezi velikostí písma a rozpalů mezi řádky na vytištěné straně.

5 Struktura

Obraťme nyní svou pozornost ke *struktuře*, a obzvlášť k prostředkům, pomocí nichž je možné vytvořit odkazy (včetně křížových odkazů) spíše kvazináhodným než sekvenčním způsobem. V nejhrubším stupni je kniha rozdělena na díly (pokud je rozsáhlá), části (je-li velká) a na kapitoly (téměř všechny knihy). Přístup k jednotlivým dílům nám nemusí činit nemístné starosti: každý z nich má napsáno jméno a číslo na svém hřbetu a na přední obálce, a pouze pokud jsou dva nebo tři díly otevřeny konkurenčně před čtenářem vedle sebe, je třeba mezi nimi rozlišovat na základě rozevřených dvojstran.

Části nejsou neobvyklé, ale spoustu potenciálních problémů spojených s identifikací jednotlivých částí lze eliminovat sekvenčním číslováním ka-

pitol nezávisle na částech, do kterých jednotlivé kapitoly připadnou; při sekvenčním číslování kapitol lze čtenáře vždy odkázat na *Kapitolu n*, aniž by ji bylo třeba označit jako *Kapitolu n v Části m*.

Nejdůležitější dělení většiny knih je však dělení na kapitoly a zde musíme začít svůj se svědomitým výzkumem *struktury*. Uvažujme klasický příklad knihy s několika kapitolami v jednom svazku, s obsahem umístěným v *přední části* (na tzv. „vstupních stranách“). Čtenář přistupuje ke knize od obsahu, vyhledá si kapitolu, její číslo a jméno (jsou-li kapitoly pojmenovány) a stránku, na níž kapitola začíná. Po výběru ze seznamu kapitol čtenář listuje knihou, aby našel první stranu zvolené kapitoly. To není náhodné hledání: čísla stran monotónně stoupají po jedné, a pokud čtenář přeběhne svůj cíl, je dostatečně obeznámen s obecným principem knihy, aby si uvědomil, že musí listovat zpět.[7] Zajímavý fenomén se však objeví, když se čtenář blíží ke stránce, která ho zajímá, přinejmenším v mnoha ne zcela optimálně navržených knihách: čísla stránek (*paginace*, jak se často říká) se tradičně střídají v levém a pravém horním rohu, kde zabírají místo v levém horním rohu rubové stránky a pravý horní roh lícové stránky; toto umístění by mělo zaručit jejich maximální viditelnost. Jenže na první stránce každé kapitoly opět tradičně bývá zvykem obvyklé záhlaví („záhlavní řádek“) potlačit, neboť design těchto stránek (podrobně probíraný v druhé části přednášky) je takový, že záhlavní řádek je zde obecně považován za esteticky nepřípustný. A tudíž právě ta strana, která (logicky) nese hledané číslo, je přesně tou stranou, na níž se (fyzicky) žádné číslo neobjevuje; a čtenář je donucen provést bližší kontrolu, aby se ujistil, že skutečně našel místo svého zájmu. Udělá to tak, že porovná poslední fyzické číslo stránky, které najde (a které v nejhorším případě nelze z nalistované stránky uvidět, pokud předchozí kapitola náhodou skončila na lícové straně, jelikož bývá rovněž zvykem začínat jednotlivé kapitoly na lícové straně a druhou půli dvojstránky pak bude tvořit vakát na rubové stránce), a číslo stránky fyzicky následující, které je rovněž nevyhnutelně z nalezené stránky neviditelné. Pochopitelně název a číslo hledané kapitoly bude na nalezené straně vi-

[7] Je zajímavé uvědomit si, že popsaný scénář je přesným opakem toho, co se obvykle děje ve skutečnosti: jelikož knihy obvykle buď leží na lavici nebo na stolku, nebo je držíme v pravé ruce s nejvyšším číslem stránky dole, je pro čtenáře daleko přirozenější listovat stránkami *pozpátku* k hledané straně, a pokud přeběhne, teprve potom listovat *vpřed*. Je tomu tak proto, že je daleko jednodušší vzít do prstů jedné ruky víc listů, často téměř všechny listy knihy, a nechat je jednotlivě padat za pomoci gravitace, než každou stránku při hledání předmětu našeho zájmu individuálně zvedat.

dět, a z její úpravy bude patrné, že se *jedná* o první stranu kapitoly, ale čtenář, dosud hledající konkrétní číslo stránky, bude přinejmenším přinucen změnit svůj vyhledávací algoritmus.

Obvykle navrhované řešení tohoto problému spočívá v tzv. *spuštěném číslování* na úvodních stranách kapitol: číslo stránky zabírá místo uprostřed paty. Vnímavý čtenář si na tuto konvenci velice rychle zvykne a modifikuje po dosažení hledané stránky směr svého pohledu do její spodní části spíše než ke krajním rohům dvojstrany. Je-li ale spuštěné číslování přijatelné na prvních stranách kapitol, proč ho nepoužít důsledně v celé knize? Mělo by to dvě výhody: (1) čtenář by byl schopen najít *jakoukoli* stránku knihy pohledem na stejné místo na každé straně bez ohledu na její typ, a (2) uvolnil by se další prostor v záhlavních řádcích pro další křížové reference, místo, jehož důležitost — jak uvidíme — stoupá spolu se složitostí (ve smyslu vnější hierarchické struktury) celé knihy.

Když jsme zaručili očíslování *všech* stránek (kromě vakátů, neboť dle významu toho slova nemůže existovat dobře míněný odkaz, který by vyžadoval, aby čtenář otočil právě na tuto stranu), zaručili jsme i to, že naše obsahy, rejstříky apod., které v případě vyhledání odkazu všechny obecně poskytují *číslo stránky*, povedou spolehlivě k cíli. Musíme teď svou pozornost obrátit k dalším technikám křížových referencí, zvláště pak metodám nalezení logických podkapitol knihy (tedy oddílů, pododdílů atd.) pomocí jejich *jména* a rovněž pomocí jejich *čísla*, pokud jsou vůbec číslovány.

Obecně řečeno, jména a čísla logických podkapitol jsou používána ke křížovým odkazům (tzn. k odkazům z textu) spíš než k odkazům přímým (např. z obsahu nebo z rejstříku); ale bez ohledu na původ odkazu, čtenář bude bezpochyby požadovat odkaz buď typu *viz oddíl 2.1.4.* nebo *viz rovněž Lagopus hyperboreus* — přičemž ani v jednom případě nebude čtenář explicitně instruován, aby si nalistoval nějakou konkrétní stránku. Často je možné tyto *nepřímé* odkazy převést na *přímé* odkazy k číslu stránky prostřednictvím obsahu nebo rejstříku, ale tento dvojstupňový proces je jak frustrující, tak časově náročný: vyžadujeme přímější metodu.

Mechanismus, kterým je tento přímý přístup ke jménům nebo číslům logicky řazených pododdílů dosahován, funguje většinou na základě *živých záhlaví*; o těch jsme se již v této přednášce zmiňovali dříve, aniž bychom podali nějakou formální definici jejich významu a účelu. Průběžné záhlaví se tak nazývá proto, že se objevuje na (téměř) každé

stránce; úvodní stránky kapitol a vakáty jsou většinou z množiny stránek, na nichž se průběžná záhlaví objevují, vyloučeny, a je-li celá strana věnována ilustraci, pak je vhodné vyloučit i tuto; ale kromě těchto speciálních případů se průběžná záhlaví objevují na každé stránce. *Obsah* záhlaví se však pochopitelně mění od stránky ke stránce: kdyby tomu tak nebylo, neměla by záhlaví vůbec smysl (což je rovněž častý případ, kdy je záhlaví použito k opakování titulu knihy na každé nebo každé druhé stránce; čtenář obyčejně zná titul čtené knihy, i když lze nalézt protipříklady, jako třeba v situaci, kdy hledáme v mnoha knihách současně; opakování titulu knihy tak nemusí být vždy důkazem špatného designu). Obecně řečeno, obsah záhlaví by měl odrážet obsah stránky, na níž se vyskytuje; tudíž, začíná-li oddíl *2.1.4: metajazykové poznámky* na straně 23, záhlaví na straně 23 by mělo tuto skutečnost téměř jistě odrážet. Avšak v hierarchicky strukturovaném díle jsou v tomto přístupu přítomny potenciální konflikty; vezměme si knihu rozdělenou na kapitoly, oddíly a pododdíly: který z těchto celků se má objevovat v záhlaví? Často užívanou konvencí jsou rozdílná záhlaví sudých a lichých stránek: sudá stránka obsahuje „důležitější“ informace (např. název a číslo kapitoly), zatímco lichá strana obsahuje informace „méně významné“ (např. název a číslo oddílu). To však nestačí: kde se objeví informace o pododdílu? Zde neexistuje žádné jednoznačné řešení: je-li kniha hodně složitá (tzn. je strukturována příliš do hloubky), pak bez ohledu na to, jak složitou úpravu záhlaví vymyslíme, vždycky dojdeme na úroveň zanoření, kterou prostě nebude možné v záhlaví zohlednit. Designér pak musí provést změnu a rozhodnout, která informace přinese čtenáři nejvíc užitku. Vynechávat lze na obou koncích škály: může se stát, že znalost názvu kapitoly je méně důležitá než vědět, ve kterém oddíle či pododdíle, pod-pododdíle nebo pod-pod-pododdíle se právě nalézáme; nebo se může stát, že vědět název kapitoly se ukáže být daleko důležitější než název pod-pod-atd. Na tomto problému musí spolupracovat úpravce s autorem.

Existuje však ještě jeden mechanismus, který je podstatně méně využíván a který přitom dovoluje do záhlaví dostat dvojnásobný objem informací: přeneseme-li číslování stran do paty stránky a uvolníme-li tak vnější okraj záhlaví k jinému použití, pak za předpokladu, že autor dá svým kapitolám, oddílům atd. *krátká* jména, bude moci každé záhlaví posloužit dvojím způsobem. Například rubová záhlaví mohou mít v sobě (vlevo) název a číslo kapitoly a zároveň (vpravo) název a číslo oddílu; lícová záhlaví pak mohou obsahovat (vlevo) pododdíl a (vpravo)

pod-pododdíl. Mezi oběma prvky v záhlaví je třeba pochopitelně nechat adekvátní prostor, aby se předešlo potenciálním nejasnostem.

Konečně pak: je *název* nebo *číslo* každého logického celku tím, co by se mělo objevit v záhlaví? Výše jsem uváděl výhradně odkazy k názvům i číslům zároveň, ale přesto musí občas docházet v určitém bodě k rozhodnutí. Dovoluje-li to místo a spolupracuje-li autor krátkými názvy, pak není důvod, aby se *obě* tyto informace v záhlaví neobjevily; při nedostatku místa nebo jsou-li názvy delší, může být nevyhnutelné vynechat čísla, aby se mohly objevit alespoň názvy; a je-li autor v pojmenování různých celků nepoučitelně mnohomluvný, pak může zůstat úpravci na výběr jednoduše jen název hierarchicky vyššího celku (např. *Kapitola, Oddíl*), za nímž následuje příslušné číslo. Toto řešení však slouží spíše autorovi než čtenáři a na autora by měl být vykonán nátlak, aby poskytl vhodné „zkrácené tvary“ čistě pro použití v záhlavích. Některá díla samozřejmě jednotlivé oddíly *pouze* číslují; v těchto knihách nemáme na vybranou: jména hierarchických celků (pokud se to hodí) a příslušná čísla.

6 Závěrem

Na dobrou knižní úpravu se lze dívat ze tří různých hledisek, kterými jsou *jednotnost*, *informace* a *struktura* (třebaže je zde mnoho dalších parametrů, na které by bylo možné v delší přednášce upozornit). Pozornost ke každému z těchto hledisek při výtvarném návrhu knihy zmnohonásobí potenciální hodnotu knihy pro čtenáře. Praktičtější rady následují v druhé části této přednášky.

Knižní úprava pro uživatele TEXu. Část druhá: Praxe

Philip Taylor

Abstrakt: V předchozí části této přednášky jsme se zabývali třemi základními faktory knižní úpravy, kterými jsou *jednotnost*, *informace* a *struktura*. Na každý z nich jsme získali jistý obecný pohled. V této části přednášky poskytneme několik praktičtějších rad, obzvláště ve dvou oblastech: poučení o skutečných rozměrech, proporcích a grafické úpravě; a dále probereme, jak některé z těchto myšlenek převést do praxe pomocí jazyka TEX. Nakonec probereme některé obtížné (a dokonce neřešitelné) problémy grafické úpravy.

Klíčová slova: Design, typografie, grafická úprava

Motto: V každé knize najdeme místo, které je nedosažitelné pomocí odkazového aparátu.

1 Jak velká je kniha?

Podobně jako je nám všem známa obecná idea knihy, chápeme všichni instinktivně i praktické horní a spodní hranice jejích rozměrů; kniha o rozměrech tři krát dva centimetry je pro většinu z nás stejně nepoužitelná jako kniha měřící tři krát dva metry. Podívám-li se při psaní na své police s knihami, pak pominu-li těch několik svazků, jejichž rozměry se zcela vymykají, mohu bezpečně tvrdit, že většina „normálních“ knih se nachází v rozmezí velikostí od 18×10 cm do 35×25 cm. Vyjádřeno v tradičnějších tiskařských jednotkách (pica — přibližně cicero): od $42\,\text{pc} \times 24\,\text{pc}$ do $80\,\text{pc} \times 64\,\text{pc}$ (ve všech případech se jedná o přibližný odhad, nikoli o nějaké přesné měření). Co je však zajímavější, je vzájemný poměr výšky a šířky u každé z těchto knih: téměř bez výjimky mají všechny knihy formát spíš na výšku než na šířku. Proč by tak tomu mělo být?

Na tuto otázku existují, domnívám se, dvě odpovědi; jedna zcela praktická a druhá trošku teoretická. Praktickou odpověď lze lehce demonstrovat: vezměte si jakoukoli knihu, jež nemá formát na výšku (tedy takovou, která má formát na délku), uchopte ji do jedné ruky a pokuste se ji

otevřít: je-li kniha malá, nebo spíš čtvercová než výrazně protáhlá, bude v ruce docela rozumně stabilní, avšak je-li velká nebo výrazně protažená do délky, bude mít tendenci se rozvírat, jelikož těžiště obou polovin výrazně přesahuje rozpětí otevřené dlaně. Pro některé typy knih (tedy ty, u nichž se předpokládá, že budou čteny v lavici nebo za stolem, případně budou spočívat otevřené na čtenářově klíně) to nemá příliš velký význam; ale ty knihy, které se budou nejpravděpodobněji číst při uchopení v ruce (což zahrnuje velkou většinu vydávaných knih), činí takováto nestabilita téměř nečitelnými, takže obecně se takovému poměru výšky a šířky vyhýbáme.

Teoretický důvod pak odkazuje k tomu, co jsme probrali v předchozí části přednášky, a týká se optimální délky řádky. Tvrdili jsme zde, že cílem je dostat se do rozmezí mezi čtyřiceti a sedmdesáti znaky na řádku, přičemž ideál leží někde u horní hranice tohoto rozmezí. Předpokládáme-li, že lidé s normálním zrakem bez potíží čtou devíti až dvanáctibodové písmo z normální vzdálenosti, ze které se čtou knihy, avšak cokoli menšího čtou již s jistými potížemi (a cokoli většího mají sklon považovat za „ponižující“ v tom smyslu, že kniha vypadá, jako by byla určena pro děti), pak to znamená, že většina knih bude směřovat k *délce řádky* někde mezi dvanácti a třiceti pica, ale spíš se bude blížit horní hranici tohoto rozmezí. Srovnáme-li tento předpoklad s rozměry knih uvedenými výše, zdají se být tyto výpočty rozumné; nejmenší knihu jsme uvažovali 24 pc na šířku (měřeno na obálce), zatímco u největší měl stejný rozměr 64 pc. Vezmeme-li v úvahu ořez stran vložených uvnitř obálky a „rozumné“ okraje (jejichž „rozumnost“ ještě budeme definovat), zjišťujeme, že délka řádku u nejmenší knihy je 17 pc, zatímco největší má řádku 48 pc (a je vysázena abnormálně velkým písmem; bylo by užitečnější takovou velkou knihu sázet ve dvou sloupcích). Zcela zřejmě zde existuje rozumný vztah mezi teorií a praxí.

V praxi jsou některé rozměry „vhodnější“ než jiné; tradičně jsou knihy tištěny v omezeném výběru formátů a termíny pro tyto druhy formátů jsou dodnes používány; například „kvarto“, „folio“ atd. Jiné, kupříkladu „čtvrtka“ nebo „royal“, se přestaly používat, navíc dnes panuje při výběru konečného formátu knihy daleko větší svoboda. Nicméně význam tu mají — jako všude jinde — i praktické skutečnosti; a tiskař nakonec bude muset vyrobit stránky knihy dělením mnohem většího listu papíru; a protože tyto větší listy jsou vyráběny ve fixních velikostech, je zřejmé, že některé finální stránky skončí s mnohem menším odpadem než jiné, a takovým rozměrům je pak pochopitelně třeba dát přednost; váš tis-

kař vám „ideální“ rozměr stránky jistě poradí, požádáte-li ho, a téměř jistě vám dopředu oznámí, jestli vámi navrhovaný rozměr nepovede ke zbytečně velkému odpadu.

Při rozhodování o rozměrech knihy jde v podstatě o tři proměnné: celkovou plochu textu včetně záhlaví a patek, okraje a konečné rozměry stránky po ořezu. Je zřejmé, že dvě z těchto proměnných můžeme libovolně určit a třetí pak jednoduše vyplývá z aritmetických a geometrických výpočtů. V praxi se člověk kloní (je-li mu dána naprostá svoboda) k tomu, že rozhodne o konečné velikosti stránky po ořezu a o rozměrech textu, a pak spočítá velikost okrajů jako rozdíl mezi těmito hodnotami; děláme-li to ale tímto způsobem, je třeba si uvědomit, že okraje jsou stejně důležité jako kterýkoli jiný prvek hotové stránky a nemohou se jednoduše podřídit libovolnému našemu rozhodnutí. „Dostatečné, ale ne příliš velké“ je vynikající axiom, na který bychom měli pamatovat při rozhodování o velikosti okrajů; například malá kniha, jejíž šířka po ořezu je 23 pc, by mohla mít vnější okraj 3 pc a délku řádku 17 pc; skutečný rozměr vnitřního okraje tak bude rovněž 3 pc, avšak *viditelný* vnitřní okraj bude poněkud menší, neboť jeho část pohltí vazba. Obecně platí, že čím je kniha silnější, tím větší je viditelná ztráta vnitřního okraje, ale technika vazby je ještě daleko důležitější, neboť dobře svázaná objemná kniha může přijít o méně místa na vnitřním okraji než mizerně svázaná kniha tenká.

Jak se zvyšují celkové rozměry knihy, měly by se zvětšovat i okraje; nerostou však stejně s formátem stránky: spíš, pokud vůbec, by se měly rozšiřovat docela pomalu, přibližně s koeficientem druhé odmocniny zvětšení rozměrů stránky nebo s jeho logaritmem. Zopakujme si, že „dostatečné, ale ne příliš velké“ by mělo platit jako pravidlo.

Dosud jsme se soustředili na vnitřní a vnější okraje; stojí za to, dříve než se budeme věnovat horním a spodním okrajům, poukázat na skutečnost, že pokud vyžadujeme okraje vnímané jako symetrické, rozhodně musíme navrhnout okraje nesymetrické; jenže tato asymetrie se na lichých a sudých stranách mění. To znamená, že chceme-li ponechat místo pro ztrátu ve vazbě, musíme o tuto ztrátu zvětšit pravý okraj sudých stránek a levý okraj stránek lichých (pravých). Toho lze dosáhnout automaticky v „knižním stylu“ LaTeXu, ale uživatelé plain TeXu budou potřebovat modifikovanou výstupní proceduru. Abychom nepotřebovali znát nic o existující výstupní rutině, používá následující část programu primitiv `\shipout`, a lze ji tedy použít ve spojení s *jakoukoli* výstupní rutinou (bez ohledu na její složitost), pokud ovšem tato rovněž nepra-

cuje průběžně s parametrem `\hoffset` (v tom případě by se musel použít program trochu chytřejší).

```
\newdimen \rectohoffset
\newdimen \versohoffset

\def \bindingloss {2 pc}  %%% nastavitelné pro danou knihu
\let \Shipout = \shipout  %%% nutná ekvivalence umožňující
                          %%% vypůjčit si primitiv
\let \then = \relax       %%% jen syntaktický bonbónek
                          %%% (promiň, Keesi!)

\rectohoffset = \hoffset \advance \rectohoffset by
                                          \bindingloss
\versohoffset = \hoffset \advance \versohoffset by
                                         -\bindingloss

\def \shipout
    {\ifodd \count 0      %%% nelze použít \pageno, neboť
     \then                %%% není v LaTeXu funkční
            \hoffset = \rectohoffset
     \else
            \hoffset = \versohoffset
     \fi
     \Shipout
    }
```

Než začneme uvažovat o skutečných rozměrech vertikálních okrajů, vyplatí se položit si jednodušší otázku týkající se proporcí. Zde se objevují jako u mnoha prvků knižní úpravy názory zhruba dvou škol: první bude zastávat názor, že horní okraj by měl být menší než spodní, a druhá právě opačný! Argumentace je v obou případech založena na vizuální vyváženosti: ti, kteří by blok textu posunuli asymetricky směrem nahoru, tvrdí, že z vizuálního hlediska celý blok „padá dolů svou vlastní vahou“, zatímco druhá škola zastává názor, že pokud sazba nemíří asymetricky ke spodnímu okraji stránky, vypadá posazená příliš nahoře a tudíž vratce. Mým přesvědčením je, že pokud zahrneme do své úvahy záhlavní a patní řádky, můžeme do jisté míry obě školy smířit; nicméně pokud záhlavní

a patní řádky nemáme, pak se mé sympatie kloní spíše ke škole tvrdící, že „nižší je lepší“, než k jejich oponentům.

Důvodem pro zahrnutí záhlaví a patních řádků do našich úvah o okrajích je skutečnost, že zatímco levé a pravé okraje jsou podle mne „jednoduché“ (tzn. že každý z nich zabírá jednoduchý pruh bílého místa), horní a spodní okraje jsou ve skutečnosti složené: je zde bílé místo nad záhlavím, bílé místo pod záhlavím, a stejně tak bílé místo pod i nad patním řádkem (je-li přítomen; není-li, pak je spodní okraj jednoduchý). Mluvíme-li ale o vizuální hustotě, patní řádek je obvykle velice světlý — často je to jen číslo strany bez jakéhokoli ornamentu — zatímco záhlaví je obvykle dost husté (viz první část přednášky, kde je rozebrán možný obsah záhlavního řádku). Výsledným efektem této skutečnosti je, že dva spodní okraje jsou okem a mozkem přijímány jako jeden pruh bílého místa, zatímco horní okraje vnímáme jako dva rozdílné celky. Oko a mozek tudíž přijímají součet dvou spodních okrajů jako jediné bílé místo ve spodní části stránky, zatímco nižší ze dvou horních okrajů mají sklon spíše ignorovat a spatřovat bílé místo pouze v horní komponentě okraje.

Pokusme se nyní sumarizovat předchozí rozbor a přijít s nějakými pevnými doporučeními. Obecně vzato je prostor nad záhlavním řádkem výrazně větší než prostor pod ním a měl by být přibližně stejný jako levý a pravý okraj (za předpokladu, že tyto nejsou přehnané; výrazně širší okraje probereme později v této kapitole). Prostor pod záhlavním řádkem je docela malý: snad 1 pica nebo tak nějak. Ve spodní části stránky je situace obrácená: zde je relativně málo místa nad patním řádkem, zato poněkud více místa pod ním. Zde je však třeba upozornit: pokud necháme nad patním řádkem stejné místo jako pod záhlavím (např. 1 pica), ocitneme se během typografické úpravy stran v tísni, neboť i když bude moci kterákoli strana mít o řádek méně, nebude moci přetéct o řádek navíc, aniž by se nespojila s patou (nebo v horším případě neposunula patní řádek směrem dolů); je tudíž nezbytné, abychom nad patním řádkem ponechali na normální stránce více místa, aby *in extremis* bylo možno stránku o jeden řádek „přešvihnout“. Tudíž je nad patním řádkem přiměřená mezera velikosti asi 2 pica, a další tři až čtyři pica dole. Mějte na paměti, že tyto vývody představují pouze prvotní přiblížení, ale že ke značně rozsáhlým variacím ve velikosti stránek budou třeba pouze relativně drobné úpravy.

Až do této chvíle veškerá diskuse o okrajích odrážela velice tradiční, ortodoxní a konservativní názory. Avšak velikost a symetrie okrajů je jednou z těch oblastí, kde *avantgardní* úpravci cítí povinnost projevit

svou individualitu. Až do nástupu takzvané „revoluce DTP systémů“ měla většina knih konzervativní okraje podle výše zmíněných pravidel; ale přibližně v době, kdy DTP doznal značného rozšíření, zjistila náhle nová generace návrhářů, že je třeba přijmout značně široké okraje, někdy bez jakékoli rozumné proporce vůči ostatním věcem na stránce.[1] Důvod tohoto náhlého zájmu o široké okraje je snad docela zajímavý, ale mám podezření, že mu není dobře rozuměno. Umím si představit několik možných důvodů: (1) Každá generace návrhářů cítí povinnost vyjádřit svou tvořivost nějakým výrazným, viditelným způsobem; jednoduché následování svých předchůdců je v nejlepším případě považováno za parafrázi a v nejhorším za plagiát. (2) Osvobodivý efekt toho, co nazvu „Grafická úprava skrze DTP“[2] dovolila úpravcům experimentovat s knižním designem takovým způsobem, který by byl dříve smeten ze stolu, ať už proto, že by marnotratná podstata jejich extrémů byla příliš zjevná, jakmile by se vyrobil skutečný papírový model návrhu, nebo prostě proto, že časová prodleva mezi vytvořením návrhu a jeho prvním zhmotněním poskytovala úpravci čas k sebereflexi; mnozí by — o tom jsem přesvědčen — své výstřelky během této zchlazovací periody potlačili. (3) Spousta těchto úprav byla vytvořena za použití prvních DTP systémů, které měly při grafickém navrhování stran značná omezení samy o sobě; mít široké okraje, kam mohly v případě potřeby přetéct nadbytečné prvky, dovolovalo návrhářům další flexibilitu při práci v rámci omezení, která měl systém DTP zabudována v sobě.

Existuje však ještě čtvrtý důvod, zcela nezávislý na DTP revoluci, který rovněž může diktovat užívání širokých okrajů; tyto závěrečné poznámky k problematice okrajů se budou vztahovat výhradně ke grafické úpravě a problémům s ní spojeným. Text, tabulky, grafika, rovnice a definice mají vždy rozdílné, a někdy vzájemně konfliktní, požadavky; text, jak jsme viděli, bude nejlépe čitelný při délce řádky někde mezi 12 pc a 30 pc; tabulky obsahující několik sloupců se nemusí do takto omezených řádků dobře vejít, problémy mohou nastat i se složitějšími obrázky

[1] Je smutným znamením našich časů, že se toto přihodilo v době, kdy se neustále rozšiřuje obecné povědomí o ekologických dopadech ze ztráty světového lesního porostu; takže máme na jedné straně environmentalisty nutící nás šetřit stromy, zatímco zároveň nám vyrostla generace návrhářů zcela zjevně zaměřená na zničení světových lesů čistě proto, aby mohli ve svých knihách udivovat širokými asymetrickými okraji...

[2] Mám tím na mysli přístup k užívání počítačů Apple MacIntosh nebo podobných systémů k produkci obrazovkové makety předběžného návrhu bez jakékoli potřeby její fyzické realizace.

(které, i když se obecně dají zmenšit, mohou se stát při příliš velkém zmenšení nečitelné); rovnice a definice také mohou vyžadovat místo širší než 30 pc, pokud se nemají rozdělit do více řádků. S výjimkou rovnic a definic není tento problém neřešitelný ani snad zvlášť obtížný: kde je dopředu známo, že bude třeba délka řádky o dost větší než 30 pc, je možno text vysázet ve dvou sloupcích s tím, že příliš široké tabulky a obrázky necháme přesáhnout do obou sloupců; jelikož tabulky a obrázky jsou obecně považovány za „pohyblivé“ jednotky (což znamená, že je možné s nimi v textu volně pohybovat, aniž by způsobily čtenáři obtíže, neboť odkazy k nim jsou uváděny téměř vždy pomocí čísla nebo jména spíš než jen prostou fyzickou přítomností), mohou se objevovat na stránce zcela samostatně, v horní nebo spodní části strany, která se k nim vztahuje, aniž by narušily plynulost textu. Na rozdíl od toho rovnicím a definicím (a podobným věcem, jako jsou části počítačových programů nebo algoritmů) *nesmí* být většinou dovoleno volně se pohybovat: autor píše téměř vždy text tak, že předpokládá, že se rovnice a definice objeví vždy přesně tam, kde jsou i v rukopise, a jednoduše nechává svůj text „plynout“ k rovnicím nebo definicím; pokud je takováto rovnice nebo definice příliš dlouhá a nelze ji zkrátit, pak je třeba přerušit oba sloupce textu, ovšem ke značné nespokojenosti čtenáře, neboť nemusí být zcela zřejmé, zda se text má číst až k výrazu a pak pokračovat pod ním v témže sloupci, nebo zda je třeba dočíst až k rovnici či definici a pokračovat potom nahoře ve druhém sloupci. Ještě horší to je, pokud se rovnice či definice nevyskytuje v prvním sloupci textu, nýbrž ve druhém, poněvadž čtenář postupuje dolů v prvním sloupci a náhle je zastaven ve své cestě naprosto irelevantním výrazem; nejenže nemusí čtenář vědět, kde má pokračovat, nevidí zpočátku ani důvod, proč k přerušení došlo. Teprve když si přečte i druhý sloupec, důvod přerušení se mu ozřejmí.

V takovýchto dílech je tedy třeba zvolit jiný přístup, a jedním z těchto alternativních přístupů je volba přehnaně širokých okrajů: text je vysázen v rozumně širokém jediném sloupci, avšak ořezová míra stránky je taková, že dovoluje, aby nejdelší rovnice či definice přesáhly do (většinou pravého) okraje, je-li to nezbytné. Návrhář pak stojí před jiným problémem: jak ospravedlnit čtenáři přítomnost těchto okrajů na stranách, kde se tyto rovnice či definice nevyskytují. Rozhodně by nemělo být v těchto případech výjimečné najít záhlaví oddílu vytištěné za těchto podmínek do okrajů stránky nebo zde objevit poznámky, které se jinak mohou objevit dole na stránce nebo dokonce jako poznámky na konci

knihy. Cokoli, co může ospravedlňovat přítomnost anomálního okraje, je považováno za přípustné!

Nakonec k mezeře uprostřed stránky: vnitřní „okraje“, které ve vícesloupcové sazbě oddělují od sebe jednotlivé sloupce. Obecně řečeno neměla by být mazera širší než střední míra levých a pravých okrajů; může být spíš poněkud užší. Někteří návrháři dávají přednost při vertikálním dělení pravidlu úzké mezery; měl bych sklon se tomuto vyhnout, pokud nejsou tato pravidla použita i jinde v daném návrhu. Podobně jako na mnoha jiných místech i zde lze dosáhnout vynikajících výsledků respektováním jednotnosti.

2 Jednotlivé prvky knihy

Když jsme vytvořili pravidla pro celkové rozměry naší knihy, je nyní správné probrat nejrůznější elementy, z kterých se kniha skládá. Na nejpovrchnější úrovni (a s odhlédnutím od desek, hřbetu a přebalu) je kniha tvořena *předními stranami* (někdy se jim říká též „vstupní“), *textem* a *zadními* neboli *koncovými stranami* (což je poněkud zavádějící, neboť kniha má dva konce, avšak tradičně se užívají „koncové stránky“ spíše než jednoznačnější „zadní stránky“).

Vstupní stránky jsou tvořeny takovými prvky, jako je patitul a titulní strana, copyright a katalogizace publikace, obsah (a někdy další obdobné tabulky) a případně i předmluva. Často je rovněž mezi vstupní stránky zařazována (obzvlášť s příchodem DTP revoluce, jelikož všichni daleko víc vnímáme typy písma a typografii obecně) „tiráž“, která by se přísně vzato měla objevit jako poslední část knihy, dnes však stále častěji zaujímá místo na stránce s copyrightem a katalogizací publikace; tiráž obsahuje podrobnosti o sazbě a použitém typu písma a rovněž v ní mohou být informace o tvůrci designu, tiskárně apod.

Mezi zadními stranami najdeme dodatky, jeden nebo více rejstříků, bibliografii (pokud ta není zvlášť u každé kapitoly nebo pokud je žádoucí mít celkový přehled biografie na konci stejně jako partikulárně na konci každé kapitoly), případně nějaký poznámkový aparát apod.

Konečně vlastní tělo knihy tvoří samotný text, obvykle dělený do jednotlivých kapitol. Může být rozdělen i na vyšší úrovni do částí.

Myslím, že lze říct, že hranice mezi těmito třemi zónami nejsou naprosto pevné: autor si může přát, aby předmluva byla spíš součástí textu než vstupních stran, což je třeba zohlednit v číslování stran, jak ještě uvidíme. Podobně mohou někteří spisovatelé považovat za důležitou součást textu i dodatky; i toto může ovlivnit číslování stran, ale bývá to méně

pravděpodobné. Ve skutečnosti se autor může rozhodnout napsat předmluvu, prolog, úvod, závěr, epilog a jeden nebo několik dodatků; návrhář pak musí spolu s autorem pečlivě probírat, zda jsou všechny tyto části správně utříděny.

Hlavní důvod pro takového rozdělení spočívá v číslování stran: vstupní strany bývají tradičně číslovány *římskými* čísly, přičemž se používají malé římské číslice (i, v, x, l, c, d, m), které jsou sázeny jako spuštěná čísla (dole uprostřed paty), zatímco hlavní text je obvykle číslován *arabskými* číslicemi (0, 1, 2, 3, 4, 5, 6, 7, 8, 9). Dodatky a ostatní zadní strany většinou pokračují sekvenčně a ve stejném stylu jako hlavní text, ale je dovoleno začít u dodatků s novým číslováním stran, kterému předřadíme písmeno „A“; využijeme-li této možnosti, rejstřík (za předpokladu, že rejstříky budou posledním elementem zadních stránek) nebude mít číslované stránky vůbec, poněvadž by bylo zcela zjevně nepřiměřené pokračovat v číslování s předznamenáním „A-“, přičemž stejně nepřiměřené by bylo vrátit se k hlavnímu číslování. Naštěstí se u rejstříků nevyžaduje, aby byly samovztažné (třebaže musím přiznat, že jsem jednou doplnil jeden rejstřík, jehož sloupce by jinak nešlo vyrovnat, naprosto falešným odkazem k termínu „loop, infinite“, přičemž číslo stránky u tohoto odkazu bylo číslem téže stránky, na níž se heslo v rejstříku nacházelo...).

Existují rovněž konvence, podle kterých se různé prvky mají objevovat na pravých a jiné na levých stránkách, nebo před kterými má být vložený vakát. Typická kniha může být číslována následujícím způsobem (připomínám, že lichá čísla označují pravé a sudá levé stránky):

i. patitul;
ii. vakát;
iii. titul;
iv. katalogizace publikace, copyright, tiráž;
v. předmluva k vydání;
vi. obecná předmluva;
vii. ditto, pokračování;
viii. vakát;
ix. obsah;
x. ditto, pokračování;
xi. slovníček;
xii. vakát;
1 první kapitola.

Z těchto věcí je nutné, aby se patitul a titul objevily na lichých stranách (přičemž vakát vložený mezi nimi umožňuje pěkný kontrast vůči

plné titulní straně); copyright spolu s katalogizací publikace se často objevuje na rubu titulní strany; předmluva nemusí nutně začínat vpravo, může to být ale návrhářovým přáním; obsah bývá normálně vpravo, jako je tomu zde; první kapitola musí v každém případě začínat na liché straně a až na ty nejvolnější typografické styly musí i všechny kapitoly následující začínat rovněž lícovou stranou. Číslo první stránky první kapitoly by mohlo být zrovna tak „13“; jde o rozhodnutí v rámci designu, zda pokračovat v číselné řadě ze vstupních stran nebo začít v hlavním textu nanovo.

Zadních stran se týká méně konvencí, ale bývá zvykem, aby *první* dodatek začínal na lícové straně; další dodatky mohou v případě nezbytnosti začínat vpravo i vlevo; rejstřík by měl rovněž normálně začínat na liché straně.

3 Typografická úprava stránek

Třebaže zdaleka největší počet stran v knize tvoří „normální“ stránky, vyžaduje jistý cit začít s rozvahou u prvních stránek úvodní kapitoly, jelikož ty přispějí velkým dílem k vizuální identitě knihy a dovolí správný stupeň umělecké licence při své tvorbě. (Je ale též vhodné upozornit, že člověk může při jejich tvorbě strávit zbytečně mnoho času marnou snahou je dobře navrhnout!)

Při navrhování knihy bývá v každém případě nezvyklé, aby lidé umístili záhlaví kapitoly (nechť je to třeba „Kapitola první“ nebo „Úvod“) na stranu úplně nahoru. U některých knih, obzvlášť u těch s velice krátkými (kratšími než dvoustránkovými) kapitolami to může hrát nebývalou roli, neboť jinak může člověk použít daleko víc stran, než je nezbytně nutné (existují však i estetické důvody, proč je za těchto podmínek třeba dát takovému designu přednost). Avšak naprostá většina knih má kapitoly, u nichž počet stránek často mívá dvouciferné hodnoty, a pro tyto knihy bývá zvykem (třebaže to není nutné) začínat nadpisem kapitoly trochu níž na stránce. Typickým příkladem je rezervování místa nad nadpisem odpovídající jedné čtvrtině až třetině délky stránky.

Dále pak vzniká otázka, co dát do nadpisu. Jsou-li kapitoly číslovány, musí se člověk rozhodnout mezi „Kapitola první“, „Jedna“, „1“ nebo mezi dalšími podobnými variantami; pokud má název, je třeba se rozhodnout, zda použít i číslo nebo pouze název (a pokud obojí, pak jaký styl číslování zvolit). Panuje názor, že „Kapitola první“ je poněkud staromódní, ale já tento názor nesdílím. Používáme-li jak čísla, tak názvy a používáme-li pouze arabské číslice, pak je rovněž možné dát obojí do

jednoho řádku, případně je oddělit dvojtečkou a místem odpovídajícím jednomu řádku; jsou-li číslo a název na dvou různých řádcích, pak bývá zvykem, aby číselný údaj předcházel názvu.

Dále je tu otázka písma: jaké písmo má tvořit nadpis? Téměř ve všech případech použijeme velký tučný font, ovšem ona „velikost“ je z velké části v oku diváka; snad lze říci, že LaTeX užívá k tomuto účelu poněkud větší fonty, než by si konservativnější návrháři možná vybrali. Použití *bezpatkového* písma pro tyto nadpisy je zcela zřejmě doporučeníhodné, nikoli však nezbytné.

Umístění: mají být nadpisy centrovány nebo zarovnány na zarážku doleva (nebo dokonce zarovnány doprava)? Obecně řečeno, centrované nadpisy mají buď lehce staromódní příchuť nebo se víc hodí pro umělecká díla; moderní vědecké publikace používají nejčastěji levé zarovnávání na zarážku, které prolíná celou knihou, včetně takovýchto nadpisů. Zarovnání doprava pravděpodobně bude vypadat *avantgardně*, ale nelze ho v tomto případě vůbec nebrat v úvahu; použije-li se, měly by být asi v designu přítomny další prvky, odkazující k „pravému“ tématu, nebo by mělo být pro kontrast a rovnováhu užito i zarovnání doleva. Je-li použit epigram, pak je pravděpodobně lepší mít nadpisy zarovnány doleva a epigram vpravo, poněvadž opak by znamenal nadměrné zdůraznění epigramu vůči titulu kapitoly.

Existuje i další prvek umístění textu, který rovněž stojí za povšimnutí: má být bílé místo nad nadpisem považováno za součást nadpisu nebo stránky? Mám tím na mysli následující věc: jestliže titul kapitoly normálně zaujímá n řádků (obvykle jeden nebo dva), avšak patologicky dlouhý titul jedné jisté kapitoly vyžaduje jeden nebo dokonce víc řádků navíc, odkud se mají tyto řádky vzít? Má se dovolit, aby se nadpis posunul na stránce směrem *nahoru* a vstoupil do vyhrazeného bílého prostoru, nebo *dolů* na stránce, kde posune pod sebou výchozí bod hlavního textu? Ani jedno z toho není ideální, ale pokud autoři budou trvat na psaní patologicky dlouhých nadpisů, jedno nebo druhé řešení musíme zvolit. Třebaže následující tvrzení není tesáno do kamene, stojí snad za povšimnutí: pokud úvodní strana kapitoly začíná řádkem obsahujícím pouze *číslo* kapitoly (nebo slovem „Kapitola“ s číslem za ním), pak by se mělo *vždycky* objevovat v přesně stejné výšce (a hlavní text se tak posune směrem dolů); jestliže však stránka začíná *názvem* kapitoly, pak můžeme připustit, aby se titul vysunul směrem nahoru, a zajistit tak, aby hlavní text začínal vždy ve stejné výšce na stránce.

A čáry: mají být nadpisy odděleny od textu nějakou horizontální čarou? Zde je pravděpodobně potřeba vrátit se k našemu tématu jednotnosti: jestliže čáry tvoří stále se vracející téma knihy, pak čára oddělující nadpis od textu bude asi vyhovovat; pokud ne, může působit rušivě.

Nakonec, než zcela opustíme téma úvodních stran kapitol, mohlo by být užitečné zrekapitulovat rady, podané v předchozí přednášce ohledně záhlaví a číslování stran: obecně vzato, řádek záhlaví nemá na první stránce kapitoly co pohledávat; bílé místo nad nadpisem by se mělo nerušeně spojit s horním okrajem. Což znamená, že číslo stránky, pokud je normálně na vnějším okraji záhlaví, musí být (na úvodní straně kapitoly) buď úplně vypuštěno nebo být přeneseno do paty. Vynechání čísla strany je v nejvyšší míře nežádoucí, neboť to činí obsah naprosto zbytečným (a rovněž potlačuje užitečnost rejstříku, pokud nějaké heslo rejstříku odkazuje k úvodní stránce kapitoly); řešením je tudíž spuštěné číslo stránky, vycentrované v patním řádku. Někdy je takovéto číslo dozdobeno nějakým drobným ornamentem, například rozdělovníkem z obou stran odděleným tenoučkým proužkem místa; třebaže je tato konvence vzata přímo z praxe běžné u psacího stroje, dělá — alespoň podle mínění autora — z čísla na stránce trošku samozřejmější věc, a má tudíž cosi do sebe.

Když jsme skončili s úvodními stranami kapitol, je dalším nejdůležitějším prvkem v knižním designu normální strana textu; tyto strany obvykle tvoří víc než 90 % knihy a je tedy záhodno vyvinout patřičné úsilí k tomu, aby vypadaly „správně“. Zabývali jsme se již okraji, mezerami, záhlavím a patou, takže se nyní můžeme soustředit na samotný text, obzvlášť pak na použité písmo a rozpal řádků.

4 Písmo a rozpal řádků

Jak již bylo naznačeno výše, text bývá normálně vysázen desetibodovým *patkovým* písmem, často s řádkováním dvanácti bodů (zde dává plain TEX přinejmenším smysluplné definice, pokud není používán nadměrně dlouhý řádek). Zdá se, že je značně rozšířena víra, že tím správným fontem je Times Roman, nicméně toto písmo tak, jak v základní podobě vypadá, je navrženo pro výjimečně krátké řádky úzkých novinových sloupců a nemá jinou výhodu, než širokou, snadnou dostupnost. Je to font příliš úzký pro řádnou délku řádky většiny knih, a pokud už *musí* být použit, lze výrazně těžit z anamorfního zmenšení písma činitelem 24/25 ve vertikálním směru. Takovéto zmenšení, pro puristy nepřípustné, přetvoří poněkud úzké typy Times Roman do kulatějších, měkčích tvarů a umožní téměř optimální kombinaci velikosti fontu a řád-

kového rozpalu pro řádky dlouhé až 27 pc. Z písma Times Roman velikosti 11/12,5 vznikne anamorfním zmenšením činitelem 24/25 písmo velikosti 10,56/12, což je podle názoru autora vysoce čitelný text.

Ovšem daleko lepší než anamorfní zmenšování Times Romanu je vybrat si písmo, které už má potřebné vlastnosti (zakulacené typy písma, vhodnost použití v dlouhých řádcích atd.) zahrnuté v sobě; příkladů jsou tucty, ale mezi nejsamozřejmější kandidáty patří Baskerville, Bembo, Caslon, Garamond a Palatino. Vyhnout bychom se měli fontům příliš výstředním: měli byste pamatovat, že *jedinou* funkcí písma je sdělení informace; je-li čtenář odváděn výstřední podstatou zvoleného fontu, přenos informace nebude optimální a výsledkem je znehodnocení knihy jako takové.

Snad v této chvíli stojí za to krátce odbočit a pohovořit o jedné zvláštní knize, se kterou jsem se poprvé setkal, když jsem byl požádán o její recenzi. Jedná se o Knuthovu *3:16 Bible Texts Illuminated*. Mojí první reakcí po otevření knihy bylo položení rétorické otázky sobě samému: „Proč to proboha vysázel v Computer Modern?“. Znal jsem Computer Modern dobře z pětisvazkové antologie *Computers and Typesetting* a sám jsem spoustu svých vlastních textů v Computer Modern vysázel, když jsem se učil TEX; dosáhl jsem však už bodu, kdy jsem cítil, že jiné fonty mi toho mohou nabídnout daleko víc, a nějaký čas jsem už v Computer Modern nedělal vůbec žádnou sazbu; tudíž mi přišlo velice ošklivé vidět knihu zabývající se studiem Bible celou vysázenou v Computer Modern, obzvláště někým, jehož názorů jsem si tak hluboce vážil.

A přesto když jsem přečetl ani ne pět stran knihy, stala se podivná věc: náhle jsem zjistil, že vůbec nevnímám typ písma; zmizel z něj jakýkoli jiný záměr a účel a font se stal čistě médiem komunikujícím fakty. Je třeba říct, že Computer Modern, založený na fontu Monotype 8a, není pro každého zrovna ideálním typem písma; a obzvláště na zařízeních dávajících slabší výsledky (jako např. laserové tiskárny) může být opravdu dost nepříjemný, s těmi přerušenými nebo dokonce zcela mizejícími slabšími tahy, jevícími se tak trochu mimo správné proporce. Když se však použije na sazečském zařízení vysoké kvality, kontrast mezi silnými a tenkými liniemi velice přispěje k estetice fontu a výsledný efekt představuje ničím nerušený design, příjemně postrádající jakoukoli výstřednost, což potlačuje vlastní osobitost písma a dovoluje informaci, aby se prodrala na povrch. Snad ani neexistuje něco takového jako špatný typ písma; to, co vnímáme jako špatné,

může být jednoduše špatně použitý dobrý font nebo nevhodná technologie.

Vraťme se však k otázkám designu a obzvlášť k designu normálních stránek textu v knize. Když jsme si zvolili svůj základní font a rozpal řádků, budeme si dále potřebovat vybrat vhodné řezy tohoto fontu pro zvláštní účely (možná budeme potřebovat zvolit si i jeden nebo více dalších typů písma ke speciálním účelům, ale obecným pravidlem by mělo být, že čím méně typů písma je v dokumentu použito, tím lépe bude dokument vypadat). Ke zvýraznění a pro cizí slova a fráze uvnitř textu je zvykem užívat kurzívu daného fontu; použití polotučného písma ke zvýraznění by mělo být přísně zavrženo, neboť tyto fonty by měly být rezervovány pro nadpisy apod. Kurzívu lze použít i pro názvy knih, lodí a dalších analogických věcí. Samosebou se rovněž rozumí, že v hladkém textu knihy nemá místo žádné podtrhávání, a nemělo by dostat příliš mnoho prostoru ani někde jinde; podobně jako je použití polotučného písma ke zdůraznění slova pozůstatkem dávných počítačových editorů (které postrádaly kurzívu a tudíž si musely vytvořit jinou konvenci pro důraz na slově), i podtrhávání je pouhým artefaktem rukou nebo na psacím stroji psaných textů a nemá místo ve vysázeném dokumentu.[3]

Je-li třeba zdůraznit slovo nebo frázi uvnitř delšího úseku již vysázeného kurzívou, bývá zvykem vrátit se pro zdůraznění k původní antikvě; autor však nevidí žádný důvod, proč by za těchto okolností nebylo možné vysázet zdůrazněnou část v polotučné kurzívě, pokud je tento řez fontu k dispozici (a s příchodem PostScriptových fontů jsou většinou tyto řezy k dispozici); je-li polotučným písmem zvýrazněná část podobná nebo kontrastuje-li s jinou částí textu knihy, která je fyzicky nedaleko, pak může být nezbytné vysázet tuto část rovněž polotučnou kurzívou, i když se vyskytuje v kontextu, kde by normálně kurzíva *neměla* být použita; tímto způsobem získá čtenář přiměřený typografický návod, které dvě části patří nějakým způsobem k sobě.

Kurzíva (což je vysoce stylizovaný řez fontu) by neměla být zaměňována skloněnými nebo zkosenými řezy, které nepředstavují žádný originální design, nýbrž jen výsledek jednoduché geometrické transfor-

[3] Samozřejmě, jako téměř každé pravidlo, i tato pravidla připouštějí výjimky, a byl by to opravdu statečný autor, který by napsal, že *každé* podtržení či užití polotučného písma pro zdůraznění textu je rozhodně špatné; maximálně lze říci, že v obecné rovině jsou takováto (zne)(po)užití pokládána za nešťastná a nepatřičná a že návrhář by měl tuto konvenci přijmout a měl by si být porušených „pravidel“ vědom a měl by nad nimi případně ohrnout nos zcela vědomě a ne jednoduše proto, že si jejich existence vědom nebyl.

mace normální antikvy. Zatímco kurzíva a zkosené (oblique) písmo mají vesměs své ctěné předchůdce (zkosené písmo je obvykle rezervováno pro *bezpatkové* písmo, zatímco kurzíva je variantou písma *patkového*), skloněné (slanted) fonty jsou, zdá se, dalším artefaktem DTP revoluce. Podle názoru autora toho mají málo co nabídnout, co se estetiky týče, a i když se někdy používají na místech, kde se zdá být vhodné odlišit typograficky dva různé celky, jimž by jinak oběma přináležela kurzíva, obecně budu stát vždy proti jejich užití. Návrháři dokázali během staletí zprostředkovat značnou sumu informací, aniž by se museli uchýlit ke zkosenému písmu; lze jen doufat, že budoucí generace návrhářů dojdou k závěru, že tyto fonty nereprezentují nic víc, než co by Fowler nazval „elegantním řezem“, a jsou tudíž luxusem, bez kterého se docela dobře obejdeme.

Někdy je nezbytné, obzvlášť v knihách o lingvistice nebo jiných tématech, v nichž je jazyk jak užíván, tak rozebírán, typograficky odlišit tato dvě různá použití. Někdy postačí jednoduše uvozovky; jindy kurzíva; jsou ovšem i situace, kdy je obé již rezervováno pro jiná typografická rozlišení a zdá se, že pouze nějaká třetí forma ozřejmí, o kterém textu se hovoří a který je tím, jímž se hovoří. V takovéto situaci (a pouze v nemnoha jiných) je ospravedlnitelné vytáhnout nový řez a použít ho jako součást tohoto textu. Je-li hlavní text vysázen *patkovým* písmem (jak by měl téměř vždy být), pak by další *patkový* řez *nebyl* vhodný; třebaže dva *patkové* řezy mohou být navzájem tak odlišné jako voda a oheň, riziko záměny je přesto natolik vysoké (a estetický střet příliš divoký), aby nám dovolilo postavit proti sobě dvě *patková* písma. Druhý řez musí mít tedy *bezpatkovou* podobu a musí být vybrán tak, aby se lehce smísil s podobou hlavního textu a zároveň přitom byl od něho snadno rozeznatelný. Tento druhý řez bude muset být podobný co do váhy (visuální hustoty), horních a dolních dotažnic.

5 Nadpisy

Mottem předchozí přednášky bylo: „Pod titulky nikdy nemůže být příliš málo místa, pouze přespříliš místa!“ A do těchto několika málo slov lze shrnout podstatnou část veškeré moudrosti týkající se nadpisů. Jak již bylo zdůrazněno, nadpis *musí* být svázán s textem, k němuž patří, což je konstantně text, který bezprostředně následuje. Nadpisy mají často hierarchickou strukturu a titulky nižší úrovně se vážou k následujícímu textu úžeji než titulky úrovně vyšší; tudíž bílého místa, které odděluje nadpisy nižší úrovně od textu, je obvykle méně (a nikdy více), než bílého místa oddělujícího od textu nadpisy úrovně vyšší. V krajním případě je

nadpis *zalomený*, tvoří tedy doslova součást textu a nemá svůj vlastní řádek. Pro zalomené nadpisy je podstatné, aby si byl autor jejich užití vědom, jelikož takovéto nadpisy mohou být buď gramatickou součástí textu nebo mohou zůstat vlastní gramatickou entitou; v prvním případě je zvykem odlišit nadpis změnou fontu (kurzívou, polotučným písmem nebo dokonce i verzálkami nebo kapitálkami), nikoli však žádným horizontálním bílým místem nebo interpunkcí; u nadpisů, které jsou od ostatního textu gramaticky odlišné, použijeme rovněž změnu fontu, ale nezřídka se používá i interpunkce (např. dvojtečka) nebo dodatečný bílý prostor (např. jeden čtverčík). Takový nadpis lze vysadit o pouhé jedno bílé místo dál od předchozího textu, a určitě ne dál než o jeden prázdný řádek.

U následující úrovně se nadpis objevuje na bezprostředně předchozím řádku, který má celý pro sebe. Nesází se s žádným přidaným bílým místem, jednoduše je jen oddělen od textu normálním rozpalem odstavce. Opět může být vyznačen změněným fontem, přičemž zde lze aplikovat stejné zásady jako u nadpisů zalomených do textu, třebaže použití verzálek nebo kapitálek by bylo neobvyklé. Pro velikost bílého místa nad tímto nadpisem platí stejná pravidla jako pro zalomené titulky.

O úroveň výš snad lze použít větší font. Předpokládáme-li, že základní text je tvořen fontem velikosti 10/12, mohl by být pro tyto nadpisy vhodný dvanáctibodový font. Použil-li se polotučný font pro nadpis kterékoli z nižších úrovní, potom i tento nadpis musí být vysázen polotučně, jinak by došlo k nejasnostem (totéž platí pro *všechny* úrovně hierarchie: jakmile byl na nižší úrovni použitý polotučný font, musí být tučné nebo polotučné fonty užity i na všech úrovních vyšších. Podobně nesmí být žádný typ písma, použitý na vyšší úrovni v uspořádání, menší než font použitý u nadpisu nižší úrovně; může být stejně veliký, ale jen pokud je polotučný a ten na nižší úrovni nikoli, nebo pokud je zde jiná zřetelná typografická indikace hierarchie). Nad takovýmto nadpisem lze dovolit trochu bílého místa navíc, snad něco v rozmezí jednoho až jednoho a půl prázdného řádku.

Od tohoto bodu postačí prostá extrapolace: jak se pohybujeme nahoru v hierarchii, nadpisy se zvětšují, ztučňují a zvýrazňují oddělováním. Bílé místo pod nimi se může zvětšit, ale jen velice málo; bílé místo nad nimi se zvětšuje, avšak ne na nějakou směšnou úroveň. Cokoli nad tři prázdné řádky je téměř s jistotou výstřední, a dva prázdné řádky musí za normálních okolností stačit.

Zde je na místě zmínit se o dopadech výše vyjmenovaných pravidel na TEXovské realizace. Aby se v TEXu umožnila úspěšná grafická úprava stránky, bývá zvykem dovolit vertikálnímu bílému místu spojenému s nadpisy být flexibilní (tzv. „gumové délky“ v LATEXovské bizarní terminologii); jenže TEX má dvě značně odlišná pravidla, když pracuje s flexibilitou: je-li zadána negativní flexibilita (je tedy dovoleno stažení), pak TEX — je-li to nezbytné — tuto výhodu předdefinované možnosti stažení využije, aby dosáhl optimální grafické úpravy strany, ale nikdy nezmenší prostor pod stanovenou mez; avšak pokud je zadána *positivní* flexibilita (tedy je dovoleno roztažení), pak TEX použije tuto flexibilitu k dosažení optimální grafické úpravy ze všeho nejdřív, a je-li tato flexibilita nedostatečná, *bude pokračovat v roztahování, dokud se nedosáhne optimální grafické úpravy*, i když to třeba znamená mnohonásobně větší roztažení, než bylo původně stanoveno. Pochopitelně, v takovém případě TEX o tom podá hlášení, ale to už je pozdě: špatný čin již byl vykonán.

Důsledky takovéhoto chování jsou pro úspěšně provedený design dost hrozné: TEXu *nikdy* nesmí být dána možnost positivní roztažitelnosti, je-li vyžadována jakákoli automatizovaná kontrola horní hranice, nad kterou přerostl bílý prostor; stažení je možné použít, ale TEX je v tomto ohledu výrazně asymetrický a vzhledem k tomu, že lze použít k vyplnění ne zcela zaplněných stránek `\vfil` a jeho přátele, přičemž se vyhýbáme pro vytváření bílého místa zanořeným konstrukcím typu ... plus n pt, neexistuje žádný ekvivalent použitelný k negativní roztažitelnosti stránek, vyhýbáme-li se srážejícím konstrukcím typu ... minus n (důvodem je skutečnost, že TEX nedovolí to, co nazývá „nekonečným lepivým smršťováním“ v nerestringovaném horizontálním či vertikálním módu). Je tedy obtížným problémem zabránit TEXu, aby plně těžil z dovolené flexibility, a nakonec jen pečlivé prohlédnutí protokolového souboru a ruční zásah tam, kde TEX příliš přesáhl svou úspornost, budou dostatečnou zárukou, aby se věci nevymkly kontrole.

Vzpomeňme si ale na okamžik na diskusi o grafické úpravě založené na řádkovém rejstříku, kterou jsme vedli v předchozí části přednášky: jistě budete souhlasit, že jednoduché obložení řádků s nadpisy příkazy `\vskip` nebude mít vždy žádoucí efekt. Mnohem uspokojivější metoda pro umístění nadpisů, s jistotou, že zabírají celočíselný počet prázdných řádků (např. celočíselný násobek `\baselineskip`), spočívá na technice, kterou nazvu „pseudoboxem“: jedná se o TEXovskou konstrukci, přičemž ve skutečnosti jde o box, chovající se jako glue; následující útržek programu tuto techniku ilustruje v praxi.

```
\newbox \headerbox
\newdimen \headerheight
\newdimen \headerdepth
\def \header #{\afterassignment\afterheader
                          \setbox\headerbox=\vtop}
\def \afterheader{\noindent \aftergroup\reallyafterheader}
\def \reallyafterheader
    {\headerheight=\ht\headerbox
     \headerdepth =\dp\headerbox
     \advance\headerheight by\headerdepth
             \headerdepth=\headerheight
     \ht \headerbox = 0 pt
     \dp \headerbox = 0 pt
     \advance\headerheight by 0.5\baselineskip
     \divide \headerheight by  \baselineskip
     \multiply\headerheight by  \baselineskip
     \ifdim \headerheight < \headerdepth
            \advance \headerheight by \baselineskip
     \fi
     \box \headerbox
     \vskip \headerheight
     \noindent
     \ignorespaces
}
```

Použijeme-li tento program k vysázení velkého polotučného nadpisu uprostřed textu v tomto odstavci, jako v příkazu `\nadpis {\Huge Nadpis}`, efektem by mělo být, že zbytek odstavce bude vytištěn ve svém normálním řádkovém rejstříku;

Nadpis

Rozhodnutí, jestli bylo popsaného efektu dosaženo, ponecháváme na čtenáři! Snad je na místě stručný popis programu, neboť — alespoň pokud si je autor vědom — tato technika nebyla nikdy dříve popsána. Makro `\nadpis` nevolá žádný parametr, ale křížek před otevřenou závorkou způsobuje, že před jeho použitím je požadována otevřená složená závorka; za předpokladu že tato otevřená složená závorka je závorkou vymezujícím parametr (což by měla být, je-li makro správně použito), pak makro vloží

`\headerbox` do `\vtop`, který obsahuje parametr. Nicméně je do `\vtop` přidán ještě další token, těsně předcházející tomuto parametru, a to pomocí `\afterassignment`, a tímto tokenem je `\afterheader`. Tento token sám expanduje do tří dalších tokenů: `\noindent` (aby parametr nebyl odsazen uvnitř boxu), `\aftergroup` (aby bylo dovoleno následujícímu příznaku expandovat nikoli uvnitř boxu, ale mimo něj, jakmile byl vysazen) a `\reallyafterheader`, což je makro, které vše vykoná. Tudíž kombinovaný efekt `\afterassignment` a `\aftergroup` má potlačit odsazení parametru a způsobí, že `\reallyafterheader` bude vykonán až po vysazení celého boxu. `\reallyafterheader` začne svou práci tím, že si uschová výšku a hloubku boxu, v němž má být nadpis vysazen, a pak je sečte; výška a hloubka je nastavena na 0 pt. Podle Knuthova algoritmu z A15.8 je sečtená výška + hloubka zaokrouhlena k nejbližšímu celočíselnému násobku `\baselineskip`, a pokud je výsledek tohoto zaokrouhlení menší než původní součet, přidá se další `\baselineskip`. Výsledkem tohoto výpočtu je nejmenší celočíselný násobek `\baselineskip`, v němž je možné obsah boxu vysadit. Pak je box vysazen (připomínám, že má nulovou výšku a hloubku, a tudíž nezabírá žádné místo) a následuje vertikální skok o vypočtený celočíselný násobek `\baselineskip`, aby se uvolnil prostor pro obsah boxu a neporušila se přitom pravidelnost základního řádkového rejstříku. Konečně `\noindent` a `\ignorespaces` zajistí, že první odstavec, následující po nadpisu, je vysazen správně.

Při skutečné sazbě s pomocí tohoto programu bude třeba parametrizace, která bude indikovat úroveň nadpisu, z níž by mohlo být odvozeno (nahlédnutím do tabulky), jak naložit s dalším prázdným prostorem kolem nadpisu; navíc by měl umožnit sazbu na praporec uvnitř boxu s nadpisem a korektní zacházení s nadpisem následujícím okamžitě po jiném nadpisu (prázdná místa by se neměla sčítat). Jsou možné i další zjemňující úpravy.

6 Odstavce

Pokoušíme-li se podat praktické rady pro praxi knižní úpravy, je nezbytné oddělit od sebe situace, ke kterým dochází zřídka (úvodní strany kapitol, nadpisy atd.), a ty, které tvoří podstatu knihy (normální strany, odstavce atd.). Zde se zamyslíme nad tím, co prolíná většinou stránek knihy, nad odstavcem.

Naštěstí jsou „pravidla“ pro odstavce velice přímočará a jasná, ale jelikož je k vidění tolik příkladů, které buď očividně tato pravidla porušují, nebo si je prostě neuvědomují, je i zde potřeba se o těchto pravidlech

alespoň v krátkosti zmínit. Je však třeba poznamenat, že tato pravidla jsou nedílnou součástí každé kultury; a byl jsem upozorněn jednou z význačných francouzských autorit,[4] že níže uvedená zásada týkající se prvního odstavce v každém oddíle by při aplikaci ve francouzském prostředí nebyla správná.

- První odstavec nového oddílu není odražen. Toto pravidlo je mnohem častěji porušováno než zachováváno, až se někdy ptám, jestli jeho existence vůbec vešla ve známost. Z důvodů mně zcela nejasných, LaTeX dělá vše pro to, aby odrazil, co se dá, což se mi zdá přinejlepším rozporuplné a přinejhorším neomluvitelné. Jsem velice potěšen, že v tomto materiálu tuto chybu nenalézám.
- Odstavec je buď odražen nebo odsazen bílým místem od předchozího materiálu. Za normálních okolností se považuje za nevhodné udělat obojí; hrubou chybou je neudělat ani to, ani ono. Důvodem, proč je to druhé tak hrozný zločin, je skutečnost, že pokud není odstavec ani odražený ani odsazený od předchozího textu, může se přihodit, že předchozí text skončí přesně na pravém okraji stránky a existenci odstavce, který následuje, můžeme jen stěží uhodnout. Nebudeme mít *žádný* typografický klíč, který by nám řekl, že začal nový odstavec.
- Rozpal řádků a písmo uvnitř odstavce jsou jednotné. To se může zdát samozřejmé, ale pokud je dokument vysázen s minimálním rozpalem řádků nezbytným pro obyčejný text bez ozdob, pak verzálka s akcentem může lehce posunout celý takový řádek dolů. V takovém případě je třeba buď zvětšit rozpal řádků v celém dokumentu nebo musíme podniknout speciální kroky, abychom skryli výšku písmena s akcentem (přičemž je třeba se ujistit, že nedochází k nešťastnému spojení se spodní dotažnicí písmene ve vyšším řádku). „Jednotou“ aplikovanou na písmo nemám na mysli, že každičká ozdůbka v textu musí být vytištěna stejným typem písma; samozřejmě může vzniknout potřeba kurzívy, někde případně i *bezpatkového* písma, jak jsme se zmínili výše. Ale všechny znaky v odstavci musí *vypadat* jednotně a musí tudíž pocházet z velice příbuzného nebo dobře vybraného písma. Například je možné první slovní spojení v každém odstavci knihy vysadit počáteční verzálkou a dále kapitálkami; za předpokladu, že se dobře spojí s ostatním textem, nebudou proti tomu žádné námitky. Podobně může být první písmeno odstavce zmenšená verzálka; opět za předpokladu,

[4] Jedná se o Bernarda Gaulla, bývalého a budoucího presidenta francouzské společnosti uživatelů TeXu GUTenberg.

že vplyne do dalšího textu, je to naprosto plnohodnotné řešení designu (a někdy i velice módní, mohu-li zde vyjádřit své osobní stanovisko).

- Odstavec by neměl končit v posledním řádku jen částí slova. Za předpokladu, že vůbec dovolujeme dělení slov (což bude třeba, má-li být text oboustranně zarovnán), pak by měl poslední řádek odstavce končit přinejmenším celým jedním slovem a lépe několika slovy. Nastavení `\parfillskip` v plain TEXu (i v LATEXu) to nezajistí; vhodnější nastavení by mělo být: `\parfillskip = 0 pt plus 0.7\hsize`, což podpoří delší poslední řádky za cenu drobného rozředění některých jiných řádků.

7 Obrázky, diagramy, tabulky a ostatní „plovoucí“ materiál

Třebaže je toho mnohem více, co lze (a by se mělo) říci obecně o knižní úpravě, mám pocit, že existuje jedna oblast, o níž se ještě na závěr zmínit musím. Jedná se o celou oblast vsuvek neboli (jak jsou označeny v LATEXu) „plovoucích objektů“ (floats). V jistém obecném smyslu je můžeme označit jako výtvarné celky, i když se může stát, že jsou obsahově sestaveny pouze s písmen a čísel. Co je však opravdu charakterizuje, je skutečnost, že je k nim bez výjimky nepřímo odkazováno; to znamená, že autor k nim odkazuje spíš výrazy jako *viz Obr. 1* nebo *viz Tabulku 2.4*, než implicitně pozicí v textu jako takovém, např. *jak je vidět níže*. Díky své podstatě nepřímého odkazování, mohou být tyto objekty posouvány z místa na místo, ovšem jejich pečlivé umístění je jedním z největších umění grafické úpravy. Základní pravidlo pro tyto objekty zní: *musí být vidět z místa, odkud se k nim odkazuje*. Jednou málo vychvalovanou předností TEXu je, jak dobře si umí poradit s tímto úkolem, pokud jde o poznámky pod čarou, což je jednoduchý případ takového objektu; prohlédnete-li si pečlivě materiál s velkým množstvím poznámek pod čarou vysázený v TEXu, budete pravděpodobně překvapeni, kolikrát se odkaz objevuje na posledním řádku stránky (tedy než se objeví samotná poznámka). Pokud jste o tom dříve nepřemýšleli, můžete třeba jen poznamenat: „To je ale štěstí; ještě jeden řádek a odkaz k poznámce a poznámka sama by se objevily na různých stranách“. Pokuste se ale teď najít místo, kde se toto přihodilo; hledejte, jak můžete: řeknu vám rovnou, že je nenajdete. Z toho samozřejmě plyne, že se jedná o něco víc než o prosté štěstí, co způsobuje to pravidelné objevování se odkazu k poznámce a poznámky samotné na stejné straně, které je tak pravidelné, spolehlivé a konzistentní. A ono to samozřejmě *je* něco víc než štěstí; po celou dobu, kdy TEX postupně načítá text, pečlivě sleduje,

kolik místa tvoří poznámky pod čarou a kolik hlavní text; a okamžitě, jakmile dojde k přelití textu přes rozměr stránky při kombinaci těchto dvou parametrů, TEX pozná, že musí sazbu strany dokončit někde poblíž tohoto bodu a začít na nové straně.

Poznámky pod čarou jsou však, jak jsem řekl, obzvlášť jednoduchý příklad pro takové vsouvání objektů; nikomu nevadí, jestli text poznámky začne sice na stránce, kde je k ní odkaz, ale pokračuje dál na další straně (nikomu kromě beznadějných pedantů, přesně řečeno). Ale diagramy, tabulky, obrázky atd., to je jiná káva. Jsou to ze své podstaty nedělitelné celky a mohou se tudíž na dané stránce buď objevit, nebo neobjevit; neexistují žádná pravidla, která by dovolovala část diagramu, tabulky či obrázku umístit dole na stránce a zbytek na další.

Teď si tedy představte sami sebe v pozici TEXu, tentokrát nenačítáte text a poznámky pod čarou, nýbrž text a (řekněme) diagramy. TEX pokračuje v přibírání materiálu do sazebnice jako předtím a najde odkaz k diagramu; řekněme že stránka je přitom pouze z třetiny plná. Je-li diagram maximálně ve velikosti dvou třetin strany, je po problému: TEX prostě přidá diagram na seznam věcí, které se na stránce mají objevit a pokračuje dál. Nechť je teď tady ale ještě druhý odkaz, dejme tomu ve dvou třetinách stránky: TEX se podívá, jak velký je diagram, a zjistí, že sám potřebuje polovinu strany. Co TEX udělá? První možnost můžeme pochopitelně vyloučit; nemůžete mít odkaz k diagramu, za nímž je jiný diagram, protože část diagramu by přesáhla spodní okraj stránky. Dobře, jakou další možnost máme? Vzpomeňte si, že diagram může *cestovat*. Pokusme se tedy přenést diagram nahoru na stránku, na níž k němu bylo odkazováno: není v tom žádný problém, diagram se objeví nahoře a text stlačí pod sebe. Část textu spadne pochopitelně pod spodní okraj strany, protože víme, že máme text velikosti dvou třetin strany a polovinu strany zabírá diagram, tudíž 1/6 stránky s textem spadne pod spodní okraj. To však není problém, neboť text lze rozdělit téměř v jakémkoli místě: TEX si tudíž vybere nejbližší přípustný bod k rozdělení a přenese zbývající materiál na další stranu.

Co se stane? Vzpomeňte si, co je *v materiálu* přeneseném na další stranu: odkaz k diagramu, který byl původcem popsaných potíží! Máme tedy nyní diagram na straně n a odkaz k němu na straně $n+1$. Pokud je $n+1 \equiv 1 \pmod 2$ (pardon, pokud je $n+1$ liché číslo!), pak nejde vlastně o žádný problém, neboť odkaz k diagramu se objeví na pravé straně dvojstránky a diagram sám vlevo, takže vše je v pořádku. Pokud je ale $n+1$ *sudé*, pak se celá konstrukce zřítí: neboť diagram je na pravé straně

dvojstránky a odkaz k němu na rubové straně dvojstránky následující; a když čtenář konečně dospěje k odkazu na diagram, diagram sám už není vidět. A bez ohledu na to, co dokázal TEX v této situaci udělat, nebude umět problém vyřešit bez pomoci.

V grafickém návrhu strany jsou tedy skryty problémy, které jednoduše *nelze* vyřešit naivní aplikací pravidel; všechna pravidla jsou dobrá, ale vždycky přijde chvíle, kdy se autorův text stane neslučitelný s pravidly knižního designu, a v takové situaci vám nezbude než se spojit s autorem a pokusit se přesvědčit ho, aby problematickou část textu přepsal. Je-li autor mrtvý a text je vytesán do kamenných desek, budete muset spoustu práce vykonat manuálně, třeba i přesázet celou posloupnost odstavců vždy o jeden řádek delších, než by pro ně bylo ideální, jen aby se vám podařilo dostat odkaz na vhodnější stránku. Ale podstoupíte-li to, a hotová kniha je vytištěna, podíváte se na ni a *víte*, že už neexistuje vylepšení, které byste v ní ještě mohli udělat, zalije vám útroby hřejivé teplo a budete v té chvíli vědět, že to stálo za to. Hodně štěstí!

The Computer Centre,
Royal Holloway and Bedford New College,
University of London,
Egham Hill, Egham, Surrey TW20 OEX,
United Kingdom.
E-mail: `Philip Taylor (RHBNC) <P.Taylor@Vax.Rhbnc.Ac.Uk>`

Z angličtiny přeložil Ladislav Šenkyřík.

Computer Typesetting or Electronic Publishing? New trends in scientific publication

Philip Taylor

Abstract. Whilst computer (assisted) typesetting has completely replaced more traditional technologies such as hot lead during the last fifteen years, it too is now coming under threat from an even more radical technology in the form of electronic publishing (e.p.). Unlike both traditional and computer typesetting, e.p. avoids the use of paper completely and replaces it with a computer display. The potential advantages of e.p. are obvious: massive savings in cost, virtually instantaneous transmission; but there are disadvantages too, such as the difficulty of avoiding unauthorised copying, plagiarism, and re-distribution. There are also fundamental philosophical differences: whereas typesetting systems dictate the *exact* appearance of the finished page, e.p. systems may allow a far more abstract representation, with the final appearance being left very much up to the viewer. In this paper, I will outline recent developments in both computer typesetting and electronic publishing, and compare and contrast the two approaches.

A Brief History of TEX

Until about fifteen years ago, typesetting was virtually ignored by the vast majority of mathematicians, scientists, and scholars in general: manuscripts were prepared using a typewriter, the more esoteric symbols (which meant almost all symbols for mathematicians) were laboriously inserted by hand, and the whole was then simply dispatched to the publisher. Some time later galleys would be returned, emendations noted in the margin, and once again the whole would be sent to the publisher. A similar but shorter cycle was probably repeated for the page proofs, and finally the author's intentions appeared in final form in the finished book. At no point did the author and the typesetter communicate directly, and indeed the former was almost certainly virtually unaware of the latter's existence.

The typesetter, however, was only too aware of the author: mathematical copy is traditionally referred to as 'penalty copy' in the printing trade, since it is notoriously difficult to set correctly. In the time that his colleague could set ten pages of straight text, the mathematical typesetter was barely able to accomplish a single page, and even when set he knew that there was every possibility that it would have to be re-set more than once, since mathematicians are only too keen to invent new symbols of their own when no existing symbol seems entirely appropriate. And since the typesetter would never have encountered such a symbol before, he would (quite reasonably) assume that it was simply a badly drawn version of a symbol with which he *was* familiar, and substitute the latter...

Needless to say, some of the more aware authors began experimenting with computer technology as soon as it became generally accessible, and for a while the academic world seemed convinced that if it were possible to get just a couple more symbols onto the daisy-wheel of a Diablo printer, all would become possible: there were even specialist companies who would re-mould a daisy-wheel, replacing an apparently unwanted glyph with one which its owner deemed indispensable. Of course, the approach was doomed to failure: one can no more set mathematics with a fixed set of 144 glyphs than can one with a set of 128, and despite the best efforts of all concerned, the daisy-wheel printer was soon consigned to the scrap bin.

In parallel with this, the dot-matrix printer manufacturers first began to have a significant impact. With a 7×5 dot matrix, there are potentially $\sum_{i=0}^{35} \binom{35}{i} (= 2^{35})$ different characters (a very large number indeed!), but unfortunately a number of these are virtually indistinguishable: a single dot at co-ordinates $(4, 3)$ looks astonishingly like another single dot at at co-ordinates $(4, 4)$ to even the most astute reader (I believe that there are 33 034 338 305 *distinct* characters, as opposed to a total of 34 359 738 368 characters, where a character is regarded as *distinct* if it's not simply the result of sliding another character horizontally, vertically, or both: this figure is based on an analysis by Dr Warren Dicks of the Autonomous University of Barcelona). Furthermore, the print quality of a 7×5 dot matrix printer is so appallingly bad that no attempt should *ever* be made to set a book using one – unfortunately this well-meant advice was seldom heeded at the time.

Of course, in order to exploit these technological revolutions, suitable software had to be written, and the Unix world in particular decided

to standardise on ROFF and its derivatives: NROFF, TROFF and finally DITROFF all made their mark. Unfortunately none of the ROFF derivatives ever directly supported the typesetting of mathematics, and so adjunct programs such as EQN and TBL had to be used to add mathematical functionality. There were also commercial systems, used to set publications such as the *Transactions of the American Mathematical Society*, but these were both expensive and arcane, using a rather non-mnemonic syntax to represent the possible mathematical constructions.

Fortunately (as is absolutely clear in retrospect), at least one eminent mathematician believed that something better not only could, but *should*, be created; and being not only a mathematician but a computer scientist, he decided to create it. His name was Knuth, and his creation was TEX.

Yet had it not been for a happy co-incidence, TEX might never have been born. At the time, Knuth was working on his *opus magnum*, a seven-book series entitled *The Art of Computer Programming*, and by 1977 the popularity of the early volumes of this series had proved so great that Volume 2 had already run to a second edition. Unfortunately the timing of this was such that whilst the first edition had been set using traditional hot-lead technology, the second edition was produced using one of the first phototypesetters [an aside to readers: throughout this paper I use the term *typesetter* to mean both the *person* performing the task of setting type, and the *equipment* used to achieve that end: I hope that it is always clear from the context which of these two meanings is to be inferred, since there is no other word which could easily and felicitously be substituted for either of these usages]. And whilst the new phototypesetter was more than capable *in theory* of achieving results as good as, if not better than, the traditional hot lead device used previously, the results in practice left a great deal to be desired. Knuth, as mathematician and computer scientist, was convinced that the fault lay not in the technology but in the software used to drive it, and he decided that rather than see his life's work appear in second-rate format, he would devote a short portion of his professional life to developing a suite of software which *would* exploit the full potential of the phototypesetter. Little did he know when he took this brave decision that it was to take not the anticipated one year but at least ten, although he most certainly had a demonstrably working version within his anticipated time-frame.

The first published reference to TEX is probably *Mathematical Typography*, published as report *STAN-CS-78-648* by the Computer Sci-

ence Department of Stanford University; in the bibliography to this, Knuth gives the definitive reference as being *Tau Epsilon Chi, a system for technical text* which was at the time "in preparation" and is now sadly out of print. For those interested in the subject, the former paper makes fascinating reading, and the bibliography alone makes it a more than worthwhile acquisition; it was reproduced in the *Bulletin of the American Mathematical Society*, in which form it should still be available.

TEX was both typical and atypical of programs of its era: it was typical in that it was completely script-oriented, pre-dating as it did any widely-used graphical user interface; it was atypical in that it was a completely programmable *macro* programming language, in which there were no reserved words, and in which even individual characters could change their semantics on the fly. Thus a TEX document consisted both of the text to be typeset and the commands to accomplish that typesetting, and only TEX itself could unambiguously determine whether any particular element of the document was to be interpreted as 'program' or 'data'.

Despite being created primarily in order to accomplish one particular end – the typesetting of Volume 2 of *The Art of Computer Programming* – TEX rapidly took on a life of its own, and soon became the *de facto* standard for typesetting within much of Stanford University. Before long its fame had spread, and by 1980 the TEX Users Group had sprung into existence, with members of the Steering Committee drawn from far beyond the restricted domain of Stanford faculty. The American Mathematical Society were represented on that Committee, and liaison between the AMS and Knuth was very close: Knuth assigned the TEX logo to the AMS who then applied for trademark protection to prevent it being used to describe any unauthorised modification of TEX – unfortunately this application was rejected because of a prior registration of TEX (sic) by Honeywell, but despite this lack of formal registration, Knuth's high profile and high standing ensure that the TEX logo (or its non-typeset equivalent, `TeX`) is universally recognised and respected.

Within a couple of years, it became clear that the initial implementation of TEX left something to be desired, both in terms of functionality and in terms of portability, and Knuth set out to redress both by re-implementing TEX from scratch. This time he decided to eschew SAIL ('Stanford Artificial Intelligence Language') as the language of implementation, and instead to adopt the far more widely available programming language Pascal. To further increase its portability, he adopted

only a strict subset of Pascal, encompassing only those features which he was confident could be found (or easily emulated) on all Pascal implementations; but he also decided to take this opportunity to render the program in a form which he termed 'literate': that is, he wanted people to be able to *read* the source of TEX in the same way that they might read a book, and to therefore be able to benefit by being exposed to a major piece of software engineering presented in a highly literate manner. Once again Knuth decided that there were no adequate tools available for this, and once again he digressed from the main project by breaking off to design and implement the concept of a WEB program, together with its two adjunct programs TANGLE and WEAVE.

A WEB program consists of a highly stylised dialect of Pascal, interspersed by lengthy comments describing the purpose and function of every element and module of the program (I suspect that Knuth would deny this, and say that a WEB program consists of a highly elaborate description of the workings of the program, interspersed by occasional fragments of Pascal which implement that functionality: and I suspect that he would almost certainly be right!). By permitting the elements of a Pascal program to be presented in arbitrary order (as opposed to the strict order of presentation required by the Pascal standard), WEB allows the programmer the opportunity to present the elements of a program in a natural and logical order, as opposed to the artificial order imposed by the Pascal design criterion of 'efficient compilability': it is then the task of TANGLE to paste together these fragments in the order required by Pascal, and the task of WEAVE to bring together both the program fragments and the comment fragments into a form which can immediately be typeset by TEX.

Thus for the first time TEX became self-referential: in order to be able to produce the Pascal code from the WEB source, one needed a working version of TANGLE; to be able to produce a literate listing of the WEB source, one needed a working copy of WEAVE; but both TANGLE and WEAVE are themselves written in WEB, so to produce a working TANGLE one needs a working TANGLE, and so *ad infinitum*. Of course 'bootstrapping' (as the technique is generally termed) is well understood in the Computer Science world, and it was estimated that the task of 'hand compiling' TANGLE from the WEB source was well within the competence of 'the average implementor': however, I remember only too clearly the trauma through which a colleague went when he attempted this bootstrapping for himself...

During the re-implementation, Knuth re-wrote almost the complete TEX program: he had learned much about its limitations during the first couple of years of use, and by 1982 a completely re-written TEX had emerged. This version of TEX (often referred to as TEX 82, to differentiate it from the earlier version which analogously became known as TEX 78) was rapidly ported to a wide range of machines, and is quite possibly the most widely available program in the world today, being available on every class of system from the smallest PC to the largest super-computer. Its almost universal acceptance as *the* standard package for computer typesetting is almost certainly the result of a large set of very positive attributes: the source of the program, and the vast majority of implementations, are available either free of charge or at a modest cost which covers no more than the media on which they are supplied; the program is virtually bug-free, a claim which Knuth backed up until very recently by offering a cheque for every bug found, the value of the cheque doubling each year since the scheme's inception (he still offers a cheque, but the value no longer doubles, since he estimated that before too long it might exceed the total Federal reserves...); the program is highly stable (there were virtually no major changes during the period 1982–90, and similarly there have been virtually no changes at all since 1990, nor will there be at any point in the future); and there are an enormous number of users throughout the world, most of whom are only too keen to pass on their expertise to any who need it, so any real problems resulting from a lack of experience with TEX can be rapidly resolved by a message to any one of a number of TEX-related mailing lists and news groups (even those without network access are not cut off, as the TUG (TEX Users Group) office offers telephone support from 03:00 in the morning until late in the evening – a service which is *not* restricted to members of TUG).

So, during the 1980's, TEX emerged as *the* standard package for computer typesetting: it was available on almost every conceivable system, device drivers were written for everything from dot-matrix printers to 2400 dpi phototypesetters (but *not* daisy wheel printers!), and an ever-increasing number of publications appeared which were either typeset using TEX, or were about TEX, or both. Many scientific journals adopted it (or one of its derivatives such as LATEX, which may be thought of as a somewhat restrictive but more user-friendly 'front end' to TEX) as the standard format in which papers were to be prepared. Since an author could very easily proof a paper using a local implementation of TEX,

and since TEX was *guaranteed* to produce identical results no matter on which system it was run, the number of iterations between author and publisher was reduced to the bare minimum, and all benefitted. And since TEX has been designed by a mathematician, and since a part of its raison d'être had been to allow mathematics to be typeset almost as easily as running text, its take-up by the mathematical community was if anything even faster than its take-up by the scientific and academic communities in general.

To give a simple example of why TEX is ideally suited to the typesetting of mathematics, consider the following set of equations:

$$\begin{aligned}\left(\int_{-\infty}^{\infty} e^{-x^2}\,dx\right)^2 &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-(x^2+y^2)}\,dx\,dy\\ &= \int_0^{2\pi}\int_0^{\infty} e^{-r^2} r\,dr\,d\theta\\ &= \int_0^{2\pi}\left(-\frac{e^{-r^2}}{2}\Bigg|_{r=0}^{r=\infty}\right)d\theta\\ &= \pi. \end{aligned} \tag{11}$$

A mathematician writing this by hand would almost certainly start with the left-most element of the first line, proceed from left to right, and alternate between baseline, subscript and superscript elements as logic dictated; a pure WYSIWYG ('What you see is what you get') word processor, on the other hand, would require the typist to analyse each row of the equations into horizontal strata (thus the top stratum might contain only ∞, 2, ∞ and ∞, for example) and to enter these stratum by stratum; since, in general, WYSIWYG systems do not automatically displace preceding or following lines of text horizontally when an intervening line is shortened or lengthened, the correction of such equations is tedious and error-prone in the extreme. More recent, WYSIWYG-like, systems require a different approach in which the author has to enter the formula in the order dictated by its parse-tree; needless to say, this approach too demands more of the author than should reasonably be expected.

TEX allows the mathematician to enter the formulæ in the most natural manner, starting at the left and finishing at the right; alignment is automatically maintained if insertions or deletions are made, and even the horizontal alignment of the four primary = signs is performed automatically, virtually regardless of the length of individual left- or right

elements. To clarify this, here is the exact TEX source which was used to set the table:

```
$$
\vbox \bgroup
      \eqalignno
      {\biggl (\int_{-\infty}^\infty e^{-x^2}\,dx \biggr)^2
        &=\int_{-\infty}^\infty
               \int_{-\infty}^\infty e^{-(x^2+y^2)}\,dx\,dy \cr
        &=\int_0^{2\pi}\int_0^\infty e^{-r^2}r\,dr\,d\theta \cr
        &=\int_0^{2\pi}\biggl (-{e^{-r^2}\over 2}
             \bigg \vert_{r=0}^{r=\infty}\,\biggr)\,d\theta \cr
        &=\pi.&(11)\cr
      }
      \egroup
$$
```

It is worth noting that TEX completely ignores any spaces in mathematical text, since the rules for typesetting mathematics are complex, and cannot be expected to be understood by mere mathematicians! Thus the layout of the equations above is simply for the convenience of the author, and is completely ignored by TEX, which is far more concerned by special characters such as dollars, backslashes, braces, underscores, carets and ampersands. And whilst each of these characters has a distinct meaning to TEX (a dollar symbol, for example, both introduces and terminates a stretch of mathematical text), that meaning may at any time be overridden, and either assigned to a different character or, if not needed, turned off completely. So, for example, if some particular computer lacked a backslash key, it would be trivial to assign the semantics of backslash to some other key (say, yen, if a Japanese keyboard were to be used).

Furthermore, it can be seen that TEX is highly mnemonic in its choice of control sequences ('commands', preceded by a backslash); to pick out just a few examples, `\int` represents an integral sign, `\infty` an infinity, `\exp` the exp operator (representing the exponential e) and so on. Compound subscripts and superscripts are presented in logical order, rather than in order of their appearance vertically on the page; and facilities are provided for the author to give TEX hints about the logical structure of the expression, so that (for example), `\,` is used to set off differentials such as $d\theta$ from the preceding text by a little extra white

space, thereby improving both the appearance and the legibility of the expression.

Thus the attraction of TEX for mathematicians is clear: a highly logical markup language, capable of being entered from any keyboard; access to a very wide range of mathematical symbols; professional standards of layout; widespread acceptability by journals; and the ability to proof on anything from a dot-matrix printer to a 600 dpi laser printer. Add to this the now universal ability to preview the document on the computer screen (something the early advocates of TEX could only dream of), and it is hard to explain why any mathematician with access to a computer would *not* typeset his papers using TEX!

However, use of TEX is restricted neither to mathematicians nor to North Americans, and at the TEX User Group conference in 1989, an influential and voluble group of European TEX users ganged up on Knuth and succeeded in convincing him that, despite his assertion on the previous day of the conference that the development of TEX was finished, there were features missing from the current implementation which made TEX entirely useless to the majority of the world, since whilst it behaved perfectly in unaccented languages, it was grossly deficient for typesetting any language which made more than occasional use of diacritics. And Knuth, recognising the validity of this argument, agreed that something had to be done.

The result of all this was TEX 3: TEX 82 became known simply as TEX 2, and TEX 3 became the One True TEX. In practice, this just didn't happen: those who had no need for the extended diacritic support offered by TEX 3 simply continued to use TEX 2, and for quite a while TEX macro writers had to write very defensive code which first checked the environment before making any assumptions about (for example) the number of distinct characters with which TEX could internally deal (this was 128 prior to TEX 3, and 256 thereafter). With the release of TEX 3, Knuth made it *absolutely* clear that this really did represent the end of the TEX evolutionary line: he had better things to do with his time, and TEX was now frozen (modulo any essential bug fixes, which he undertook to continue to make if and only if it could be shewn that their fixing was essential). Furthermore he made it equally plain that TEX could *not* be further evolved by anyone else: he wished to leave for his children, and for his children's children, and for all perpetuity, TEX as his creation, and not as his-creation-as-modified-by-someone-else.

In general, the TEX world took this in good part: Knuth is enormously highly respected by those who use TEX, and there were very few who advocated ignoring his wishes and who were prepared to suggest modifying TEX. But there were also a quite significant number of TEX users, among them the present author, who felt that if TEX did *not* evolve, then it would simply die. Not because of any fundamental deficiencies in TEX – it is generally accepted that there are very few – but because the world had moved on since 1978, and whilst a script-driven language might have been state-of-the-art then, it most certainly was not state-of-the-art now. Furthermore, despite increasing the number of distinct internal characters from 128 to 256, Knuth had done little if anything to enhance TEX to deal with Asian languages, in which the number of distinct characters may be measured in thousands if not in tens of thousands. And finally, there were those who felt that there were *some* areas in which a very significant increase in functionality could be gained (particularly from the perspective of the macro programmer, who is also known as a 'format writer' when the suite of macros provides a complete functional system in its own right) with relatively little investment in terms of modifying TEX.

The implementation of these ideas probably represents the leading edge of TEX technology ('TEXnology') today: companies such as Blue Sky have produced instantaneous/incremental TEX interpreters, which are capable of displaying the effects of a change to the source code of a TEX document in real time; Advent Publishing have produced 3B2, which allows both a graphical and a textual specification of a layout, automatically updating one to reflect changes in the other; John Plaice and Yannis Haralambous have implemented a 64-bit version of TEX which uses Unicode internally; and the group with which I am most closely associated (the NTS group, where NTS stands for 'New Typesetting System') have produced a completely compatible successor to TEX, called e-TEX, which adds functionality without compromising compatibility (the NTS group also wish to re-implement TEX from scratch, using a modern rapid-prototyping language such as Prolog or CLOS, the idea being to allow rapid experimentation with alternative typesetting algorithms or paradigms). Whether or not any of these ideas will catch on remains to be seen, although among Apple Macintosh *aficionados* Classic Textures (the Blue Sky product referred to above) is already highly thought of. One fundamental question is that of stability: since one of the great strengths of TEX is its stability, how will the world feel about systems

which encompass TEX but which are specifically intended to remain evolutionary and responsive, rather than fossilised and unyielding? Only time will tell.

What is perhaps worth noting is that all of these projects have ensured that Knuth's wishes are honoured not only in the letter but in the spirit: none seeks to call itself TEX (indeed, that of John Plaice and Yannis Haralambous is called Omega, which could never be confused with TEX), yet all acknowledge the debt which they owe to Knuth and to TEX: without them, none of these other projects would ever have seen the light of day.

Parallel Developments

Of course, while TEX was evolving, the rest of the world did not stand still: computer science continued to develop, and computer networking moved from the laboratory to the military and the Universities and ultimately to the whole world. Line-oriented editors fell by the wayside, and were replaced by full-screen editors (except in the rather time-warped world of MS/DOS, which continued to offer only EDLIN until comparatively recently). Script-oriented markup languages such as the ROFF family referred to earlier were challenged by increasingly sophisticated word-processors, and WYSIWYG ('What You See is What You Get'), GUI ('Graphical User Interface'), and WIMP ('Windows, Icons, Menus and Pull-down lists') became the order of the day.

At about the same time that Knuth was starting work on TEX, John Warnock and Martin Newell re-implemented an earlier language ('the Design System') as 'JaM' ('John and Martin'!) whilst working at Xerox PARC, and from this cloistered beginning ultimately emerged both the Interpress (Xerox printing protocol) and PostScript languages. Whilst Interpress remained relatively unfamiliar, Adobe PostScript took the computing world by storm: for the first time there was a *de facto* page description language, which allowed complex pages to be described algorithmically (and thus very efficiently). Although Hewlett Packard's Printer Control Language (PCL) continued (and continues) to be both widely supported and widely emulated, PostScript rapidly established itself as *the* standard for high-level printers (by which I mean laser printers and better), and fairly quickly printer manufacturers sought to provide either PostScript interpreters or PostScript emulators for their high-end products. Unfortunately (for the emulator writers)

PostScript is a complex language, and many of the earlier emulations were deficient in one or more respects; Adobe, of course, as designers of the language had far fewer problems in this respect, although even they released improved versions of their interpreter as time went by.

For a long while parts of PostScript remained a closely guarded secret: the mysterious `eexec` operator was undocumented, and whilst the PostScript manual gave information on the format of so-called 'type 3' fonts, the equally mysterious 'type 1' fonts remained undocumented. Of course, reverse engineering is a well-understood tool, and finally the barriers were broken: descriptions of `eexec` started to appear in the press, and ultimately Adobe themselves relented and gave full documentation of both `eexec` and their type-1 fonts.

Before long, type-1 fonts established themselves as as much a standard for fonts as PostScript was a standard for page-description languages; companies such as Corel started to release type-1 fonts of their own, closely modelled on industry-standard fonts but sufficiently different (at least in name) to avoid accusations of font piracy (although this latter problem continues to worry top font designers such as Hermann Zapf to this day). All the major font foundries started to offer their fonts in type-1 format, and many gave a commitment to have *all* of their fonts in type-1 format within the foreseeable future. The so-called 'font magic' which enabled early Adobe fonts to render well even on relatively low resolution devices such as 300 dpi laser printers was renamed 'font hinting', and this too was eventually documented by Adobe. New features continued to be added to the PostScript language, and in 1990, Adobe announced a completely new version of the PostScript language, 'PostScript Level 2'. This new version unified all previous additions to the language, and added many new features as well, such as the ability to have compact (binary) representations of a PostScript document as well as the earlier (ASCII) representation; new colour models were introduced, and support was added for composite fonts.

PostScript was originally conceived as an embedded language for printers, but before long it became clear that a version of PostScript which could drive a computer screen would be extremely useful. Adobe created their own version of this called 'Display PostScript', but in the meantime L. Peter Deutsch had started work on a PostScript interpreter of his own, called 'Ghostscript', and fundamental to its functionality was the ability to drive the screen of any computer on which it was used (it also contained drivers for a wide-range of non-PostScript

printers, as well as pseudo-drivers for some of the more popular graphics interchange formats). During 1995 Peter finally announced Ghostscript version 3, which provided almost a complete Level 2 emulation, and whilst the official Adobe interpreter remained a licensed (and relatively expensive) product, Ghostscript was and remains free of charge to those who do not use it for profit-making purposes; a very significant debt of gratitude is owed by the computer world to L. Peter Deutsch, both to his skill in writing Ghostscript and to his generosity in making it so freely available, and also to the many individuals who have donated their own drivers and/or enhancements to the Ghostscript project (PS-View, from Bogusław Jackowski and Piotr Pianowski warrants special mention).

From the ARPAnet to the Web

A few years before Knuth started work on TEX, the American military as personified by [D]ARPA (the [Defence] Advanced Research Projects Agency), had initiated a pilot project to link computers over very wide distances; whilst local computer links were not uncommon, links across thousands of miles were unheard of, but [D]ARPA realised the potential military importance of such links and therefore initiated a whole series of research projects aimed at making this a reality. Whilst these projects initially started in isolation, as soon as the pilot network was available the project gained a momentum – indeed, a very existence – of its own, and the whole development strategy henceforth was established by discussion *across*, as well as *about*, the network. This network, known as the ARPAnet for obvious reasons, evolved a mechanism for distributed discussion and voting known as the 'Request for Comments' ('RFC'), and any new idea for anything from a protocol to a picnic was likely to find itself the subject of an RFC. From these RFCs emerged some of the most important *de facto* standards on which we still to this day: TCP ('Transmission Control Protocol'), IP ('Internet Protocol'), SMTP ('Simple Mail Transfer Protocol') and so on were all enshrined in the published versions of the RFCs, and each was allocated a unique number: electronic mail, for example, was addressed by and specified in, RFC 822.

Although the American military launched the networking initiative, it was the American universities which were actually the primary contributors to its success, and once the network was well established it ceased to be 'the ARPAnet' and became instead 'the Internet', the name by which

it is still known today. Strictly speaking, the Internet is not a network *per se*, but a network-of-networks; however, the distinction is of little significance, and most now regard the Internet simply as *the* international computer network. From its military origins, where permission-to-connect almost required a personal interview with a five-star general, the Internet has now become the network to which even the most humble private citizen may aspire to gain access: Internet service providers have sprung up across much of the Western world, and connecting to the Internet today requires little more formality than a letter (and a fairly modest cheque!) to an Internet service provider, together with the purchase of a equally modest personal computer and a modem: at the time of writing, there are Internet connections from something like 150 countries throughout the world (the number of actual Internet nodes is *far* harder to gauge, but it is already estimated to lie between five and ten *million*).

Initially the protocols used, and services provided, on the Internet were very primitive: FTP ('File Transfer Protocol'), TELNET (remote terminal access), PING (check if a remote node is alive), and SMTP were probably the most common, with FINGER (check if a remote user is logged in) coming not far behind. But whilst the end-user protocols were fairly simple, the underlying mechanisms were not, and the DNS ('Domain Name Service') provided a quite sophisticated mechanism for a distributed node lookup protocol. As more experience was gained, the range of protocols and services increased, and things such as Usenet News (a distributed bulletin board) and NFS ('Network File System', providing remote access to a complete file system) were added. Then the information explosion really took off, and tools for information retrieval and display began to proliferate: GOPHER and WAIS ('Go for', and 'Wide Area Information Service, respectively) were early candidates, shortly followed by WWW (the 'World Wide Web', now usually shortened to 'the Web'). It should be emphasised that there is *no* connection between WEB programs and the World Wide Web; within this document at least, the former is consistently shewn in upper case, whilst the latter is consistently shewn in mixed case.

With the advent of the Web came one major breakthrough: whereas previously each protocol had specified its own unique method of identifying a remote resource, WWW brought with it the concept of the URL (the 'Universal Resource Locator'), so that from within a single program (the 'browser'), almost *any* Internet resource could be specified.

For example, a remote FTP resource would commence `ftp://`, a remote GOPHER resource would be `gopher://`, and the Web's native resource, HTTP ('HyperText Transfer Protocol') would commence `http://` (aware readers may appreciate that this is a slight over-simplification, but the deviations from reality are essentially very small).

With the Web and URLs came unified browsers: tools such as MOSAIC which allowed access to a wide range of Internet resources from a single graphical front end. Even if a resource had no unique URL, it was still possible to associate with it an adjunct renderer which would display it correctly: thus, for example, although there is no unique URL for an MPEG file ('Motion Picture Expert Group': a compact standard for encoding and storing full-motion video), a correctly configured browser such as MOSAIC could identify an MPEG resource from its file type (the portion of the file name which follows the period), and on down-loading such a resource would then spawn off an instantiation of the appropriate renderer, so that down-loading and viewing were essentially indivisible entities.

Native-mode documents for access over the Web are coded in a language called HTML ('HyperText Markup Language'): this is a direct derivative of an earlier (but still current) specification for a generalised markup language called SGML ('Standard Generalised Markup Language'), and a conformant HTML document is also normally a conformant SGML document, although as is often the case the converse does not necessarily obtain. Both HTML and (typical but not all) SGML documents are characterised by the frequent occurrence of tags which are enclosed in angle-brackets: they therefore resemble the 'metalinguistic variables' of a much earlier standard – the BNF (or Backus Naur/Normal Form) of the original Algol-60 report – although they do not perform the same function. In an HTML document, each tag specifies the *nature* of the entity to which it refers: whilst this can be augmented by a specification of how the entity should appear, in the purest form only the nature of the entity is specified, and it is left to the browser to determine how the entity should appear. This represents a very significant philosophical breakthrough: no longer need a document be formatted by its author, the reader then requiring the technology to resolve that format; instead, using HTML, a document is simply tagged using high-level content-oriented markup, and the reader may then display that document using whatever technology is available. For example, most Internet systems are capable of running a browser called LYNX: this is a purely

textual browser, and so it makes no attempt to represent subtleties of the document; it simply takes advantage of whatever text-mode functionality is available to it (for example, emboldening or underlining) to display the document to the best of its ability. Images which would normally require a graphics mode browser to resolve are simply displayed as the word `[IMAGE]`. On more sophisticated systems, graphics mode browsers such as MOSAIC (previously referred to), or the now ubiquitous NETSCAPE, can be used: these will exploit the graphics capability of their platforms to the full, and are capable of displaying full-colour, and even motion-picture-insertions, either using inherent functionality or through the medium of adjunct software which has been used to customise the browser.

But an HTML document is far more than just a passive entity: elements of it can be designated as 'hot spots', and if a hot spot is selected (using the mouse on a graphical system, or the tab and/or cursor keys on a line-mode system), a further document may be downloaded and displayed entirely automatically: the document containing the hot spot and the document referred to by the hot spot do not need to originate from, or be stored at, the same site: a document stored at (say) the University of Western Ontario can reference, through a hot spot, another document stored at (say) the University of Queensland. Furthermore, although the discussion so far has been concerned with 'documents', hot spots can in fact be linked to *any* Internet resource, provided only that the resource is specifiable via a URL. Thus a document which was fetched using HTTP can reference another document that can be fetched using only GOPHER; that document could specify a third document which is accessible only via FTP: that could contain a reference to a Usenet Newsgroup; and so on.

Yet even this does not represent the limits of a Web document: such documents can also be *forms*, with fields which must be completed by the reader; when the form is completed, a further hot spot can transfer it to a remote site, where it will be interpreted and acted upon. In this way, the original ARCHIE protocol (ARCHIE is an Internet tool for locating files available via anonymous FTP) has been extended from its traditional usage in which it is launched from a command line invocation specifying the file of interest and some constraints on the manner of search; with the HTML version (a.k.a. 'Archiplex'), the Archie user invokes his preferred Web browser to fetch an Archie form from a convenient server; he then completes the form, and uses a hot spot to return it to the Archie server;

the latter then locates the file of interest, and returns a list of places at which it can be found, where the list of places is possibly constrained by options selected on the form by the user (for example, he may say that he's only interested in copies of the file that can be found within his own domain). The list of hits is then displayed by the browser, and once again using the mouse, tab key and/or cursor keys, the user selects one instance of the file of interest; the file has associated with it a hot spot, so the instant he selects the file from the list, a request is issued to retrieve the file; assuming that there are no hiccups, the file is fetched entirely automatically and displayed on the originating screen. If the file is not displayable for some reason (perhaps it is an executable image, or something else for which the concept of 'display' is ill-defined), the browser will inform the user and ask if he wishes to save it to a local disk.

Whilst previous introductions such as GOPHER and WAIS had a relatively modest impact on overall use of the Internet, which in general continued to be used mainly by academics and hackers, the introduction of HTML and the concept of the Web brought about a revolution in Internet usage: commercial companies clamoured to get on-line, governments put up their own Web pages, and every man and his dog suddenly appeared to be beset by the need to create a unique and highly individualistic 'home page' (Web documents are often regarded as being divided into pages, by analogy with a paper document, and a 'home page' is a (usually brief) document giving information about the individual who owns it; many institutions provide facilities whereby each user can create his or her own home page without formal approval). The reason for this sudden change in usage patterns is not hard to explain: whereas the more traditional Internet tools such as FTP required a modicum of expertise before they could be successfully used, the various Web browsers were intentionally designed to be 'user friendly' from the very outset, and this user friendliness together with the ability to seamlessly download and display documents in an astonishing variety of formats without any expertise whatsoever resulted in an unprecedented rate of take-up and an almost universal acceptance. There can be little doubt that the current near-exponential rise in Internet registrations and usage results almost entirely from the concept and ease of use of the Web.

The Web and Publishing: Unlikely Bedfellows?

Whilst it might initially seem that the two themes of this paper represent quite distinct branches of the evolutionary tree, it did not take long for those involved in publishing to realise the untapped potential of the Web: even prior to the establishment of the Web there had been some experimental use of the Internet for electronic publishing. In particular, the so-called e-journal *EJournal* (subtitled Electronic Journal for Humanists) was a direct electronic analogue of a more traditional journal, containing scholarly essays as well as shorter "letters to the editor". However, *EJournal* uses simply ASCII text as the communications medium, whilst the Web potentially allows even greater richness of medium than any traditionally produced journal, since unlike a paper journal a Web journal could contain not only text and static graphics but full motion video and sound as well.

In July 1994 the American Mathematical Society launched a project entitled "New Media", chaired by Frank Quinn of Virginia Tech., to investigate the possibility of developing a multimedium, interactive, hypertextual version of TEX: the brief of the sub-committee established to investigate this was to "co[-]ordinate the development of a technical authoring tool which will integrate text, graphics video, non[-]linear documents, hypertext links, and interactive computation. [The] tool should share the characteristics of the TEX typesetting system which have made it so remarkably useful: open file structures, open and portable source code, a stable standard core, and an uncompromising commitment to the highest quality. [It] is expected to be an extension of TEX.".

The rationale behind this proposal is also interesting: "Educational communities need interactive texts. Technical communities need hypertext and non[-]linear document types to tie together complex or cumulative efforts. Users of computation need better ways to document and illustrate programs. All these capabilities are available now in primitive forms, and authors are pushing ahead. Some are writing interactive texts using computer mathematics programs. Others are experimenting with hypertext extensions of TEX, [WWW] documents, etc. Commercial publishers are experimenting with hypertext, CD ROM publication, and linked databases. In a few years we can expect powerful tools for constructing interactive multimedia documents. But they may be 'accessible' in the same sense that typesetting was accessible before TEX: publishers will use expensive proprietary systems with closed file formats,

and authors will use a multitude of free or inexpensive systems which require professional resetting to get professional results. Our experience with TEX shows that this fragmentation is undesirable and unnecessary. The HyperMath Project is being organized to avoid it.".

The introductory document for the project then went on to explain: "The HyperMath Project is primarily a framework to co[-]ordinate work already in progress. Several groups have already incorporated simple hypertext links into versions of TEX. The NTS (New Typesetting System) group is exploring improvements to traditional paper-and-ink typesetting. Most implementations of TEX have methods for incorporating graphic material, and there are publicly available packages which do this. The 'Interactive Mathematics Text' project and many groups in the calculus reform movement are using Mathematica, Maple, MatLab, and other programs to write interactive texts. These and similar initiatives can be brought together in the development of a general tool. But the opportunity is limited. As development proceeds, the costs of switching to a common standard increase, and the benefits become less obvious. We should not let this opportunity pass. The Project will sponsor working groups and conferences. The working groups will develop standards and goals, and work on prototypes. Communications among working groups will be maintained to ensure coherence and uniformity. And contacts will be developed between developers and end-users to ensure that real needs are being addressed. Working groups are planned in the following areas: traditional text; non[-]linear documents (including hypertext); inclusions (graphics, video, and sound); interactivity; and users. The first HyperMath conference is planned for the San Francisco area in conjunction with the combined math society meeting early in January, 1995, contingent on funding. The 'New Media' subcommittee of the Publications Policy Committee of the American Mathematical Society will serve as the advisory board for the HyperMath Project."

This was heady stuff: sadly by September of the same year it had been abandoned as being "too ambitious", and replaced by a more incremental approach, now entitled "non-traditional forms of publication". Whereas the earlier project had been predicated on the development and adoption of enhanced TEX, the new project proposed that "the AMS should adopt the Adobe portable document format 2.0 as the standard (output) format for electronic publication of documents". It then went on to explain that "This does not mean giving up TEX, nor does it solve all

TEX problems. It is a proposed replacement for DVI as output, not the use of TEX source in authoring." What did this mean?

The first thing to realise is that by now all three threads of this paper have finally come together: TEX, Adobe, and the Web. Whilst Adobe had been very successful in developing PostScript as a page-description language, and marketing embedded PostScript interpreters for incorporation in laser printers and the like, it had been somewhat less successful in ensuring that Display PostScript became established as another *de facto* standard. Indeed, with the advent of Ghostscript, a significant future for Display PostScript was by no means certain, and the proliferation of Web-based browsers (MOSAIC, NETSCAPE and the like) which could slave Ghostscript was a further challenge to Adobe's position in the marketplace. Unlike Adobe's PostScript interpreters and Display PostScript systems, Web browsers were (and remain) freely available: that is, they are literally available *free of charge*, even when they are as sophisticated as NETSCAPE (which is developed and supplied by a commercial organisation). Whilst Adobe could maintain its niche as a supplier of PostScript interpreters, it was becoming clear this was a limited, and possibly even diminishing, market: if Web-based publication rather than paper-based publication ever became the norm, the rôle of PostScript printers and image setters might be seriously challenged as more and more documents were read from a computer screen rather than from paper. It was therefore no great surprise when Adobe finally announced (there had been clues previously, such as their work on so-called 'multiple master fonts' and Carousel) their alternative to a Web browser as a universal document rendering engine: Adobe Acrobat. Just as with the Web browsers, Adobe Acrobat is available free of charge (indeed, they send complete CD ROMs containing a full multi-lingual installation kit at the slightest provocation). And, rather like NETSCAPE, who seek to recover the costs of developing their browser by selling their Web server, Adobe will endeavour to recover the cost of the development and production of their Acrobat reader by selling the technology which is required to produce an Acrobat document in the first place.

And what is an Acrobat document? The very same thing that the AMS are investigating as a possible standard for their mathematical publications: something written in Adobe Portable Document Format (PDF). And although in theory one can develop applications of one's own which will write PDF, in practice many will elect simply to purchase Adobe Ac-

robat (which acts as a pseudo-printer driver for MicroSoft Windows, Apple Macintosh or Unix systems), or Adobe Acrobat Pro[fessional] (which also includes Adobe's "Distiller" to convert PostScript documents into PDF documents), or Adobe Acrobat Capture (which uses the TWAIN protocol for scanners to generate PDF documents directly from a scanner). Thus despite their apparent generosity in giving away Acrobat free of charge, Adobe are (of course) really seeking to increase their market share by encouraging the purchase of other Adobe products.

HTML **or** PDF**?**

With HTML and PDF emerging as *the* two portable hypertext exchange standards, organisations (and, to a lesser extent, individuals) are going to be forced to make a choice. It may well be that for some applications the choice will be clear-cut, but for others there may seem little to choose between the two. It is therefore worth exploring the basic differences between HTML and PDF, in order to better allow an informed choice to be made.

HTML, being SGML, is essentially a very high level, content-oriented, markup language: its *forte* is the specification of the content of a document, and its weakness is the relatively little control that an HTML author has over the final appearance of the document. Because it is so high level, it is not possible using the current received wisdom of computer science to automatically generate HTML from an arbitrary document: if a word-processor, for example, is used to prepare a document, and if that document has been created *ex nihilo* without consideration for its logical structure, so that only the final appearance of the document has been considered, then it is almost certainly impossible to reverse-engineer the document to ascertain its logical structure: in these circumstances HTML would have little option but to represent it as an indivisible bit-map, thereby effectively wasting almost all of HTML's functionality. Despite this restriction, HTML has much to offer, for two main reasons: (1) the tools needed to generate it are already in the public domain, although the interface between those tools and pre-existing software such as word-processors is unlikely to be available (it is far more likely that word-processor packages will start to be shipped with HTML drivers, but their use may require a major re-think by the user concerning the the way in which a document is created); and (2) high-level markup is increasingly recognised as being *the* way in which to mark

up a document: as experience of the use of typesetting systems such as TEX/LATEX is gained, it becomes ever more clear that low-level, form-oriented, markup is simply a dead-end and should rapidly be expunged from the practices of responsible authors.

PDF, on the other hand, consists essentially of a strict subset of POSTSCRIPT with the added functionality of hypertext: PDF documents can reference other PDF documents using hot spots, rather like HTML. According to the PDF blurb (this paper is written before my copy of Adobe Acrobat Pro has arrived, so what follows must be taken as speculative at the moment):

- ▷ Create electronic documents as easily as printing from existing applications with PDF Writer.
- ▷ Protect files with passwords; control access, printing, changing the document, adding and changing notes, copying text and graphics.
- ▷ Find exactly what is needed across multiple PDF files by searching on keywords, author, title, subject synonyms, etc.
- ▷ Re-use information easily by extracting, copying, reordering and replacing pages among PDF files – with bookmarks, links and notes preserved.
- ▷ Create custom views into information.
- ▷ Add value, set priorities and maintain a dynamic information network with links, bookmarks, notes and connections to external applications and documents.
- ▷ Take advantage of third-party plug-ins to add new features to Acrobat.
- ▷ Integrate Acrobat with desktop applications with Acrobat's support for OLE automation, Notes F/X, AppleEvents, and more.

Although perhaps it is too soon to compare HTML and PDF with any real accuracy, it would seem that at the moment they are intended for, and best suited for, rather different applications: HTML documents can either be created *ex nihilo* (for those who have no better way, simply cloning and modifying an existing HTML document is an excellent way to get started), or by using an HTML editor (of which there are already several in the public domain), or by using a package or packages (for example, a suitable word-processor) for which an HTML driver already exists. PDF documents may be created using one of the Adobe tools – Acrobat Writer, Distiller or Capture – depending on whether or not the source documents pre-exist. As HTML allows only a degree of control in the formatting and placement of entities, it is not really suitable for the presentation of anything other than simple mathematics, although

HTML 3 demonstrates that the designers of HTML are aware of many of the limitations of the previous version, and are working towards a specification which may ultimately allow arbitrarily complex formulæ to be displayed. [A comment in the HTML 3 discussion document reads "Including support for equations and formulæ in HTML 3.0 adds relatively little complexity to a browser. The proposed format is strongly influenced by TEX."]. Of course, since HTML allows reference to be made to non-HTML documents, many of these difficulties can be overcome: an HTML browser such as NETSCAPE can be configured to invoke an external renderer if no internal renderer is suitable for the entity referenced, and in that way both DVI (from TEX) and POSTSCRIPT documents can be referenced from, and displayed from within, an HTML document. Since both DVI and POSTSCRIPT are equally suited to the accurate representation of mathematical material, there is no real reason why a mathematical document should not be displayed from within an HTML framework by an HTML browser configured with a suitable external renderer. PDF, on the other hand, has no need for external renderers, since its native mode of operation uses a strict subset of POSTSCRIPT; indeed, Adobe Acrobat is intended to be configurable *as* an external renderer for HTML browsers such as NETSCAPE! By using Adobe's 'multiple master' font technology, Acrobat can generate a reasonable substitute for any font specified in a PDF document, even if that font is not available within the system on which the document is being displayed. It is by no means unlikely that before very long a DVI-to-PDF driver will emerge, and in the true tradition of TEX it is also extremely likely that such a driver will be placed in the public domain; DVI-to-HTML is an unlikely eventuality, however, since by the time a TEX document has been converted into DVI, too much information has been lost to allow the high-level structure of the document to be re-created. On the other hand, we can certainly envision a format being created for TEX which embeds `\special`s in the DVI file to convey information about the high-level structure of the source document: since the user interface would be completely unaffected by the presence of these specials, such a format could appear to the user exactly like any of the present formats or format variants which support appropriate high-level markup ($\mathcal{AMS}$-TEX, LATEX, $\mathcal{AMS}$-LATEX, LA$\mathcal{MS}$-TEX, etc.). Such specials could then be directly mapped into HTML constructs, and thus a TEX-to-HTML route is neither impossible nor unlikely; indeed, it is surprising that no such extended format has yet been announced (at least, to my knowledge).

Finally it is worth remembering that HTML is essentially a *distributed* markup language; it is primarily intended for documents which need to reference other documents which may be anywhere on the Internet; PDF, on the other hand, is essentially Internet-unaware, and whilst it can transparently reference other documents that are visible through (say) NFS (or, using ALEX, anonymous FTP), it makes no assumptions that documents might be anywhere other than the local filestore or on a Microsoft-compatible network. [This last sentence is somewhat tentative: in the absence of the definitive PDF specification, it is somewhat difficult to accurately interpret the claim that Adobe Acrobat allows one to "Add value, set priorities and maintain a dynamic information network with links, bookmarks, notes and connections to external applications and documents', but I suspect that the 'dynamic information network' does not allow the transparent referencing of arbitrary Internet resources, although this may well come in time.]

Computer Typesetting or Electronic Publishing: Pros and Cons

Computer typeset material, particularly that typeset using TEX or a functionally equivalent system, represents the finest in typeset quality that can be easily accomplished today; where such computer typesetting software is unavailable, comparable results can only be accomplished by a skilled professional using either old-fashioned technology (e.g. hot lead) or a modern but proprietary system. This is not to say that the use of a system such as TEX *guarantees* professional quality results: there are far too many counter-examples in existence which demonstrate that in completely unskilled hands, TEX and comparable systems are capable of generating absolutely appalling results. None the less, in reasonably skilled hands, and/or using a format package which prevents the author from making design decisions, TEX is capable of generating results which meet the highest professional standards, particularly in the field of mathematics where TEX essentially performs as an 'expert system'. The disadvantage of such a system is that in its intermediate form (DVI), a TEX document is not fully portable: a DVI file contains references to, but no instances of, fonts; at the point where the DVI file is converted into its final form (usually paper, but on-screen preview is now also ubiquitous), the same fonts which were used to create the document must be available in order to render it correctly; in their absence, only a poor approximation of the intended document is possible. [It is worth noting

that the creator and the viewer/printer of a TEX document need a common set of fonts, but each needs a quite different representation of those fonts: the creator needs only the *font metrics*, which specify the height, depth and width of each glyph, and kerning and ligaturing information for the glyph set; the viewer/printer of the document can normally get by without the metrics, but instead needs the actual glyph set, either as bitmaps or as outlines.]

Electronic publishing, on the other hand, and particularly e.p. accomplished through the medium of HTML, does not place any emphasis on the quality of the end product: indeed, HTML voluntarily cedes control over the appearance of the final document to the browser used to render it, although there are some placement options which allow the author a little control over the final appearance (and there are considerably more such options in HTML 3). Within an HTML document there is no font information *per se* (again, this is true only of current HTML: HTML 3 adds the concept of style sheets, which will "[...] eventually lead to smart layout under the author's control, with rich magazine style layouts for full screen viewing, switching to simpler layouts when the window is shrunk"); instead the document consists of a set of high-level markup tags, which are mapped by the browser to a particular font or font variant. Whereas a DVI file is a monolithic entity, and makes no reference to any external resources other than fonts, an HTML file is frequently little more than a container for other HTML files, and may make reference to an extremely wide range of resources (further HTML files, images, AFS files, Usenet newsgroups, e-mail addresses, FTP-accessible files, etc.) which may be anywhere on the network, and which may themselves contain further references and so *ad infinitum*.

PDF is essentially a reasonable compromise between the two: the creator of the document specifies its appearance, and the PDF reader then displays that document to the best of its ability: if the fonts needed to display it properly are embedded, or if they are resident on the target system, then the document will be displayed exactly as the author intended; if the fonts are not accessible, then Adobe's proprietary 'multiple master' technology will be used to interpolate a substitute for the missing font(s) which allows the original line-breaks, leading, etc., to be retained. A PDF document may reference further PDF documents, but these are assumed to be available on the local filestore; there is no apparent support for the automated fetching of remote Internet documents, although the absence of the PDF documentation at the time of writing

makes analysis of this feature rather more of an informed guess that a definitive statement.

All three formats discussed allow searching to be conducted; within a DVI file there is no intrinsic support for indexing, but it would not be at all difficult for a DVI viewer to create a dynamic index to the document being viewed. Both HTML and PDF allow fully indexed documents to be referenced.

Publication in the Twenty-First Century

It is no longer possible to assume, as countless previous generations of scientists have done, that "publication" involves printing on sheets of paper which are ultimately distributed as a part of a journal or as a book: increasingly both economic and environmental pressures will dictate that only essential information be committed to paper, and anything even slightly ephemeral will be restricted to electronic distribution. At least two *de facto* standards have already emerged for electronic publication: HTML, which originates in the distributed and anarchic world of the Internet; and PDF, which originates in the commercial world. At the time of writing, HTML is the better established, and two freeware browsers are widely used (MOSAIC and NETSCAPE), with a third (ARENA), being developed specifically to support HTML 3; for PDF, there is only one reader currently available (Acrobat), and that too is classified as freeware. HTML devolves to the browser most of the decisions concerning the actual appearance of a document; PDF allows the author to make most of those decisions, but reserves the right to substitute interpolated fonts if the genuine article are not available at the point of rendering. HTML is essentially a distributed protocol, and will allow bibliographies to reference cited texts no matter where they are in the world (so long as they are on-line), thereby adding truly incalculable value to the bibliography of a document; PDF, it would appear, is essentially a local protocol; whilst bibliographies could still cite full-text sources, those sources would need to be available to the system on which the bibliography is being read.

Many issues remain to be resolved before the world can truly move to electronic publishing as the mainstream form: Internet access in every home, office, library, vehicle, and restaurant will be just a start. There remains the very contentious issue of copyright: whilst there are usually economic costs associated with the photo-copying of a printed

document, the costs of copying an electronic document are virtually nil, and therefore the enforcement of copyright for electronic publications is a major concern. It is highly likely that some form of encryption and licensing will emerge to prevent the unauthorised copying and/or re-distribution of electronic texts. From the psychological and physiological point of view, displays will need to become significantly better (in many senses: weight, resolution, glare, portability, etc.) before the electronic book completely replaces the printed equivalent: few of us, going on holiday today, would choose to take a notebook computer with a CD ROM containing the complete works of Shakespeare in preference to a couple of (disposable) paperbacks... Although originally developed as front ends for the generation of printed material, typesetting systems such as TEX will almost certainly have a major rôle to play as front ends for electronic publishing, since (for example) the linear representation of mathematical formulæ is equally convenient and applicable whether one's mathematics are eventually to appear on paper or on a computer screen. Within ten years, HTML and PDF will appear passé: new standards emerge faster than most of us can keep up, and today's technology is tomorrow's door-prop. But the future of the book (or even the newspaper) as the normal means of communication is surely as doomed as that of the petrol-driven car as the normal means of conveyance; the first to guess exactly what form the replacement will take may become as rich as today's newspaper magnates and publishing house principals; or perhaps the converse will occur, and the Internet will finally cause the collapse of the publishing empires, as academics and authors suddenly realise that they are no longer beholden to the few. Self-publishing may become the norm, or peer review may take on an entirely different form; perhaps a two-tier hierarchy of electronic publishing will emerge, with unrefereed papers being available via each academic's home page whilst those that have survived the refereeing process will be available from prestigious and highly accredited archives. What is certain is that almost all of the readers of this paper will find out for themselves what the future holds, at least as regards computer typesetting and electronic publishing: the future is just around the corner, and approaching at an ever increasing speed.

Within the last forty-eight hours, I have learned of two new facts which significantly impinge on the material above: NETSCAPE have licensed the use ofPDF technology from Adobe, which will allow them to incorporate a PDF renderer within their HTML browser, and Michel

Goossens & Sebastian Rahtz have demonstrated the feasibility of using Adobe's 'multiple master' fonts from with TEX; further details of the latter, including very useful information on multiple master fonts, are given in the *Baskerville* issue cited in the Bibliography.

I would like to thank Professor Adam Jakubowski and Jerzy Ludwichowski for making it possible for me to present this paper, to Elżbieta Kuczyńska and Bogumiła Rykaczewska-Wiorogórska (University of Warszawa) for kindly providing two alternative translations of the abstract into Polish, and to Professors Adam Jakubowski and Andrzej Jonscher for providing reverse translations into English to enable me to check the accuracy of the initial translation. I would like to thank Dr Warren Dicks of the Autonomous University of Barcelona for his analysis of the problem of the number of visually distinguishable configurations of an $m \times n$ matrix, as used to establish the number of distinct characters which can be generated by a 7×5 dot matrix printer. I would like to thank Dr Frank Quinn of Virginia Tech. and the American Mathematical Society for granting me permission to reproduce extracts from their documents on "The New Media" and "Non-traditional forms of publication", and finally I would like to thank Barbara Beeton of that same Society for allowing herself to be persuaded to review the paper before publication and for her many helpful comments; needless to say, any errors which remain are solely my responsibility.

Philip Taylor, RHBNC, University of London; July 28th, 1995

I have not given formal references in the text, since I feel that they are inappropriate in a paper of this nature; however, the following short list of publications may be of interest to those who wish to pursue further the topics discussed here.

"Mathematical Typography" by Donald E. Knuth, *Bulletin of the American Mathematical Society* (new series) 1 (March 1979), 337–372. [Reprinted as part 1 of *TEX and METAFONT: New Directions in Typesetting* (Providence, R.I: American Mathematical Society, and Bedford, Mass: Digital Press, 1979).]

"Tau Epsilon Chi, a system for technical text" by Donald E. Knuth, Stanford Computer Science Report 675 (Stanford, California, September 1978), 198 pp. [Reprinted as part 2 of *TEX and METAFONT*, the book cited above.]

"The WEB system of structured documentation" by Donald E. Knuth, Stanford Computer Science Report 980 (Stanford, California, September 1983), 206 pp.

"Literate programming" by Donald E. Knuth, *The Computer Journal* 27 (1984), 97–111.

"Using Adobe Type 1 Multiple Master fonts with TEX" by Michel Goossens and Sebastian Rahtz, *Baskerville* Vol. 5, No. 3 (UK TEX Users' Group, June 1995, ISSN 1354-5930), 4–8.

HTTP: A Protocol for Networked Information
`http://www.w3.org/WWW/Protocols/HTTP/HTTP2.html`

A Quick Review of HTML 3.0
`http://www.w3.org/hypertext/WWW/Arena/tour/start.html`

HyperText Markup Language Specification Version 3.0
`http://www.hpl.hp.co.uk/people/dsr/html3/CoverPage/html`

Document Type Definition for the HyperText Markup Language (HTML DTD)
`http://www.w3.org/hypertext/WWW/MarkUp/html3-dtd.txt`

Adobe Acrobat `http://www.adobe.com/Acrobat/Acrobat0.html`

Dvě recenze

Miroslav Dont

G. Grätzer: Math into LaTeX;
A Simple Introduction to LaTeX and $\mathcal{A}_{\mathcal{M}}\mathcal{S}$-LaTeX

Druhé vydání, Birkhäuser, Boston 1995 (asi listopad), asi 350 stran, ISBN 0-8176-3805-9, cena asi $42,50.

Některé údaje o této knize jsou poněkud neurčité. Je to dáno tím, že v době, kdy píši tuto recenzi (srpen 1995), kniha ještě nevyšla. Ke knize jsem se totiž dostal tak, že jsem měl možnost pomoci autorovi s korekturami (jako asi dalších 30 dobrovolníků). Měl jsem tedy v ruce preprint knihy a recenze se vlastně týká tohoto preprintu. Autor píše, že konečná

podoba knihy se v mnohém liší od preprintu, neboť na posledních úpravách strávil mnoho času. Předpokládám však, že alespoň obsahově bude stejná a že v tomto smyslu recenze (ostatně ne příliš podrobná) bude odpovídat skutečnosti.

Jedná se o další knihu, jejíž druhé vydání bylo umožněno díky vzniku LaTeXu 2_ε; u této knihy ještě navíc díky nové verzi $\mathcal{AMS}$-LaTeXu ($\mathcal{AMS}$-LaTeX verze 1.2). Název tohoto druhého vydání se nepatrně liší od názvu vydání prvního — *Math into TEX, A Simple Introduction to* $\mathcal{AMS}$-LaTeX. Recenzi prvního vydání je možné nalézt ve Zpravodaji č. 4 z r. 1993 (Jiří Veselý: *Recenze s přívažkem*, str. 184–188). Z názvu prvního vydání by se mohlo zdát, že se jedná pouze o popis maker $\mathcal{AMS}$-LaTeXu. Název druhého vydání je přiléhavější, neboť je z něho zřejmé, že se jedná o popis LaTeXu (v podtitulu se ještě říká, že se jedná o LaTeX 2_ε) *a zároveň* $\mathcal{AMS}$-LaTeXu. Zde to chce asi jisté vysvětlení. Připomínám, že $\mathcal{AMS}$-LaTeX vznikl původně jako rozšíření LaTeXu 2.09, a to rozšíření ve dvou základních směrech. Pokud se týče fontů, $\mathcal{AMS}$-LaTeX používal NFSS. Z tohoto důvodu $\mathcal{AMS}$-LaTeX potřeboval samostatný formátový soubor; proto se tedy při použití „standardní“ LaTeX a $\mathcal{AMS}$-LaTeX striktně odlišovaly (toto byla asi jedna z psychologických bariér, kvůli kterým mnoho lidí váhalo $\mathcal{AMS}$-LaTeX použít). Druhé základní rozšíření $\mathcal{AMS}$-LaTeXu oproti LaTeXu 2.09 jsou makra pro psaní matematiky převzatá z $\mathcal{AMS}$-TeXu (tato makra jsou ovšem upravená pro LaTeX — především mají LaTeXovou syntaxi). Dá se tedy v jistém smyslu mluvit o spojení LaTeXu a $\mathcal{AMS}$-TeXu. J. Veselý ve výše zmíněné recenzi mluví trochu záhadně o „poněkud exotickém kříženci“ — podle mne se nejedná o nic exotického, ale naopak o spojení velice přirozené. Dalším rozšířením je ještě definice tzv. $\mathcal{AMS}$-fontů, tj. například Eulerovy fraktury ($\mathfrak{a}$, $\mathfrak{A}$, $\mathfrak{b}$, $\mathfrak{B}$, $\mathfrak{c}$, $\mathfrak{C}$, ...), Eulerova scriptu ($\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, ...), azbuky, `\Bbb` (blackboard bold — $\mathbb{A}$, $\mathbb{B}$, $\mathbb{C}$, ...) a značného množství matematických symbolů (a definice příkazů pro tisk těchto symbolů). Dále jsou zde např. příkazy pro snadný tisk tučných matematických symbolů, pro „umisťování čehokoli nad (nebo pod) cokoli“ a mnoho dalších drobných rozšíření.

Jelikož v LaTeXu verze 2e je již NFSS zabudováno jako standardní součást, není nyní třeba pro $\mathcal{AMS}$-LaTeX mít samostatný formát. Příslušná rozšiřující makra jsou obsažena ve `.sty` souborech, z nichž základní je `amsmath.sty` (kromě `.sty` souborů obsahuje balík $\mathcal{AMS}$-LaTeXu také některé `.cls` soubory; nebudu se zde pouštět do vysvětlení rozdílu mezi těmito dvěma typy souborů). Padá tak jedna ze zábran, pro kterou někteří uživatelé LaTeXu nepoužívali $\mathcal{AMS}$-LaTeX. Začátek zdrojového souboru

českého článku pro LaTeX 2ε s použitím maker AMS-LaTeXu ve standardním LaTeXovém stylu article (a např. v 11-ti bodovém písmu) může vypadat např. takto:

```
\documentclass[11pt,draft]{article}
\usepackage{amsmath}
\usepackage{czech}
 ...
\begin{document}
```

Bez příkazu `\usepackage{amsmath}` by se jednalo o „standardní“ LaTeX (LaTeX 2ε); nyní je zřejmější, že AMS-LaTeX je opravdu pouhá nadstavba LaTeXu. Současná verze AMS-LaTeXu, která spolupracuje s LaTeXem 2ε, je označena jako 1.2; předchozí verze, která vyžadovala samostatný formát na základě LaTeXu 2.09, byla 1.1. Mezi těmito dvěma verzemi existují některé drobné rozdíly.

Součástí balíku AMS-LaTeX je také podrobný manuál. Tento manuál však předpokládá znalost LaTeXu (příslušné verze). Předložená kniha G. Grätzera je zároveň úvod do LaTeXu i AMS-LaTeXu. Mluví o LaTeXu 2ε jednoduše jako o LaTeXu, což odpovídá politice tvůrců nové verze LaTeXu, podle které LaTeX 2ε má být standard. Tím se liší např. od knihy H. Kopky a P. W. Dalyho, kde se mluví zároveň i o LaTeXu 2.09. Makra AMS-LaTeXu jsou chápána jako přirozená součást LaTeXu (přitom ovšem, pokud je řeč o příkazech definovaných v souborech AMS-LaTeXu, po straně textu je vždy označen soubor, který je třeba volat). Účelem knihy není podat co nejúplnější popis LaTeXu resp. AMS-LaTeXu, ale naučit čtenáře pomocí těchto prostředků psát matematické texty. Je to kniha psaná matematikem převážně pro matematiky (ale nejenom pro ně). Autor přitom dobře ví, že většina matematiků se nechce stát odborníky na TeX či LaTeX, ale prostě potřebuje nějakým způsobem svůj text napsat — nejpřirozenějším prostředkem je k tomu v současné době pravděpodobně právě AMS-LaTeX.

První část knihy s názvem *A short course* obsahuje pouze jednu kapitolu *Typing your first article*. Již podle názvu je zřejmé, že tato kapitola je určena úplným začátečníkům v TeXu (LaTeXu). Čtenář se tu postupně dozvídá, co všechno musí obsahovat každý LaTeXovský soubor, co jsou to soubory `.tex` a `.dvi`, jak se píše a dělí text (dělením zde myslím dělení na sekce atd.) a především základní věci o psaní matematických vzorců. Již v tomto okamžiku autor ukazuje, jak i zdánlivě složité formule

je možné postupně sestavit z jednodušších částí (odstavce *Some building blocks of a formula* a *Building a formula step-by-step*). Hlavní část knihy je druhá, která se jmenuje *A leisurely course*. Bez ohledu na to, co bylo řečeno v předchozí části, začíná opět od začátku, tentokrát však podrobněji a úplněji. Obsahuje 5 kapitol s názvy *Typing text*, *Displayed text*, *Typing math*, *Multiline math displays* a *The LaTeX document*.

Kniha má ještě další tři části:

Document classes. Popis standardních `.cls` souborů a to jak „standardního“ LaTeXu, tak `.cls` souborů dodávaných v rámci $\mathcal{AMS}$-LaTeXu.

Customizing. Příkazy a okolí definované uživatelem v LaTeXu a obecněji makra v TeXu.

Long bibliographies and indices. Použití BibTeXu a MakeIndexu.

Kniha je zakončena dodatky, které obsahují především tabulky symbolů a dále např. několik slov o PostScriptových fontech a něco o tom, jak lze získat TeX (a LaTeX a $\mathcal{AMS}$-LaTeX) na síti. Zajímavý je dodatek F, který obsahuje užitečné rady, jak konvertovat starší text do $\mathcal{AMS}$-LaTeXu verze 1.2. Jedná se o úpravy z TeXu (tj. z Plainu), LaTeXu, $\mathcal{AMS}$-TeXu a $\mathcal{AMS}$-LaTeXu verze 1.1.

Kniha obsahuje velké množství příkladů — skoro by se dalo říci, že je celá postavena na příkladech. Všechny tyto příklady lze získat pomocí FTP v elektronické podobě (v dodatku E se říká kde). Lze ji doporučit jako skvělý úvod do LaTeXu především matematikům (nejenom jim). Přitom text jde, zvláště pokud se týče psaní matematických formulí, i relativně značně do hloubky, takže užitek z četby může mít i poměrně zkušený uživatel. Jsem přesvědčen, že na základě (pouze) této knihy může každý matematik být schopen dát dohromady slušnou sazbu svých textů.

H. Kopka, P. W. Daly: A Guide to LaTeX 2_ε, Document Preparation for Beginners and Advanced Users

Druhé vydání, Addison-Wesley 1995, 554 stran, ISBN 0-201-42777-X, cena $42,50.

V roce 1994 byl dán do oběhu dlouho očekávaný LaTeX 2_ε. Jeden z hlavních účelů nové realizace LaTeXu bylo sjednotit a standardizovat různé verze a rozšíření LaTeXu, které v té době byly v používání. Například vedle sebe existovaly „standardní LaTeX“ — LaTeX 2.09, LaTeX s NFSS (NFSS — New Font Selection Scheme) a $\mathcal{AMS}$-LaTeX, který ob-

sahoval NFSS a dále většinu maker z $\mathcal{AMS}$-TEXu pro psaní matematiky (v LATEXovské syntaxi). Každá z těchto verzí měla svůj vlastní formát. Kromě toho existoval např. také SliTEX pro tvorbu slidů — opět jako samostatný formát. LATEX 2_ε obsahuje NFSS (nyní ve vyšší verzi) jako standard. $\mathcal{AMS}$-LATEX a SliTEX nejsou už samostatné formáty, ale pouze stylové soubory (nyní nazývané „package“, které se volají pomocí příkazu `\usepackage` — do detailů se zde nebudu pouštět). Měl by tedy existovat pouze jeden základní formát a všechny nadstavby by se měly realizovat pomocí „packageů“. Z hlediska uživatele se LATEX 2_ε od „standardního LATEXu“ liší především ve způsobu volání fontů, ostatní příkazy by měly fungovat stejně jako v LATEXu 2.09 (to, že některé příkazy mají jistá drobná, ale velice užitečná rozšíření, nemusí uživatel zvyklý na LATEX 2.09 zpočátku ani zaregistrovat).

Existence nové realizace LATEXu poskytuje autorům uživatelských příruček LATEXu příležitost k novým přepracovaným vydáním. Především tu je druhé vydání knihy L. Lamporta *LATEX—A Document Preparation System*, popisující LATEX 2_ε. Druhé vydání knihy H. Kopky a P. W. Dalyho je ve skutečnosti již druhé přepracování úspěšné německé knihy H. Kopky o LATEXu — první vydání v angličtině nebylo totiž pouhým překladem, ale obsahovalo navíc informace o mezinárodním LATEXu a dále některé aplikace vycházející ze zkušeností P. W. Dalyho. Druhé vydání je tedy přepracování pro LATEX 2_ε. Zajímavým rysem tohoto přepracování je, že nepopisuje pouze LATEX 2_ε, ale souběžně i LATEX 2.09; čtenář je tak upozorněn na všechny rozdíly mezi těmito dvěma verzemi LATEXu (zjistí se, že jich není zase tak mnoho). Pro začátečníka, který nezná LATEX 2.09 a chce začít používat rovnou LATEX 2_ε, to může být ovšem do jisté míry matoucí; pro úplného začátečníka bych ale v každém případě doporučil Lamportův základní manuál.

Kniha je rozdělena do devíti kapitol a šesti dodatků. Prvních sedm kapitol je základní popis LATEXu a v podstatě odpovídá Lamportovu manuálu. Jsou zde však navíc některé příklady vycházející ze zkušeností autorů a dále úlohy určené čtenářům. Nejméně jsem byl spokojen s kapitolou 5 o psaní matematiky. Můžeme si ovšem všimnout, že ani Lamport se této otázce příliš nevěnuje. Možná je to dáno tím, že LATEX vlastně nemá příliš mnoho vlastních prostředků pro psaní matematických vzorců (přesněji řečeno, vlastní prostředky LATEXu pro psaní matematiky nejsou úplné; vzpomínám si, že jeden z prvních problémů, když jsem začínal s LATEXem (a zároveň s TEXem) byl, že v celém Lamportovi jsem nenašel, jak vytisknout binomický koeficient). Lze ovšem

použít většinu prostředků z Plainu, které LaTeX přebírá, které však zde (ani v Lamportovi — viz výše uvedený příklad binomického koeficientu) nejsou popsány (obě knihy pouze konstatují, že většina příkazů Plainu funguje i v LaTeXu a explicite jsou vyjmenovány ty příkazy Plainu, které v LaTeXu použít *nelze*; toto je jeden z důvodů, proč i uživatel LaTeXu by si měl přečíst TeXbook). Pro psaní matematiky v LaTeXu je třeba v každém případě doporučit balík maker $\mathcal{A}\mathcal{M}\mathcal{S}$-LaTeX (soubor `amsmath.sty` a další).

Kapitola 6 popisuje okolí `picture` pro tvorbu jednoduchých obrázků v LaTeXu. Je obšírnější než původní Lamportův popis a obsahuje mnoho zajímavých příkladů (osobně se přiznávám, že již delší dobu používám jiné prostředky pro tvorbu obrázků než je okolí `picture`). Kapitola 8 se jmenuje *Advanced Features* a podrobněji popisuje např. NFSS a velice stručně např. standardní stylové soubory. Není mi příliš jasné, proč je popis tvorby křížových referencí až v kapitole s tímto názvem — možnost křížových referencí je (podle mého názoru) důležitá vlastnost LaTeXu. Devátá kapitola se velice podrobně věnuje popisu chybových hlášení, a to jak LaTeXu, tak samotného TeXu.

Oddíl dodatků je poměrně rozsáhlý. Je zde např. popis `letter.cls` pro psaní dopisů, užití BibTeXu, základní pokyny pro tvůrce `.cls` a `.sty` souborů, instalace LaTeXu, použití PostScriptových fontů v LaTeXu atd. Samostatný dodatek je věnován popisu Computer Modern fontů. Zajímavý je však poslední dodatek, který obsahuje abecední seznam všech příkazů LaTeXu se stručným popisem a odkazem na stranu knihy, kde je popis podrobnější (rozumí se nikoli vnitřních příkazů LaTeXu). Takto důkladně zpracovaný seznam příkazů LaTeXu jsem nikde jinde neviděl.

Nakonec bych si dovolil jednu poznámku k typografii knihy. Stalo se jistým zvykem, že knihy psané v TeXu již nepoužívají CM fonty, ale některé fonty PostScriptové. Je možná diskutabilní, zda by manuál k TeXu (nebo LaTeXu), který byl původně postaven na CM fontech, neměl být také tištěn v CM fontech (např. tabulky řeckých písmen a některých symbolů se liší od toho, co uživatel běžně dostane) — to je ale otázka vkusu. Co se mi však v knize skutečně nelíbilo, je to, že pokud na některých místech je třeba použít fontu generovaného METAFONTem (např. skoro na každé stránce je něco v `\tt`, dále všechny popisované fonty v dodatku o CM fontech nebo také fonty používané v obrázcích v kapitole 5), pak tyto fonty jsou tištěny v podstatně menším rozlišení než okolní text. Nevím, zda se jedná o nedopatření autorů, kteří psali text v LaTeXu, či o chybu vydavatele.

Pohádka o třech bratřích aneb „Jak praotec Čech v zájmu národa počal potomstvo v požehnaném věku“

Magdalena Wagnerová

Byl, žil starý otec Čech. Byl tak starý, že bychom ho mohli s klidem nazvat — praotec Čech. Ne snad, že by jednoho dne stanul na vrcholu hory Říp, rozhlédl se a pravil cosi o mléku a strdí, všem pohádkám netřeba věřit.

Ovšem skutečností zůstává, že se jeho předci usadili kdesi v blízkosti již řečené posvátné hory, protože na její vrchol by stěží vyšplhali — strašně rádi totiž jedli, odpočívali, jedli a zase odpočívali — a začali se množit jako králíci. A když jich bylo několik desítek, řekli si, že by mohli založit národ, protože už jsou daleko početnější nežli nějaký jeden obyčejný rod. To se potom pletli s Lechy ze sousední vesnice a kdovíjakou jinou verbeží…

A tak založili kdesi na úpatí hory Říp národ, který nazvali po sobě, jak jinak, nazvali jej národ Čechů. A dál se množili jako králíci, takže nezbývalo, než se začít o své děti starat.

„Jednou jsme národ, musíme dát svým dětem národní vzdělání,“ pravil moudře jeden z Čechů. I postavili první školu. Když postavili školu, museli sehnat učitelku. A učitelka musela něco učit. I jala se ta dobrá žena pídit po historii národa, jehož potomky má vyučovat, a tak dali Češi hlavy dohromady. Sedli u piva, ožrali se, a protože byli strašně vychloubační, sepsali pověst o posvátné hoře, která se jak přílba hlíny tyčila za humny. Konečně byli národem, jak má být, národem s národní minulostí, na kterou mohou být hrdí, kterou mohou opěvovat, oslavovat, vzývat a uctívat, malovat na obrazy, řezat ze dřeva, zapisovat do kronik i učebnic, veršovat do básní a eposů a upravovat, nadnášet a přizpůsobovat každý podle svého naturelu, jak už se s národní minulostí začasto na světě stává. A dál už to každý zná…

Tak šel čas, staří umírali, noví se rodili, první učitelku vystřídala druhá a třetí a nikdo už vlastně nevěděl, jestli nějaký Čech na hoře zvané Říp,

jež se rozkládala hned za humny, stanul anebo ne. Nikdo si nepamatoval, jestli snad jeho dědek zažil onen okamžik, kdy nějaký dědek pohlédl do kraje a prohlásil cosi o mléku a strdí.

Jen starý otec, stařičký praotec Čech ještě žil a díval se na svůj národ. Pozoroval minulost, jak se s ní svévolně zachází. A když se starosta rozhodl, že zboural hospodu, ve které se jeho dávno zemřelí sousedé kdysi ožrali a sepsali bajku o strdí, protože už její kapacity nestačí poptávce a stále více Čechů utíká k pivu místo k rodinnému krbu, z posledních sil bouchl pěstí do stolu a vzkřikl směrem ke své ženě:

„Stará, nastal čas!“

A z posledních sil počal tři syny, kterým zamýšlel předat pravdu o minulosti svého národa.

Za devět měsíců se narodili tři chlapci. Zdraví, silní, k plné spokojenosti obou — tak trochu překvapených — rodičů. Otec jim dal jména Čech, Mech a Tech.

Když hoši dorostli dospělosti, posadil je již vskutku velice starý praotec okolo stolu a z posledních sil se nadechl, aby začal vyprávět o národní minulosti, jak ji pamatoval z celého národa už jen on.

Čech viděl, že povídání bude na dlouhé lokte, a tak vzal džbánek a přičinlivě doběhl pro pivo. To podstatné z vyprávění — kterak český národ k výstupu na horu Říp přišel — neslyšel. Vrátil se ve chvíli, kdy otec vzpomínal na hospodu, ve které byly všechny události stran národní minulosti poprvé vyřčeny nahlas.

„Vida, jak je taková hospoda v životě národa důležitá!“ pomyslel si vnímavý hoch a odnesl si tento drahocenný poznatek do života…

Mech byl prostředním ze synů, a jako takový byl i středního vzrůstu a průměrné inteligence. Tolik informací najednou pro něj bylo příliš. Tolik informací, které se podstatně odlišují od těch, jež do něj učitelé pracně vtloukali po mnoho let, pro něj bylo ještě složitější. Jeho mozek, obrostlý ze všech stran, po chvíli odmítl poslušnost, a tak nebohý chlapec v tu ránu zapomněl, co otec říkal v předchozí větě. Ale protože to byl hošík vcelku s dobrým srdcem, trochu se zastyděl a řekl si:

„Musím si zapamatovat alespoň něco!“

Zapamatoval si poslední větu. Větu o otcově rozhodnutí i v požehnaném věku přece jen zplodit potomstvo, aby měl komu předávat svoji moudrost.

Tuto větu si Mech zapamatoval velice dobře, a tak celý svůj dlouhý následující život zaměřil k potomstvu, kterému chtěl po vzoru svého otce moudrost předávat. Ovšem kde nic není, ani smrt nebere. Jelikož mu

pánbůh příliš chytrosti nenadělil, jak již bylo řečeno, zůstalo v případě Mecha pouze u těch dětí…

Tech viděl, jak dopadli oba bratři, a tak se rozhodl, že na to půjde od lesa. Byl nejmenší, nejchoulostivější a nejčastěji nemocný, jak už to u těch posledních bývá. O to více se snažil nad vším vyzrát rozumem, kde mu pánbůh nenadělil sil fyzických.

Když mu matka nakázala, aby skočil pro další džbán piva, zalekl se poprvé:

„Co když dopadnu jako Čech?“

Do hospody nešel a další pivo si nechal bez uzardění přinést jako nějaký pán.

Když mu otec zdůraznil, aby si všechna jeho slova pečlivě uložil do paměti, protože je posledním z národa, kdo může znát pravdu o jeho historii, zalekl se podruhé:

„Co když dopadnu jako Mech?“

A vytvořil program, který nazval svým jménem a kam mohl uložit všechny informace, jež vyšly z úst umírajícího otce, pro případ, že by jeho mozek obrostl stejným způsobem jako bratrův.

A když oba rodiče ve vzácné shodě zdvihli varovný ukazovák, aby poukázali na jisté nebezpečí spojené s přílišným odhalováním pravd o národní minulosti, zalekl se chytrý Tech potřetí a svůj program zkomplikoval natolik, že mu rozuměla jen hrstka skutečně vyvolených. A ty zase nezajímala národní minulost, protože každý skutečný odborník se zajímá o svůj obor, a nikoli o drby.

Otec viděl, že o jeho duchovní dědictví je postaráno, a spokojeně naposledy vydechl. Matka jej do hrobu následovala po několika dnech. Čech se přihlásil k dobrovolníkům, kteří zahájili stavbu nové hospody, a Mech počal prvního potomka, na kterého již čekala další nová, mladá a nadšená učitelka s novotou zářící učebnicí o velkolepé národní minulosti.

Tech zatlačil oběma rodičům oči, důstojně je pohřbil a pro všechny případy svůj program přejmenoval na „Tex“, aby zamaskoval vlastní totožnost. Protože český národ byl odjakživa opatrný, Tech svůj národ znal, a tak pravděpodobně nejlépe věděl, co činí. A činil dobře, protože historie šla dál a objevily se desítky a stovky historiků, kterým pravda o vzniku českého národa nešla pod fousy. A tak kdyby Tech svůj program nepřejmenoval, kdovíjak by dopadl?

Dobrou noc všem texistům přeje

Magdalena Wagnerová

TEX v hypersvětě. Popis makra HyperTEX s několika bonbónky

Michal Bulant

Nenechte se, prosím mýlit poněkud bombastickým názvem! Nepůjde zde o žádný návod, ukazující, jak z poněkud (na první pohled) nepřítulného TEXu udělat barvami a obrázky zářící program (nejlépe komerční), ve kterém se můžete radostí zbláznit nad tím, na co všechno můžete kliknout (a kde všude se můžete zahrabat). V článku hodlám popisovat pouze jedno „vylepšení“ TEXu jako nástroje pro komunikaci zejména vědecké komunity — z toho je zřejmé, že půjde o vylepšení na úrovni `.dvi`-souborů a `dvi`-prohlížečů, které je k ničemu tomu, kdo používá pro prohlížení TEXových výstupů pouze tiskárnu.

Dost ale řečí a podívejme se na to, o co vlastně jde. Pod slovem **hypertext** si jistě každý něco představí. Zjednodušeně řečeno (co si pod tím představuji já), jde o takový systém pro tvorbu textu, který „umí“ do textu přidávat další informace, zejména odkazy na jiné části téhož textu (jako příklad lze uvést LATEX s jeho `\ref` a `\label`), potom také obrázky a různé zvuky (těmi se ale zmíněné makro nezabývá, a nebudu se jimi proto zabývat ani já). Tím však požadavky na hypertextový systém nekončí. Měl by umět uvedené odkazy (v počítačovém slangu nazývané **linky**) také procházet, tj. například kliknutí myši na takovém linku (místo, kde je v LATEXu `\ref`) by mělo vyvolat prohlížení místa, na které link ukazuje (`\label`). To je také v podstatě jediný přínos maker HyperTEX.

O co tedy jde?

Makra, o kterých se vyjadřuji pod souhrnným názvem **HyperTEX**, vznikla podle autorových slov (tím je Tanmoy Bhattacharya) na základě popularity World Wide Webu (zkráceně WWW) „řádícího“ na Internetu. Je to rozšíření TEXu ve formě maker, které umožňuje „vsunout“

do `.dvi`-souboru odkazy na jiné části dokumentu a rozšíření některých prohlížečů `.dvi`-souborů, umožňující těmito odkazy procházet.[1]

Jak mnozí jistě víte, soubory ve formátu HTML[2], nad kterými pracuje WWW, mohou obsahovat jisté příkazy pro formátování dokumentu (např. volbu fontu). Není to však sázecí systém, a jak podotýká autor HyperTEXu: „...je snazší implementovat hypertextové schopnosti pomocí maker TEXu než TEXové schopnosti pomocí příkazů HTML.“

V jazyku HTML (a potažmo v HyperTEXu, který se od něj schopnosti učí) lze odkazovat jednak na jiné části téhož dokumentu, jednak na jiné dokumenty. Zatímco ale ve světě WWW je častější používání druhé varianty (tzv. externí linky), dá se očekávat, že při použití HyperTEXu bude standardnější (a snadnější) varianta první. A jak vypadá odkaz při použití jazyka HTML?

```
Pokud chceš více informací o~HyperTeXu, podívej se na
<a href="http://xxx.lanl.gov/"> náš server xxx</a>
```

V okně WWW prohlížeče se pak objeví (samozřejmě i v závislosti na jiném kontextu, např. fontu nebo barvě)

Pokud chceš více informací o HyperTeXu,
podívej se na náš server xxx

Po najetí myši na podtrženou část se pak většinou na spodním okraji okna objeví celá adresa (tzv. URL[3]) odkazované části (v našem případě `http://xxx.lanl.gov/`). Po kliknutí myši by nás měl prohlížeč na ono místo „přenést“.

Toto je tzv. externí odkaz. Při použití odkazů uvnitř dokumentu (nebo při odkazech do externích dokumentů jinam než na začátek) se používá poněkud jiná syntaxe. Nejprve je třeba místo, kam budeme chtít skočit, definovat příkazem

```
<a name="symbolické jméno odkazované části">text</a>
```

a poté můžeme v jiném místě dokumentu uvést odkaz

```
<a href="volitelně jméno dokumentu#symbolické jméno odkazo-
vané části">podtržený text</a>.
```

Toto je odkaz interní, případně externí se specifikovaným cílem uvnitř jiného dokumentu.

---

[1] Jak mnohý čtenář jistě tuší, odkazy jsou do `.dvi`-souboru „dopraveny“ pomocí primitivu `\special`.

[2] Hypertext Markup Language

[3] Uniform Resource Locator

Jak to dělá HyperTEX?

HyperTEX přidává k někdy již dost nepřehlednému zástupu různých `\special`ů dalších pět příkazů, z nichž na ukázku postačí tři, totiž:

`\special{html:<a href = "hrefstring">}`,
`\special{html:<a name ="namestring">}`
a `\special{html:</a>}`

sloužící pro uvození a ukončení podtržené části.

Tato činnost je zautomatizovaná v souboru maker `hyperbasics.tex`. Ten definuje základní makra `\hyperdef` a `\hyperref` (resp. `\href`), pomocí kterých pak uživatel definuje odkaz uvnitř dokumentu (resp. vně dokumentu). Syntaxe příkazu `\hyperdef` je

`\hyperdef\TeXcs{kategorie}{jméno}{text}`,

kde `\TeXcs` znamená (v podstatě) libovolný příkaz TEXu, `kategorie` slouží pro odlišení odkazů (standardně např. *page*, *section* aj.), `jméno` rozlišuje odkazy uvnitř kategorie a `text` je vysázený text, na který je možno se odkázat pomocí příštího (nebo i předchozího, zde je ovšem nutno více průchodů TEXem) `\hyperref\TeXcs`. TEX v tomto případě „pošle“ do `.dvi`-souboru sekvenci `hmtl:<A name="kategorie.jméno">` (následovanou samozřejmě vzorně vysázeným textem a koncovým `</A>`).

Dalším hojně používaným příkazem může být `\href{URL}{text}`, který umožňuje do dokumentu vsunout odkaz na jakýkoli jiný dokument pomocí mechanismu URL. Náš dřívější příklad bychom tedy napsali nyní takto:

`Pokud chceš více informací o HyperTeXu,`
`podívej se na  \href{http://xxx.lanl.gov/}{náš server xxx}.`[4]

Jak a čím na to (pokud možno co nejpohodlněji)?

Na základě maker HyperTEX bylo postaveno několik balíků od různých autorů. Samotný autor základních maket T. Bhattacharya modifikoval `plain.bst` pro BiBTEX, dále vytvořil formát HLATEX, který pomocí modifikovaných stylů (nyní s příponou `.hty` — tedy asi htyly?) v podstatě bez vědomí autora dokumentu převádí struktury LATEXu (tj. kapitoly, sekce, odkazy, ale i stránky v obsahu) na hypertextové odkazy. Podobný balík má být i `hyper.dtx` od M. Mehlicha, ale je určen pro LATEX 2_ε. Pro

[4] TEX se samozřejmě nezabývá tím, jak ovladač odkazovaný dokument získá, pouze odkaz předá do `.dvi`-souboru.

LaTeX 2ε je určen i zcela nový, ale kompatibilní `hyperref.dtx` S. Rahtze a Y. Haralambouse.[5] Dále jsou to zejména makra `hyperwebmac.tex` a `hypercwebmac.tex`, která jsem naproti tomu používal velmi často při prohlížení různých „webovských“ zdrojových kódů.

Zejména v LaTeXu[6], který má již implementovánu jistou strukturu odkazů, je tedy používání HyperTeXu triviální (jako mnoho prefabrikovaných věcí) — stačí jen na začátku překládaného dokumentu napsat `\input hyperlatex` nebo ještě lépe vytvořit formát HLaTeX pomocí již připraveného souboru `hlatex.tex`, který od běžného uživatele LaTeXu odstíní i načítání souboru maker (takže vlastně ani neví, co se vše při překladu děje, ale na to jsme si již zvykli nejen u počítačů).

Ale jak odkazy přečíst?

Prohlížečů `.dvi`-souborů s hypertextovými rozšířeními zatím není mnoho. Pro mne (a asi většinu čtenářů z neDOSového prostředí) je ale nezajímavějším z nich program `xhdvi`, vzniklý z oblíbeného `xdvi` přidáním schopností procházet odkazy. Dalším je pak `TeXview.app` — prohlížeč pro NeXTSTEP. Prohlížeč `xhdvi` funguje přesně tak, jak bylo posáno v případě WWW-prohlížečů. Najedete-li kurzorem myši na podtržený text, objeví se ve spodní části obrazovky symbolické jméno odkazovaného místa (někdy poněkud nerozluštitelné) a po případném kliknutí se na ono místo přesunete.[7]

Hloubavého čtenáře jistě napadla otázka, co tyto prohlížeče udělají s tzv. externími linky. Jak se dá tušit, tuto úlohu nezvládnou (a ani na ni nejsou stavěné), proto si na ni zavolají pomocníka — např. standardní WWW-prohlížeč `Mosaic`, kterému požadovaný odkaz předají jako parametr. K tomu jsou dodávány shellové scripty, které je ale nutno doladit pro konkrétní systém.

[5] Tuto informaci berte, prosím, s rezervou, protože LaTeX 2ε používám jen když opravdu musím.

[6] Vzhledem ke svým velmi chabým znalostem LaTeX 2ε, slovem LaTeX bude většinou míněn „starý“ LaTeX 2.09.

[7] Kdyby to tak fungovalo i v reálném světě, asi bychom nepotřebovali psát žádné Bulletiny.

A co slibované bonbónky?

Kromě těchto prohlížečů je k dispozici ještě upravený program Toma Rokického `dvips`, nazývaný v této podobě `dvihps`. Ten umí udělat z `.dvi`-souboru PostScriptový soubor, obsahující tzv. *pdfmarks*, což jsou značky, pomocí nichž převede program `distiller` (firmy Adobe) PostScriptový soubor na nyní populární (a v budoucnu možná standard prolinkovaných textů) `.pdf`-soubor[8], který lze prohlížet např. pomocí programu `Acrobat Reader`.

Druhou možností je použít upravený program `GhostView`, který tyto značky rovněž umí procházet.

Co zbývá říci na závěr?

Všechny zmiňované balíky naleznete v archívu **CTAN**, tedy i v jeho kopii na `ftp.muni.cz` v adresáři `support/hypertex/` nebo přímo na adrese, která se několikrát objevila v článku. Dále je užitečné si přečíst `FAQ` k tomuto tématu — objevily se tuším i v diskusní skupině `cstex`.

Makra jsou užitečná zejména při prohlížení rozsáhlejších dokumentací nebo textů, kdy lze tyto texty (zejména v případě LaTeXů) po menším úsilí přeložit s přidanými hypetextovými schopnostmi, které usnadňují sledování různých odkazů v textu (nebo např. v rejstříku). Rovněž je možné do svých textů umísťovat odkazy na jiné dokumenty. To však asi nebude příliš časté použití maker zejména s ohledem na jejich (zatím) nízké rozšíření.

[8] Portable Document Format

Posun data

Karel Horák

Potřebujeme-li posunout aktuální datum, můžeme použít následující makro, které bere ve svých výpočtech v úvahu běžná pravidla pro tvorbu gregoriánského kalendáře. Pro komplikovanější účely (např. výpočet dat v hlubší minulosti) by asi bylo potřeba postupovat poněkud sofistikovaněji. Použití makra je následující:

```
\advancedate{365}
 \today

\day=26 \month=2 \advancedate{14} \today
\year 1996 \day=26 \month=2 \advancedate{14} \today
```

Nezapomeňte však, že použití sekvence `\today` v sobě skrývá nebezpečí, že při každé kompilaci bude mít váš dokument aktuální datum (které lze ovšem nastavit, jak je z uvedené ukázky vidět).

```
\edef\AtSignCode{\the\catcode`\@}
\catcode`@ 11
\newif\iff@uryear \newcount\Ye@r
\def\@dvancedate{{\Ye@r\year
         \divide\Ye@r4 \multiply\Ye@r4 \advance\Ye@r-\year
                \ifnum\Ye@r=0 \global\f@uryeartrue\fi }%
    \edef\m@nthtype{\ifcase\month\or A\or\or A\or B\or
      A\or B\or A\or A\or B\or A\or B\or A\fi}\def\@@{A}%
 \ifnum\day<28 \advance\day1
  \else\ifnum\day=28 \ifnum\month=2
    \iff@uryear\advance\day1 \else\month=3 \day=1 \fi
                    \else\advance\day1 \fi
    \else\ifnum\day=29 \ifnum\month=2 \month=3 \day=1
                       \else\advance\day1 \fi
     \else\ifnum\day=30 \ifx\m@nthtype\@@ \advance\day1
                        \else\day=1 \advance\month1 \fi
      \else\ifnum\day=31 \day=1
```

```
           \ifnum\month=12 \month=1 \advance\year1
            \else\advance\month1 \fi
      \fi
     \fi
    \fi
   \fi
  \fi
}

\newcount\h@lpcnt
\def\advancedate #1{\h@lpcnt=#1\relax
    \loop\ifnum\h@lpcnt>0
    \@dvancedate\advance\h@lpcnt-1 \repeat}
\def\today{\number\day.~\ifcase\month\or
  ledna\or února\or března\or dubna\or května\or
  června\or července\or srpna\or září\or října\or
  listopadu\or prosince\fi
  \space\number\year}

\catcode`\@=\AtSignCode
```

Vydalo:	Československé sdružení uživatelů TEXu a Nadácia Jura Hronca vlastním nákladem jako interní publikaci
Obálka:	Bohumil Bednář
Počet výtisků:	650
Uzávěrka:	17. června 1996
Tisk a distribuce:	KOPP, Máchova 16, 370 01 České Budějovice
Adresa:	CSTUG, c/o FI MU, Botanická 68a, 602 00 Brno
e-mail:	`cstug@cstug.cz`

Zřízené poštovní aliasy sdružení CSTUG:

`bulletin@cstug.cz`, `zpravodaj@cstug.cz`
  korespondence ohledně Zpravodaje sdružení

`board@cstug.cz`
  korespondence členům výboru

`cstug@cstug.cz`, `president@cstug.cz`
  korespondence předsedovi sdružení

`cstug-members@cstug.cz`
  korespondence členům sdružení

`cstug-faq`
  řešené otázky s odpověďmi navrhované k zařazení do dokumentu CSFAQ

`secretary@cstug.cz`, `orders@cstug.cz`
  korespondence administrativní síle sdružení

ftp server sdružení:
  `ftp://ftp.cstug.cz/pub/tex/`

www server sdružení:
  `http://www.cstug.cz/cstug/`

Podávání novinových zásilek povoleno Oblastní správou pošt
v Českých Budějovicích, j.zn. P 3.202/94 ze dne 19. července 1994