O možnostech pdfTEXu

František Chvála

Tento článek vznikl přepracováním textu, který jsem psal pro sebe souběžně s tím, jak jsem se seznamoval s (počeštěným) pdfTEXem a testoval jeho činnost. Užíval jsem formát pdfcsplain na následujících platformách: Windows 98, MiKTEX 2.4 s pdfTEXem verze 1.11b, Acrobat Reader 5.0, QuickTime 6.5.1; Windows XP, MiKTEX 2.4 s pdfTEXem verze 1.21a, Adobe Reader 7.0, QuickTime 6.5.2 .

Článek je sbírkou konkrétních návodů, jak využít možností pdfTEXu při tvorbě PDF dokumentů. Rozhodně nemá být učebnicí ani manuálem, spíše utříděnými poznámkami. Tématy, k nimž jsem neměl dost chuti a/nebo znalostí, jsem se nezabýval; přesto si myslím, že článek může poskytnout dobrou představu o tom, co je pdfTEX a co umí.

Pokud čtenář s pdfTEXem ještě nezačal, bylo by patrně dobré, aby si na začátek přečetl hezký přehledový článek [6] o pdfTEXu, pak si zběžně prolistoval manuál [2], pokusil se zpracovat pdfTEXem vzorový soubor z distribuce pdfTEXu (s případným odstraněním nefungujících konstrukcí), následně se vrátil k manuálu [2] a pročetl jej. Pak konečně může přejít k tomuto textu. Bude-li se problematice pdfTEXu chtít více věnovat, jistě si časem vezme k ruce referenční manuál PDF [8].

Pro vysazení článku jsem užil pdfTEX. V tištěné verzi si samozřejmě musíte odepřít zvuk a video, jakož i radost z klikání. A dále místo barev uvidíte jen stupně šedi, a ikony a rámečky ohraničující plochy odkazů se nezobrazí vůbec.

Proto současně s vydáním článku byla uvolněna i jeho elektronická verze. Najdete ji na adrese `http://bulletin.cstug.cz/bul20051.shtml` . Při jejím prohlížení už by mělo být vše, jak náleží, až na ukázky zvuku a videa. Pokud budete chtít, aby i ony se vám předvedly, bude třeba, abyste si článek ve formátu PDF zkopírovali a do stejného adresáře umístili následující soubory:

– vaši oblíbenou zvukovou nahrávku ve formátu WAV (trvající alespoň čtvrt minuty) pod jménem `audio.wav`
– vaši oblíbenou (ozvučenou) videonahrávku ve formátu MOV pod jménem `video.mov` ; může vám dobře posloužit reklamní videoklip, nalézající se v distribuci aplikace QuickTime

V PDF prohlížeči pak budete číst onen zkopírovaný dokument.

Děkuji panu Vítu Zýkovi za připomínky k rukopisu, na základě kterých jsem konečnou verzi článku opravil a upravil.

Obsah

Úvod

Program pdfTEX je rozšíření TEXu, které umožňuje vytvářet dokumenty ve formátu PDF. Aby do takového dokumentu mohly být včleněny hyperlinky a jiné specifické prostředky formátu PDF, je pdfTEX obohacen o další primitivní řídicí sekvence.

Jestliže na začátku zdrojového textu uvedeme příkaz

`\pdfoutput =` *číslo*

pak podle hodnoty *číslo* určíme režim práce pdfTEXu:

číslo `= 0` ..... výstup bude ve formátu DVI
číslo `= 1` ..... výstup bude ve formátu PDF

(Místo `0` můžeme zapsat i jakékoliv záporné celé číslo a místo `1` i jakékoliv větší celé číslo.) Při nastavení výstupu do DVI jsou určitá omezení, týkající se nových řídicích sekvencí; více o tom později (viz intermezzo).

Některé primitivy pdfTEXu mohou obsahovat zápisy v kódu PDF, a to buď operátory nebo parametry. Označení operátorů následují za operandy (postfixová notace); *příklad:* `0.8 0.3 0.4 rg` (`rg` je označení operátoru, a ten je aplikován na tři číselné hodnoty 0.8, 0.3 a 0.4). Parametry se zapisují jako dvojice

klíč hodnota

Přitom *klíč* je posloupnost znaků, zapsaná s uvozujícím lomítkem. Význam klíče je určen z kontextu (klíče zapsané touž posloupností znaků nemusí znamenat pokaždé totéž). Klíč určuje význam *hodnoty* a její datový typ. Následující (neúplný) výčet ukazuje, k jakým typům může *hodnota* náležet. V ilustrativních příkladech vždy uvádíme dvojice *klíč hodnota*:

– jméno: posloupnost znaků; zapisuje se s uvozujícím lomítkem
příklad: `/N /GoBack` (`/N` je klíč, `/GoBack` je hodnota)
– číslo: celé nebo reálné s desetinnou tečkou (ne v semilogaritmickém tvaru)
příklad: `/St 1` *příklad:* `/Rate -0.45`
– logická hodnota: `true` nebo `false`
příklad: `/Open true`
– textový řetězec: posloupnost znaků; zapisuje se v kulatých závorkách `(...)`
příklad: `/URI (http://www.cstug.cz)`
– pole: kolekce objektů (které nemusí být téhož typu); zapisují se mezi hranatými závorkami `[...]`, jsou odděleny mezerami
příklad: `/Border [0 0 1.5 [3 0.7]]` (za klíčem máme pole o čtyřech prvcích, první tři jsou čísla, čtvrtý je pole o dvou prvcích, jimiž jsou čísla)
– slovník: kolekce dvojic *klíč hodnota*; zapisují se mezi dvojitými úhlovými závorkami `<<...>>`
příklad: `<< /ShowControls true /Mode /Repeat /FWScale [7 10] >>`

Dvojice *klíč hodnota* mohou být ve slovníku uvedeny v libovolném pořadí, avšak sobě příslušné položky *klíč* a *hodnota* musí být zapsány za sebou a jedině v tomto pořadí. Ve slovníku se nesmí tentýž klíč vyskytovat vícekrát; jinak by jemu příslušná hodnota nebyla definována.

Úmluva. V dalším výkladu bude vždy termín *dimenze* znamenat TEXovsky zapsanou délkovou míru.

1. Parametry dokumentu

Nevyhovují-li vám implicitní nastavení rozměrů stránek (papíru, na který se tiskne, okna prohlížeče apod.) a umístění textu na nich, můžete je změnit. Rozměry stránek lze předepsat příkazy

`\pdfpagewidth` *dimenze* .... šířka
`\pdfpageheight` *dimenze* ... výška

Umístění textu na stránkách je definováno polohou jeho levého horního rohu; ta se předepíše příkazy

`\pdfhorigin` *dimenze* ..... vzdálenost od levého okraje stránky
`\pdfvorigin` *dimenze* ..... vzdálenost od horního okraje stránky

Doporučuje se standardní velikost parametrů `\pdfhorigin` a `\pdfvorigin` (což je 1 in) neměnit, není-li k tomu vážný důvod, a případný posun textu na stránce docílit pomocí parametrů `\hoffset` a `\voffset`. Jestliže se při zvětšení tisku příkazem `\magnification` nemá měnit velikost stránky ani se nemá posouvat plocha tisku, je vhodné rozměry stránky a rozměry definující pozici levého horního rohu tisku určit jako `true`.

Dále lze nastavit úroveň komprese textu a obrázků příkazem

`\pdfcompresslevel` *číslo*

kde *číslo* je celé číslo mezi 0 a 9 (včetně); čím je větší, tím větší je komprese, 0 = žádná komprese. Nulovou kompresi předepíšeme, chceme-li, aby PDF kód souboru byl dobře čitelný pro programátora. Přesnost reálných čísel ve výstupu do PDF předepíšeme příkazem

`\pdfdecimaldigits` *číslo*

kde *číslo* je celé číslo mezi 0 a 4 (včetně), udává počet desetinných míst. Můžeme zvolit i rozlišení natahovaných bitmapových fontů `.pk`, tak, že uvedeme

`\pdfpkresolution` *číslo*

přičemž celé *číslo* udává rozlišení v DPI. Rozumnou hodnotou je např. 600. Poznamenejme, že fonty ve formátu PK bývají v prohlížečích nekvalitně zobrazeny; proto, pokud možno, užívejte postscriptové fonty ve formátu Type 1.

2. Užití kódu PDF pro úpravy tisku

Do zdrojového textu dokumentu lze včlenit i operace měnící vzhled tisku. Zapisují se přímo v kódu PDF jako argument příkazu

`\pdfliteral{...}`.

Budeme se zabývat dvěma oblastmi použití:

– práce s barvou (sekce 2.1 a 2.2)
– geometrické transformace (sekce 2.3)

2.1. Zvýrazňování barvou

Barvu určujeme jako superpozici několika barevných složek. Máme na výběr

– systém RGB pro míchání barevných světel, primárně se užívá pro monitory; složky: červená (R – *red*), zelená (G – *green*), modrá (B – *blue*);
intenzity složek (barevných světel) jsou specifikovány čísly mezi nulou a jedničkou (0 = nulová intenzita, 1 = plná intenzita)
– systém CMYK pro míchání barevných inkoustů, primárně se užívá pro tisk; složky: modrozelená (C– *cyan*), purpurová (M– *magenta*), žlutá (Y– *yellow*), černá (K – *key colour*);
koncentrace složek (barevných inkoustů) jsou specifikovány čísly mezi nulou a jedničkou (0 = nulová koncentrace, 1 = plná koncentrace)

Systém RGB je *aditivní*: žluté světlo dostaneme sečtením červeného a zeleného světla. Systém CMYK je *subtraktivní*: barvivo se jeví žluté proto, že z bílého světla, které na něj dopadá, se odečte modrozelená a purpurová složka (absorbují se), a zbylá žlutá složka se odráží.

V systému CMYK je k dispozici samostatně i černá barva – složka K, třebaže ji lze namíchat ze složek C, M a Y. Teoreticky by v obou případech měl být výsledek týž, ve skutečnosti ale složka K dává sytou čerň, zatímco směs $C+M+Y$ spíše temně hnědou barvu. Užitím složky K se zlepší kvalita a sníží cena tisku černého textu.

Existují jednoduché algoritmy pro převod mezi systémy RGB a CMYK:

RGB → CMYK	CMYK → RGB
$K = \min\{1-R, 1-G, 1-B\}$	$R = \max\{1-C-K, 0\}$
$C = 1-R-K$	$G = \max\{1-M-K, 0\}$
$M = 1-G-K$	$B = \max\{1-Y-K, 0\}$
$Y = 1-B-K$	

Je ale třeba si uvědomit, že v praxi je tento převod nedokonalý. Jistě to vidíte v následující ukázce: ve dvojicích vzorků barev (RGB a CMYK) k opravdové shodě nedochází. (K věrnému převodu barev mezi oběma systémy se užívají programy pracující s tzv. ICC barevnými profily.)

Barva	R	G	B	C	M	Y	K	RGB	CMYK
červená	1	0	0	0	1	1	0		
žlutá	1	1	0	0	0	1	0		
zelená	0	1	0	1	0	1	0		
modrozelená	0	1	1	1	0	0	0		
modrá	0	0	1	1	1	0	0		
purpurová	1	0	1	0	1	0	0		
šedá	0.5	0.5	0.5	0	0	0	0.5		
černá	0	0	0	0	0	0	1		

2.1.1. Zvýraznění tisku. V jazyce PDF se kód pro změnu barvy tisku vyskytuje ve dvou variantách: jedna ovlivní písmo a výplně, druhá pak tahy (obrysy). Operátor pro první variantu se zapisuje malými písmeny, pro druhou variantu velkými. Program pdfTEX interpretuje linky jako výplně, mají-li větší tloušťku než 1 bp, a jako tahy v opačném případě. Přepnutí barvy tisku se docílí kódem

R G B `rg` (písmo a výplně), resp. *R G B* `RG` (tahy)

kde R, G, B jsou barevné složky v systému RGB, nebo kódem

C M Y K `k` (písmo a výplně), resp. *C M Y K* `K` (tahy)

kde C, M, Y, K jsou barevné složky v systému CMYK.

Do stupňů šedi přepínáme též kódem

šeď `g` (písmo a výplně), resp. *šeď* `G` (tahy)

kde *šeď* je číslo mezi 0 a 1. Čím menší číslo, tím tmavší odstín; černá barva odpovídá nule, bílá jedné.

Příklad:

```
\pdfliteral{1 0 0 rg}{\it Česko\/}\pdfliteral{0 0 0 rg}
je zkratkové slovo, které znamená: \pdfliteral{1 0 0 rg}%
{\it Če\pdfliteral{0 0 0 rg}chy, Morava a Slez%
\pdfliteral{1 0 0 rg}sko\/}\pdfliteral{0 0 0 rg}.
```

Výstup:

Česko je zkratkové slovo, které znamená: *Čechy, Morava a Slezsko*.

Příklad:

```
\pdfliteral{0 1 1 0 k}{\it Česko\/}\pdfliteral{0 0 0 1 k}
je zkratkové slovo, které znamená: \pdfliteral{0 1 1 0 k}%
{\it Če\pdfliteral{0 0 0 1 k}chy, Morava a Slez%
\pdfliteral{0 1 1 0 k}sko\/}\pdfliteral{0 0 0 1 k}.
```

Výstup:

Česko je zkratkové slovo, které znamená: *Čechy, Morava a Slezsko*.

V souboru `pdfcolor.tex` z distribuce ConTEXtu jsou definovány (pomocí `\pdfliteral`) TEXové řídicí sekvence přepínající do barevných odstínů písma. (Představu, které barvy jsou v tomto souboru předdefinovány, poskytne tabulka v následujícím odst. 2.1.2.) Předchozí příklad lze přepsat takto:

```
\input pdfcolor.tex
......
\Red{\it Česko\/} \Black je zkratkové slovo, které znamená:
\Red{\it Če\Black chy, Morava a Slez\Red sko\/}\Black.
```

2.1.2. Přepínače barev. Následující tabulka uvádí TEXové řídicí sekvence pro změnu barvy, definované v souboru `pdfcolor.tex`:

Řídicí sekvence	C	M	Y	K	Řídicí sekvence	C	M	Y	K
`\GreenYellow`	0.15	0	0.69	0	`\RoyalPurple`	0.75	0.90	0	0
`\Yellow`	0	0	1	0	`\BlueViolet`	0.86	0.91	0	0.04
`\Goldenrod`	0	0.10	0.84	0	`\Periwinkle`	0.57	0.55	0	0
`\Dandelion`	0	0.29	0.84	0	`\CadetBlue`	0.62	0.57	0.23	0
`\Apricot`	0	0.32	0.52	0	`\CornflowerBlue`	0.65	0.13	0	0
`\Peach`	0	0.50	0.70	0	`\MidnightBlue`	0.98	0.13	0	0.43
`\Melon`	0	0.46	0.50	0	`\NavyBlue`	0.94	0.54	0	0
`\YellowOrange`	0	0.42	1	0	`\RoyalBlue`	1	0.50	0	0
`\Orange`	0	0.61	0.87	0	`\Blue`	1	1	0	0
`\BurntOrange`	0	0.51	1	0	`\Cerulean`	0.94	0.11	0	0
`\Bittersweet`	0	0.75	1	0.24	`\Cyan`	1	0	0	0
`\RedOrange`	0	0.77	0.87	0	`\ProcessBlue`	0.96	0	0	0
`\Mahogany`	0	0.85	0.87	0.35	`\SkyBlue`	0.62	0	0.12	0
`\Maroon`	0	0.87	0.68	0.32	`\Turquoise`	0.85	0	0.20	0
`\BrickRed`	0	0.89	0.94	0.28	`\TealBlue`	0.86	0	0.34	0.02
`\Red`	0	1	1	0	`\Aquamarine`	0.82	0	0.30	0
`\OrangeRed`	0	1	0.50	0	`\BlueGreen`	0.85	0	0.33	0
`\RubineRed`	0	1	0.13	0	`\Emerald`	1	0	0.50	0
`\WildStrawberry`	0	0.96	0.39	0	`\JungleGreen`	0.99	0	0.52	0
`\Salmon`	0	0.53	0.38	0	`\SeaGreen`	0.69	0	0.50	0
`\CarnationPink`	0	0.63	0	0	`\Green`	1	0	1	0
`\Magenta`	0	1	0	0	`\ForestGreen`	0.91	0	0.88	0.12
`\VioletRed`	0	0.81	0	0	`\PineGreen`	0.92	0	0.59	0.25
`\Rhodamine`	0	0.82	0	0	`\LimeGreen`	0.50	0	1	0
`\Mulberry`	0.34	0.90	0	0.02	`\YellowGreen`	0.44	0	0.74	0
`\RedViolet`	0.07	0.90	0	0.34	`\SpringGreen`	0.26	0	0.76	0
`\Fuchsia`	0.47	0.91	0	0.08	`\OliveGreen`	0.64	0	0.95	0.40
`\Lavender`	0	0.48	0	0	`\RawSienna`	0	0.72	1	0.45
`\Thistle`	0.12	0.59	0	0	`\Sepia`	0	0.83	1	0.70
`\Orchid`	0.32	0.64	0	0	`\Brown`	0	0.81	1	0.60
`\DarkOrchid`	0.40	0.80	0.20	0	`\Tan`	0.14	0.42	0.56	0
`\Purple`	0.45	0.86	0	0	`\Gray`	0	0	0	0.50
`\Plum`	0.50	1	0	0	`\Black`	0	0	0	1
`\Violet`	0.79	0.88	0	0	`\White`	0	0	0	0

Pozor – pokud natahujeme soubor `pdfcolor.tex` (viz 2.1.1), musíme pamatovat na to, že kromě zavádění přepínačů barev se v něm také předefinovávají makra `\makeheadline` a `\makefootline`, užívaná ve výstupní rutině, a že se zde nastavuje černá barva tisku jako výchozí. Pokud tyto efekty jsou nežádoucí, upravíme si zmíněný soubor podle potřeb (a samozřejmě přejmenujeme). Upravený soubor může vypadat takto:

```
\def\GreenYellow{\pdfliteral{0.15 0    0.69 0    k 0.15 0    0.69 0    K}}
\def\Yellow     {\pdfliteral{0    0    1    0    k 0    0    1    0    K}}
\def\Goldenrod  {\pdfliteral{0    0.10 0.84 0    k 0    0.10 0.84 0    K}}
.....
\def\Gray       {\pdfliteral{0    0    0    0.50 k 0    0    0    0.50 K}}
\def\Black      {\pdfliteral{0    0    0    1    k 0    0    0    1    K}}
\def\White      {\pdfliteral{0    0    0    0    k 0    0    0    0    K}}
```

Bývá vhodné si v dokumentu nadefinovat implicitní barvu, v níž je sázen běžný text; řekněme `\def\Default{\pdfliteral{ 0 g 0 G}}`. Po barevném zvýraznění části textu pak stačí uvést příkaz `\Default` k návratu do „obyčejné“ barvy. Snáze se pak dělají případné změny v barvě základní textové masy.

Úmluva. V tělech definic představených v následujících odst. 2.1.3 a 2.1.4 se předpokládá, že příkaz `\Default`, je-li definován, je přepínačem do implicitní barvy (pokud definován není, přepne se do černé barvy).

2.1.3. Zvýraznění pozadí. Tisk textu na barevném pozadí je trochu složitější. Postup může být následující: Zjistíme rozměry textu, trochu je zvětšíme, vytiskneme pozadí jakožto `\vrule` s oněmi rozměry a nakonec přes toto pozadí vytiskneme text; přitom text má formu obsahu `\hboxu` (viz makro `\Hbox`) nebo obsahu `\vboxu` (viz makro `\Vbox`). Při užití makra `\Hbox` je přídavný horní a dolní okraj pozadí určen podpěrou `\strut`. Nevyhovují-li nám její rozměry, nadefinujeme si vlastní podpěru `\Strut`, a tu makro použije.

```
\def\Hbox#1#2#3{% první parametr: barva pozadí
                % druhý parametr: barva textu
                % třetí parametr: obsah horizontálního boxu
  \ifx\Strut\undefined\let\Strut\strut\fi
  \setbox0=\hbox{\thinspace\Strut #3\thinspace}%
  \hbox{#1\vrule height\ht0 depth\dp0 width\wd0
  \hskip-\wd0 #2\unhbox0
  \ifx\Default\undefined\pdfliteral{0 g 0 G}\else\Default\fi}}
\def\Vbox#1#2#3{% první parametr: barva pozadí
                % druhý parametr: barva textu
                % třetí parametr: obsah vertikálního boxu
  \setbox0=\hbox{\ \vbox{\vrule height10pt width0pt#3%
                         \vrule depth5pt width0pt}\ }%
  \vbox{#1\vrule height\ht0 depth\dp0 width\wd0
  \hskip-\wd0 #2\unhbox0
  \ifx\Default\undefined\pdfliteral{0 g 0 G}\else\Default\fi}}
```

Příklad:

```
Zde je ukázka \Hbox{\pdfliteral{0 0.05 0.2 0 k}}%
{\pdfliteral{0.95 0 1 0 k}}{\bf zvýraznění textu} barvami.
```

Výstup:

Zde je ukázka **zvýraznění textu** barvami.

Příklad:

```
\Vbox{\pdfliteral{0.9 g}}{\pdfliteral{0 0 1 0 k}}{\hsize5.4cm
A nyní se podívejme na trochu delší barevný text na barevném
pozadí. Vlastní text zapisujeme jako ve {\tt\char92vbox}u --
v našem případě mu předchází specifikace šířky. Pozadí trochu
přesahuje plochu textu.}
```

Výstup:

A nyní se podívejme na trochu delší barevný text na barevném pozadí. Vlastní text zapisujeme stejně jako ve \vboxu – v našem případě mu předchází specifikace šířky. Pozadí trochu přesahuje plochu textu.

Následující příklad ukazuje, jak obarvit pozadí na celé stránce.

Příklad:

```
\newbox\background
\setbox\background =
  \hbox{\pdfliteral{0.95 1 0.95 rg}% barva pozadí
    \vrule height\pdfpageheight depth0mm width\pdfpagewidth}
    % \vrule o rozměrech stránky
\def\pageBG{%
  \vbox to 0pt{\vskip-\pdfvorigin \vskip-\voffset
    \vskip-\topskip
    \hbox to 0pt{\hskip-\pdfhorigin \hskip-\hoffset
      \copy\background\hss}\vss}}
......
\vfill\eject
% následující stránka bude mít barevné pozadí:
\pageBG
```

Text stránky

2.1.4. Barevná matematika. Barevné zvýraznění vzorců nebo jejich částí se provede obdobně jako zvýraznění obyčejného textu. Předveďme si, že je třeba dát pozor na některá úskalí. V následující ukázce se snažíme vzorec vysázet v odstínu zelené barvy. Užitím operátoru `rg` ale dostaneme jakousi strakatinu:

```
$$
\pdfliteral{0 0.8 0.1 rg}
\sqrt{{1\over p^2}+{1\over q^2}}
\pdfliteral{0 g}
$$
```

$$\pdfliteral{0 0.8 0.1 rg}\sqrt{{1\over p^2}+{1\over q^2}}\pdfliteral{0 g}$$

To proto, že ne všechny součásti vyskytujících se symbolů pocházejí z fontů. Chybu napravíme užitím operátorů `rg` i `RG`:

```
$$
\pdfliteral{0 0.8 0.1 rg 0 0.8 0.1 RG}
\sqrt{{1\over p^2}+{1\over q^2}}
\pdfliteral{0 g 0 G}
$$
```

$$\pdfliteral{0 0.8 0.1 rg 0 0.8 0.1 RG}\sqrt{{1\over p^2}+{1\over q^2}}\pdfliteral{0 g 0 G}$$

Poznamenejme, že s týmž problémem se setkáme i v textu při sazbě podtržítka `\underbar`.

V další ukázce uvidíme, že může podstatně záležet na bližší volbě umístění přepínače barvy. Následující pokus vedl k nesprávné vertikální pozici exponentu:

```
$$
\pdfliteral{0 0.8 0.1 rg}
\left(\sum_{i=1}^n a_i\right)
\pdfliteral{1 0 0.2 rg}^{2k-1}
\pdfliteral{0 g}
$$
```

$$\pdfliteral{0 0.8 0.1 rg}\left(\sum_{i=1}^n a_i\right)\pdfliteral{1 0 0.2 rg}^{2k-1}\pdfliteral{0 g}$$

Zde byl přepínač do červenavé barvy umístěn bezprostředně za uzavírací kulatou závorku. Pokud jej vložíme až do exponentu, vzorec se vysadí správně:

```
$$
\pdfliteral{0 0.8 0.1 rg}
\left(\sum_{i=1}^n a_i\right)
^{\pdfliteral{1 0 0.2 rg}2k-1}
\pdfliteral{0 g}
$$
```

$$\pdfliteral{0 0.8 0.1 rg}\left(\sum_{i=1}^n a_i\right)^{\pdfliteral{1 0 0.2 rg}2k-1}\pdfliteral{0 g}$$

Následující makro `\Mbox` podbarví část vzorce. V makru je TEXový kód dotyčné části vzorce (třetí parametr) interpretován jako formule ve vnitřním matematickém módu s předepsaným `\displaystyle`. Má-li být uvažovaná část vzorce v jiném stylu, je nutno v makru tento styl předepsat před vlastním kódem.

```
\def\Mbox#1#2#3{% první parametr: barva pozadí části vzorce
                % druhý parametr: barva této části vzorce
                % třetí parametr: TeXový kód této části vzorce
  \setbox0=\hbox{$\displaystyle#3$}
  \dimen0=\ht0 \advance\dimen0 by \smallskipamount
  \dimen1=\dp0 \advance\dimen1 by \smallskipamount
  \dimen2=\wd0 \advance\dimen2 by \smallskipamount
  \def\Strut{\vrule height\dimen0 depth\dimen1 width0pt}
  \vbox{\hbox{#1\vrule height\dimen0 depth\dimen1 width\dimen2
        \hskip-\dimen2\hskip0.5\smallskipamount
        #2\box0\hskip0.5\smallskipamount}}
  \ifx\Default\undefined\pdfliteral{0 g 0 G}\else\Default\fi}
```

Schéma užití makra:

```
$$
kód úvodní nepodbarvené části vzorce
\Mbox{ barva pozadí v podbarvené části vzorce }
      { barva tisku v podbarvené části vzorce }
      { kód podbarvené části vzorce }
kód závěrečné nepodbarvené části vzorce
$$
```

nebo

```
$
kód úvodní nepodbarvené části vzorce
\Mbox{ barva pozadí v podbarvené části vzorce }
      { barva tisku v podbarvené části vzorce }
      {\textstyle  kód podbarvené části vzorce }
kód závěrečné nepodbarvené části vzorce
$
```

Příklad:

```
$$
\displaylines{
  \big(Kf\big)({\bf u})=\int\mkern-9mu\int\mkern-9mu\int
  f({\bf u'})F({\bf u'_*}) \Mbox{\pdfliteral{0.8 1 0.8 rg}}
  {\pdfliteral{1 0.2 0 rg}}{B(\vartheta,w)}\,{\rm d}{\bf u_*}
  {\rm d}\vartheta\,{\rm d}\varepsilon\cr
  \nu({\bf u})=2\pi\int\mkern-9mu\int F({\bf u_*})
  \Mbox{\pdfliteral{0.8 1 0.8 rg}}{\pdfliteral{1 0.2 0 rg}}
  {B(\vartheta,w)}\,{\rm d}{\bf u_*}{\rm d}\vartheta\cr}
$$
```

Výstup:

$$\bigl(Kf\bigr)(\mathbf{u}) = \iiint f(\mathbf{u}')F(\mathbf{u}'_*)\,B(\vartheta,w)\,\mathrm{d}\mathbf{u}_*\mathrm{d}\vartheta\,\mathrm{d}\varepsilon$$

$$\nu(\mathbf{u}) = 2\pi\iint F(\mathbf{u}_*)\,B(\vartheta,w)\,\mathrm{d}\mathbf{u}_*\mathrm{d}\vartheta$$

Následující makro `\Disp` podbarví vzorec sázený na samostatném řádku. Podbarvení je přes celou šířku tiskového zrcadla.

```
\def\Disp#1#2#3{% první parametr: barva pozadí
                % druhý parametr: barva textu
                % třetí parametr: vzorec
  \smallskip
  \setbox0=\vbox{\hbox{}#3\hbox{}}
  \dimen0=\ht0 \advance\dimen0 by \dp0
  \line{\hss
        \vbox{\offinterlineskip
              \hbox to 0pt{%
              #1\vrule height\ht0 depth\dp0 width\hsize \hss}%
              \vskip-\dimen0%
              \hbox to \hsize{#2\hfill\box0\hfill}}%
        \hss}%
  \ifx\Default\undefined\pdfliteral{0 g 0 G}\else\Default\fi
  \smallskip}%
```

Příklad:

```
\Disp{\pdfliteral{1 1 0 rg}}
  {\pdfliteral{1 0 0 rg 1 0 0 RG}}
  {$$
   \eqalign{{\bf u'}&={\bf u}-{2\over1+\kappa}
   \bf (w\cdot e)\,e \cr
   {\bf u'_*}&={\bf u_*}+{2\kappa\over1+\kappa}
   \bf (w\cdot e)\,e \cr}
   $$}
```

Výstup:

$$\mathbf{u}' = \mathbf{u} - \frac{2}{1+\kappa}(\mathbf{w}\cdot\mathbf{e})\,\mathbf{e}$$

$$\mathbf{u}'_* = \mathbf{u}_* + \frac{2\kappa}{1+\kappa}(\mathbf{w}\cdot\mathbf{e})\,\mathbf{e}$$

Podobnou funkci jako `\Disp` mají makra `\Eqbox` a `\Eqboxno`; rozdíl je v tom, že podbarvení přesahuje plochu vzorce jen o malé přídavné okraje. Makro `\Eqbox` je určeno pro nečíslované vzorce, makro `\Eqboxno` pro číslované. Jejich jádrem je pomocné makro `\eqbox`, které uvedeme nejdříve:

```
\def\eqbox#1#2#3{% první parametr: barva pozadí
                 % druhý parametr: barva vzorce
                 % třetí parametr: vzorec
  \setbox0=\hbox{$\displaystyle#3$}%
  \dimen0=\ht0 \dimen1=\dp0 \dimen2=\wd0
  \advance\dimen0 by \medskipamount
  \advance\dimen1 by \medskipamount
  \advance\dimen2 by 2\medskipamount
  \def\Strut{\vrule height\dimen0 depth\dimen1 width0pt}%
  \vbox{\hbox{#1\vrule height\dimen0 depth\dimen1 width\dimen2
              \hskip-\dimen2\hskip\medskipamount
              #2\box0\hskip\medskipamount}}%
  \ifx\Default\undefined\pdfliteral{0 g 0 G}\else\Default\fi}
```

Nyní makro `\Eqbox`:

```
\def\Eqbox#1#2#3{% první parametr: barva pozadí
                 % druhý parametr: barva vzorce
                 % třetí parametr: vzorec
  \smallskip
  \line{\hfil\eqbox{#1}{#2}{#3}\hfil}
  \smallskip}
```

Příklad:

```
\Eqbox{\pdfliteral{1 1 0 rg}}{\pdfliteral{1 0 0 rg 1 0 0 RG}}
  {$$
   \eqalign{{\bf u'}&={\bf u}-{2\over1+\kappa}
   \bf (w\cdot e)\,e\cr
   {\bf u'_*}&={\bf u_*}+{2\kappa\over1+\kappa}
   \bf (w\cdot e)\,e\cr}
   $$}
```

Výstup:

$$\mathbf{u}' = \mathbf{u} - \frac{2}{1+\kappa}(\mathbf{w}\cdot\mathbf{e})\,\mathbf{e}$$
$$\mathbf{u}'_* = \mathbf{u}_* + \frac{2\kappa}{1+\kappa}(\mathbf{w}\cdot\mathbf{e})\,\mathbf{e}$$

A ještě makro `\Eqbox`:

```
\def\Eqboxno#1#2#3#4{% první parametr: barva pozadí
                     % druhý parametr: barva vzorce
                     % třetí parametr: vzorec
                     % čtvrtý parametr: číslo vzorce
  \smallskip
  \line{\hphantom{#4}\hfil\eqbox{#1}{#2}{#3}\hfil#4}
  \smallskip}
```

Příklad:

```
\Eqboxno{\pdfliteral{1 1 0 rg}}
  {\pdfliteral{1 0 0 rg 1 0 0 RG}}
  {$$ \eqalign{{\bf u'}&={\bf u}-{2\over1+\kappa}
   \bf (w\cdot e)\,e\cr
   {\bf u'_*}&={\bf u_*}+{2\kappa\over1+\kappa}
   \bf (w\cdot e)\,e\cr} $$}
  {(123)}
```

Výstup:

$$\begin{aligned}\mathbf{u}' &= \mathbf{u} - \frac{2}{1+\kappa}(\mathbf{w}\cdot\mathbf{e})\,\mathbf{e}\\ \mathbf{u}'_* &= \mathbf{u}_* + \frac{2\kappa}{1+\kappa}(\mathbf{w}\cdot\mathbf{e})\,\mathbf{e}\end{aligned} \tag{123}$$

2.2. Směšování barev

Zabývejme se podrobněji případem, kdy barva má být nanesena do místa, kde již nějaká barva je. Dosud jsme mlčky předpokládali, že později nanesená barva nahradí v místě překrytí barvu dřívější. Takové je implicitní nastavení, nabízejí se však i jiné volby. Můžeme si totiž nadefinovat grafický stav určující výslednou barvu v místě překrytí jako funkci dříve a později nanesené barvy. Tento grafický stav si pojmenujeme (viz položku *jméno* v následujícím schématu), a pak jej můžeme aktivovat operátorem `gs`, který je užit na jméno grafického stavu; tento zápis se uvádí v argumentu příkazu `\pdfliteral`. Přitom *jméno* je posloupnost znaků s uvozujícím lomítkem, která neobsahuje znaky `( ) [ ] { } < > / %` ani mezery.

Můžeme si definovat a pojmenovat i více grafických stavů a pak podle potřeby mezi nimi přepínat. Syntaxe:

```
\pdfpageresources
  {/ExtGState << jméno << /Type /ExtGState parametry >>
                 jméno << /Type /ExtGState parametry >>
                 .........
                 jméno << /Type /ExtGState parametry >>
              >>}
```

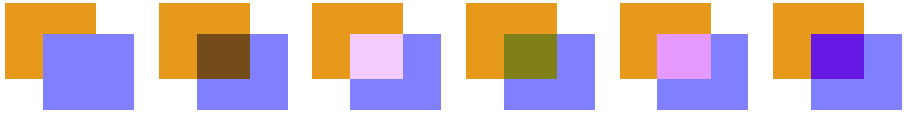
Postupně se budeme seznamovat s *parametry* grafického stavu. Především je to

`/BM` *směšovací_mód*

kde položka *směšovací_mód* určuje, jaká barva vznikne smíšením barev spolu interagujících. Konkrétně, pro tuto položku si můžeme vybrat z možností:

`/Normal`	...	pozdější barva přemaluje dřívější
`/Multiply`	...	podobný efekt jako míchání barev na paletě
`/Screen`	...	podobný efekt jako projekce barevných fólií
`/Darker`	...	za každou složku výsledné barvy vezmeme tmavší z odpovídajících složek barev původních
`/Lighter`	...	za každou složku výsledné barvy vezmeme světlejší z odpovídajících složek barev původních
`/Difference`	...	každá složka výsledné barvy je rozdílem odpovídajících složek barev původních

Ukázka. Srovnejme různé směšovací módy. V každé dvojici obdélníčků je levý namalován dříve než pravý. V systému RGB mají složky 0.9 0.6 0.1 a 0.5 0.5 1:



Uvedené módy směšování barev lze popsat matematicky. Předpokládejme, že pracujeme v systému RGB. Označme z_1 složku první barvy, z_2 odpovídající složku druhé barvy a z odpovídající složku barvy výsledné. Potom můžeme psát $z = B(z_1, z_2)$, kde B je tzv. směšovací funkce. Jak je definována pro uvedené směšovací módy, vidíte v následující tabulce.

Směšovací mód	$B(z_1, z_2)$
`/Normal`	z_2
`/Multiply`	$z_1 \cdot z_2$
`/Screen`	$z_1 + z_2 - z_1 \cdot z_2$
`/Darker`	$\min\{z_1, z_2\}$
`/Lighter`	$\max\{z_1, z_2\}$
`/Difference`	$\lvert z_1 - z_2 \rvert$

Tyto vzorce platí i pro stupně šedi. Jestliže pracujeme v systému CMYK, pak pro využití uvedených vzorců se automaticky provede převod do systému RGB.

Existují i takové způsoby směšování barev, které nelze vyjádřit po složkách. Jejich příklady nyní následují.

Další možnosti volby položky *směšovací_mód*:

`/Hue`	...	kombinace: tón první barvy – sytost a jas druhé barvy
`/Saturation`	...	kombinace: sytost první barvy – tón a jas druhé barvy
`/Luminosity`	...	kombinace: jas první barvy – tón a sytost druhé barvy

Ukázka. Srovnejme tyto směšovací módy pro uspořádání shodné s předchozím.

Příklad. V ukázkách předvedené dvojice obdélníčků byly namalovány následujícím kódem (v němž za *jméno* se dosazuje pojmenování směšovacího módu zavedené příkazem `\pdfpageresources`):

```
\def\twoRectangles#1#2{%
  #1\vrule width12mm height10mm depth0mm
  \hskip-7mm
  #2\vrule width12mm height6mm depth4mm}
\twoRectangles{\pdfliteral{jméno gs 0.9 0.6 0.1 rg}}%
  {\pdfliteral{0.5 0.5 1 rg}}
```

Poznamenejme, že v módech `/Normal`, `/Hue`, `/Saturation` a `/Luminosity` je výsledná barva závislá na pořadí, v jakém jsou obě původní barvy naneseny. V ostatních zde uvedených módech na tomto pořadí nezáleží.

Hodnoty parametrů, které byly stanoveny v předcházejících voláních `gs`, zůstávají v platnosti, dokud nejsou přepsány.

2.2.1. Překrývání ploch. Předveďme si efekty opakovaného překrývání ploch.

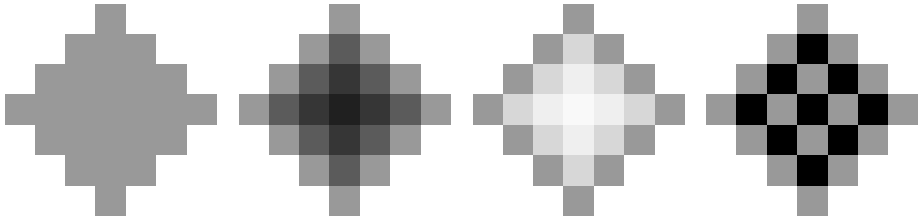
Příklad. Mějme následující obdélníky:

Namalujeme je v uvedeném pořadí, ale vhodným posunem ve vodorovném směru docílíme jejich částečného překrývání. Tuto studii vytvoříme ve čtyřech variantách – pro směšovací módy `/Normal`, `/Multiply`, `/Screen` a `/Difference`.

Všimněte si barvy míst překrytí.

```
% grafické stavy:
\pdfpageresources{/ExtGState <<
  /nor << /Type /ExtGState /BM /Normal >>
  /mul << /Type /ExtGState /BM /Multiply >>
  /sc  << /Type /ExtGState /BM /Screen >>
  /dif << /Type /ExtGState /BM /Difference >> >>}
% makro pro obdélník:
\def\rectangle#1#2{% parametry: šířka a výška obdélníka
  \dimen0=#1\hskip-\dimen0
  \vrule width2\dimen0 height#2 depth#2\hskip-\dimen0}
% makro pro skupinu částečně se překrývajících obdélníků:
\def\Rectangles#1{#1% grafický stav & barva obdélníků
  \rectangle{2mm}{14mm}\rectangle{6mm}{10mm}%
  \rectangle{10mm}{6mm}\rectangle{14mm}{2mm}}
% malujeme:
\line{\hss\Rectangles{\pdfliteral{0.6 g /nor gs}}\hskip88pt
          \Rectangles{\pdfliteral{/mul gs}}\hskip88pt
          \Rectangles{\pdfliteral{/sc  gs}}\hskip88pt
          \Rectangles{\pdfliteral{/dif gs}}\hss}
\pdfliteral{/nor gs 0 g}% návrat k implicitnímu stavu
```

Výstup:



/Normal /Multiply /Screen /Difference

2.2.2. Překrývání písmen. Zkusme postup z předchozího příkladu upravit tak, aby se místo obdélníků překrývala písmena; třeba takto:

```
\pdfpageresources{/ExtGState << ... >>}
{\font\inch=csinch scaled 750
\inch\pdfliteral{jméno gs 0.6 g}%
A\hskip-1.47cm B\hskip-1.45cm C}%
\pdfliteral{/nor gs 0 g}
```

kde *jméno* je pojmenování směšovacího módu. Zjistíme, že ve *všech* užitých směšovacích módech dostaneme obrázek, který je vytištěn vpravo od uvedeného kódu. Tedy v místech překrytí se barva nemění! Důvodem je, zhruba řečeno, to, že text „ABC“ je brán jako celek (nic na tom nemění fakt, že mezi písmeny jsou předepsány posuny a dochází k jejich překrývání), a proto je vybarven najednou. Pokud ale před každým písmenem zopakujeme vyvolání grafického stavu, vybarvení v místech překrytí bude obdobné jako v případě obdélníků.

Máme ale zajímavější možnost. Mezi grafickými stavy definujeme jeden, který je určen parametrem `/TK false`, a druhý, který jeho funkci ruší, a je určen parametrem `/TK true`. První z nich pojmenujme např. `/no-tk`, a druhý `/tk`:

```
\pdfpageresources
  {/ExtGState << .........
                 /no-tk << /Type /ExtGState /TK false >>
                 /tk << /Type /ExtGState /TK true >>
                 .........
              >>}
```

Při aktivaci grafického stavu `/no-tk` se na každý znak textu nahlíží jako na samostatný objekt, takže při jejich překrývání dochází ke skládání barev podle nastavení směšovacího módu. Grafický stav `/tk` odpovídá implicitnímu nastavení.

Příklad:

```
% grafické stavy:
\pdfpageresources{/ExtGState <<
  /nor   << /Type /ExtGState /BM /Normal >>
  /mul   << /Type /ExtGState /BM /Multiply >>
  /sc    << /Type /ExtGState /BM /Screen >>
  /dif   << /Type /ExtGState /BM /Difference >>
  /no-tk << /Type /ExtGState /TK false >>
  /tk    << /Type /ExtGState /TK true >> >>}
% makro pro skupinu částečně se překrývajících písmen:
\def\ABC#1{#1% grafický stav & barva písmen
  {\font\inch=csinch scaled 750\inch
   A\hskip-1.47cm B\hskip-1.45cm C}}
% malujeme:
\line{\hss\ABC{\pdfliteral{/nor gs /no-tk gs 0.6 g}}\hskip88pt
          \ABC{\pdfliteral{/mul gs /no-tk gs 0.6 g}}\hskip88pt
          \ABC{\pdfliteral{/sc  gs /no-tk gs 0.6 g}}\hskip88pt
          \ABC{\pdfliteral{/dif gs /no-tk gs 0.6 g}}\hss}
\pdfliteral{/nor gs /tk gs 0 g}% návrat k implicitnímu stavu
```

Výstup:

`/Normal` `/Multiply` `/Screen` `/Difference`

Grafický stav může být určen i více parametry. Kdybychom v předchozím příkladu mezi grafickými stavy definovali

```
% grafické stavy:
\pdfpageresources{/ExtGState <<
  .........
  /default << /Type /ExtGState /BM /Normal /TK true >>
  >>}
```

pak příkazem `\pdfliteral{/default gs 0 g}` bychom mohli vyvolat návrat k implicitnímu stavu místo uvedeného `\pdfliteral{/nor gs /tk gs 0 g}`.

2.2.3. Průsvitné barvy. Jinými užitečnými *parametry* příkazu

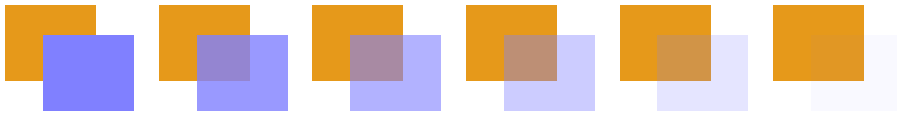
```
\pdfpageresources
  {/ExtGState << jméno << /Type /ExtGState parametry >>
                 .........
              >>}
```

jsou

`/ca` *číslo* resp. `/CA` *číslo*

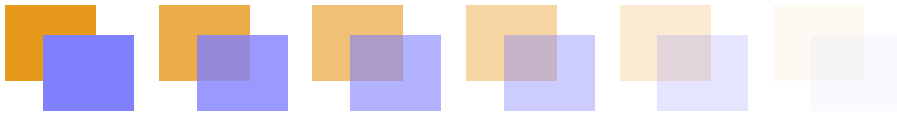
vymezující, nakolik jsou barvy průsvitné. První z nich ovlivní barvy znaků a výplní, druhý barvy tahů (obrysů). Přitom *číslo* o velikosti mezi 0 a 1 (včetně) udává míru průsvitnosti; čím je *číslo* větší, tím méně průsvitné jsou barvy: při jedničce jsou zcela neprůsvitné, při nule se ztrácejí (nejsou definovány).

Ukázka. V každé dvojici obdélníčků je levý namalován dříve než pravý, a to ve směšovacím módu `/Normal`. Barvy obdélníčků jsou definovány v systému RGB – mají složky 0.9 0.6 0.1 a 0.5 0.5 1. U druhého z obdélníčků jsou předepsány uvedené míry průsvitnosti:



1.0 0.8 0.6 0.4 0.2 0.05

Ukázka. Totéž, ale s průsvitností předepsanou shodně u obou obdélníčků:

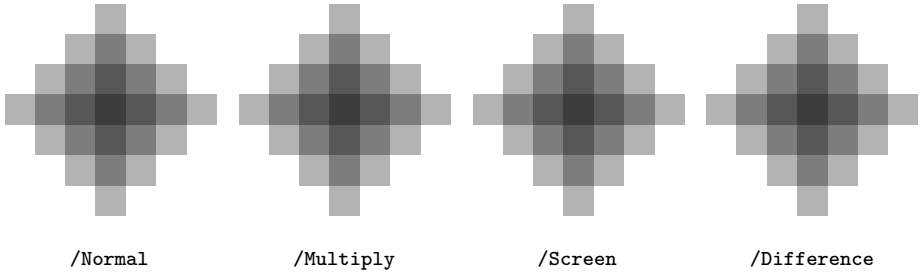


V následujícím příkladu uvidíme, že průsvitnost barev způsobí nejen dojem jakési jejich vybledlosti, ale může znatelně pozměnit efekty při jejich směšování.

Příklad. Modifikujme příklad s částečně se překrývajícími obdélníky z podsekce 2.2.1 tak, že barva obdélníků bude černá, přičemž přidáme požadavek její průsvitnosti specifikované parametrem `/ca 0.3`.

```
% grafické stavy:
\pdfpageresources{/ExtGState <<
  .........
  /op3  << /Type /ExtGState /ca 0.3 >>
  /op10 << /Type /ExtGState /ca 1 >> >>}
% makro pro obdélník:
\def\rectangle#1#2{.....}
% makro pro skupinu částečně se překrývajících obdélníků:
\def\Rectangles#1{.....}
% malujeme:
\line{\hss\Rectangles{\pdfliteral{/nor gs 0.6 g}}
          \Rectangles{\pdfliteral{/mul gs 0.6 g}}
          \Rectangles{\pdfliteral{/sc gs 0.6 g}}
          \Rectangles{\pdfliteral{/dif gs 0.6 g}}\hss}
\pdfliteral{/nor gs /op10 gs 0 g}% implicitní stav
```

Výstup:



Uvědomíte-li si, že tento příklad byl odvozen z příkladu z podsekce 2.2.1 jen změnou barvy obdélníků a její průsvitnosti, možná vás překvapí, jak podstatně se výstupy v obou příkladech liší. A co víc, v naposledy uvedeném příkladu jsou si všechny čtyři obrazce velmi podobné.

Nanesme na prázdné místo v dokumentu barvu, jejíž míra průsvitnosti α (tj. *číslo* za klíčem `/ca`, resp. `/CA`) je menší než jedna. Pak můžeme určit takovou neprůsvitnou barvu, jejímž užitím dosáhneme stejného vizuálního dojmu; konkrétně, pro každou složku z průsvitné barvy přejdeme k odpovídající složce z' neprůsvitné barvy podle předpisu:

$$z' = 1 - \alpha\,(1 - z)$$

Pokud by tedy šlo jen o barevný text, pak by pojem průsvitné barvy byl zbytečný. Jiná situace ovšem nastává při směšování barev. O tom, jak jsou definovány složky výsledné barvy, již byla řeč. Ještě uvedeme, jak je to s její průsvitností. Označíme-li míru průsvitnosti první barvy α_1, druhé barvy α_2 a výsledné barvy α, platí vztah

$$\alpha = \alpha_1 + \alpha_2 - \alpha_1 \cdot \alpha_2$$

Mají-li symboly z_1, z_2, z a B týž význam jako na začátku sekce 2.2, pak složky z výsledné barvy jsou dány následující formulí:

$$z = \left(1 - \frac{\alpha_1}{\alpha}\right) z_1 + \frac{\alpha_2}{\alpha}\left[(1 - \alpha_1)\, z_2 + \alpha_1 B(z_1, z_2)\right]$$

Tedy složky z výsledné barvy jsou dány jako jistý vážený průměr odpovídajících složek z_1 a z_2 mísících se barev a směšovací funkce $B(z_1, z_2)$. V případě, že formuli pro směšování barev nelze vyjádřit po jednotlivých složkách, pojem směšovací funkce je třeba pozměnit: nyní to bude vektorová funkce $\mathbf{B}$, která vektorům $\mathbf{z}_1$ a $\mathbf{z}_2$ složek dvou spolu se mísících barev přiřazuje vektor $\mathbf{z}$ složek výsledné barvy: $\mathbf{z} = \mathbf{B}(\mathbf{z}_1, \mathbf{z}_2)$. Pak formule pro směšování průsvitných barev bude mít následujícím tvar:

$$\mathbf{z} = \left(1 - \frac{\alpha_1}{\alpha}\right) \mathbf{z}_1 + \frac{\alpha_2}{\alpha}\left[(1 - \alpha_1)\, \mathbf{z}_2 + \alpha_1 \mathbf{B}(\mathbf{z}_1, \mathbf{z}_2)\right]$$

2.3. Geometrické transformace

Jazyk PDF umožňuje na tiskový materiál aplikovat afinní transformace: posunutí, změnu měřítek, otočení, zkosení, a jejich kombinace. V jazyce PDF jsou afinní transformace specifikovány operátorem `cm`:

$a\ \ b\ \ c\ \ d\ \ e\ \ f$ `cm`

kde a, b, c, d, e, f jsou čísla.

Význam: přechod od původních souřadnic x, y k novým souřadnicím x', y' je určen vztahem

$$\begin{aligned} x' &= ax + cy + e \\ y' &= bx + dy + f \end{aligned} \qquad \text{neboli} \qquad \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & c \\ b & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix}$$

Aktuální grafický stav je mimo jiné popsán i aktuální transformací souřadnic tiskového materiálu; je-li následně specifikována transformace operátorem `cm`, pak aktuální transformace se s touto složí (je jí modifikována).

Speciální případy:

– identická transformace:
  `1 0 0 1 0 0 cm`
– posunutí o e postscriptových bodů (bp) vodorovně a o f postscriptových bodů svisle:
  `1 0 0 1` e f `cm`
– a-násobné zvětšení ve vodorovném směru[1] a d-násobné zvětšení ve směru svislém[2]:
  a `0 0` d `0 0 cm`
– otočení o úhel ϑ:
  $\cos\vartheta$ $\sin\vartheta$ $-\sin\vartheta$ $\cos\vartheta$ `0 0 cm`
– zkosení: vodorovná osa se stočí o úhel φ doleva a svislá osa o úhel ψ doprava:
  `1` $\operatorname{tg}\varphi$ $\operatorname{tg}\psi$ `1 0 0 cm`

Pomocí operátoru `cm` můžeme transformovat části textu uložené v TEXových registrech typu „box“. Počátek souřadnic leží v referenčním bodu boxu. Před užitím operátoru `cm` zpravidla ukládáme aktuální grafický stav, abychom se k němu mohli později vrátit. K tomu slouží následující operátory:

`q` ..... uložení aktuálního grafického stavu
`Q` ..... obnovení naposledy uloženého grafického stavu

Rozměry boxu s materiálem, jenž má být transformován, je nutno předepsat jako nulové, aby nedošlo k posunu aktuálního bodu sazby. O řádné umístění příslušného tisku v dokumentu se pak musíme postarat. Schéma základního postupu pro transformaci textu může vypadat takto:

```
\setbox0=\hbox{\pdfliteral{q a b c d e f cm}text}
\wd0=0pt \ht0=0pt \dp0=0pt
posun\box0\pdfliteral{Q}posun
```

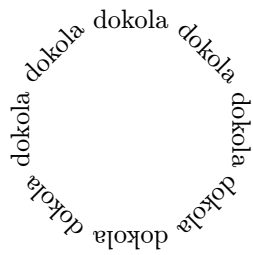
Přitom *posun* referenčního bodu sazby před výstupem boxu a *posun* po něm závisí na původních rozměrech boxu a na transformaci.

V odst. 2.3.1–2.3.8 navrhneme pdfTEXová makra pro konkrétní typy afinních transformací. V těchto makrech již budou posuny referenčního bodu zahrnuty.

[1] O zvětšení v užším slova smyslu lze mluvit, je-li $a > 1$. Pokud $0 < a < 1$, jedná se ovšem o zmenšení. A jestliže $a < 0$, pak dostáváme pravolevé překlopení se zvětšením/zmenšením, které je dáno koeficientem $|a|$.

[2] Viz pozn. 1 s příslušnými změnami: píšeme d místo a, a při $d < 0$ máme překlopení ve směru svislém.

Předepíšeme-li více transformací (ne nutně bezprostředně) za sebou, pak tyto transformace se nepřepisují, ale postupně se skládají. Předveďme si to na postupu, kterým byl vytvořen obrazec vpravo. Vidíte, že je tam každé „dokola“ pootočeno oproti předchozímu výskytu tohoto slova o 45° ve směru hodinových ručiček. To lze snadno uskutečnit pomocí cyklu `\loop` ... `\repeat`, v němž se zmíněné otočení na slovo „dokola“ opakovaně aplikuje. Obrazec byl vytvořen následujícím jednoduchým kódem:




```
\newcount\cnt % počitadlo průchodů cyklem
\setbox0=\hbox{\cnt=1 \pdfliteral{q}%
  \thinspace dokola\thinspace
  \loop \advance\cnt by 1 \ifnum \cnt<9
    \pdfliteral{0.7071 -0.7071 0.7071 0.7071 0 0 cm}%
    \thinspace dokola\thinspace
  \repeat}%
\wd0=0pt \ht0=0pt \dp0=0pt
\box0\pdfliteral{Q}
```

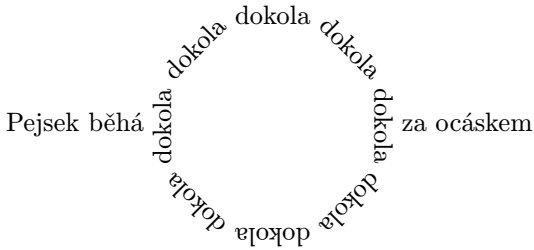
Všimněte si, že uvnitř cyklu se nevyskytují operátory `q` a `Q`, a uvědomte si, že $\sin 45° = 0{,}7071 = \cos 45°$. Má-li být obrazec součástí textu, je nutno si trochu pohrát s určením jeho umístění a k tomu účelu uvedený kód vhodně modifikovat. Vidíte to v následující ukázce.

Příklad:

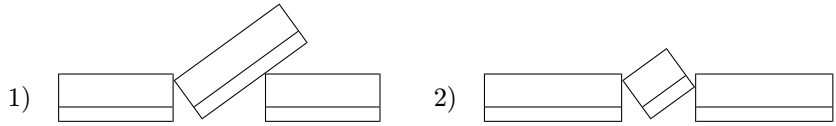
```
\newcount\cnt % počitadlo průchodů cyklem
\newcount\yshift % uloží se vertikální posun obrazce
% do boxu 1 se uloží slovo, jehož opakováním se tvoří obrazec:
\setbox1=\hbox{\thinspace dokola\thinspace}
% do \dimen1 se uloží vzdálenost referenčního bodu boxu 1
%  od levé strany obrazce:
\dimen1=0.7071\wd1\advance\dimen1 by \ht1
\noindent Pejsek běhá
% do \dimen2 se uloží vzdálenost referenčního bodu boxu 1
%  od vodorovné osy:
\dimen2=0.7071\wd1
\advance \dimen2 by 0.5\wd1
% k \dimen2 přičteme 2,5 pt, aby představovala vzdálenost
%  referenčního bodu boxu 1 od vodorovné osy textu:
\advance \dimen2 by 2.5pt
```

```
% \yshift je číselně rovno dimenzi \dimen2 vyjádřené v bp:
\yshift=\dimen2 \divide \yshift by 65782 % 1 bp = 65782 sp
\setbox0=\hbox{\cnt=1%
% začínáme se slovem zapsaným vodorovně zleva doprava:
  \pdfliteral{q 1 0 0 1 0 \the\yshift\space cm}\copy1
% postupně vytváříme obrazec:
  \loop \advance\cnt by 1 \ifnum \cnt<9
    \pdfliteral{0.7071 -0.7071 0.7071 0.7071 0 0 cm}\copy1
  \repeat}%
\wd0=0pt \ht0=0pt \dp0=0pt
\hskip\dimen1
\box0\pdfliteral{Q}%
% v \dimen1 bude vzdálenost referenčního bodu boxu 1
%  od pravé strany obrazce:
\advance\dimen1 by \wd1 \hskip\dimen1\ %
za ocáskem.
```

Výstup:



2.3.1. Otočení o ostrý úhel doleva. Text se makrem `\rotate` otočí o předepsaný úhel ϑ. V makru jsou (podmínkou `ifdim`) rozlišeny dva kvalitativně odlišné případy:



(Jsou zde znázorněny `\hboxy` s vyznačeným účařím, prostřední z nich otočený makrem `\rotate`. Všimněte si pozice pravého dolního rohu otáčeného boxu.) Otočení se provede kolem referenčního bodu, pak následuje posun otočeného boxu doprava a ještě příslušný posun textu za ním. Elementární trigonometrické úvahy, na kterých je makro založeno, zde nejsou rozebírány: jejich sledování by pro čtenáře bylo určitě stejně nudné, jako jejich popisování pro autora.

```
\def\rotate#1#2#3{% první parametr: text
                  % druhý parametr: sin(theta)
                  % třetí parametr: cos(theta)
  \setbox0\hbox{\pdfliteral{q #3 #2 -#2 #3 0 0 cm}#1}%
  \dimen0=#2\wd0 \advance\dimen0 by -#3\dp0 \dimen1=#2\ht0
  \ifdim\dimen0>\ht0  % případ 1):
    \dimen2=#3\ht0 \advance\dimen2 by \dp0
    \dimen3=1000sp \dimen3=#2\dimen3
    \divide\dimen2 by\dimen3 \dimen2=1000\dimen2
  \else               % případ 2):
    \dimen2=#3\wd0 \advance\dimen2 by #2\dp0
  \fi
  \dimen0=#2\wd0 \advance\dimen0 by #3\ht0                %%%%
  \dimen3=#3\dp0                                          %%%%
  \wd0=0pt \ht0=0pt \dp0=0pt
  \hskip\dimen1
  \vrule height\dimen0 depth\dimen3 width0pt % podpěra
  \box0\pdfliteral{Q}\hskip\dimen2\relax}
```

Příklad:

```
Stoupejme \rotate{stále výš a výše}{0.64279}{0.76604}
-- až na střechu světa.
```

Výstup:

Stoupejme stále výš a výše – až na střechu světa.

Pozn.: úhel otočení je $40°$; $\sin 40° = 0{,}64279$; $\cos 40° = 0{,}76604$.

Čtěte:

Teď se budeme zabývat takovou úpravou více-
řádkového textu, aby řádky směřovaly šikmo
vzhůru, byly rovnoběžné při konstantní vzdá-
lenosti mezi sousedícími účařími (i za cenu pří-
padného překrývání se řádků), měly předepsa-
nou délku (až na východový řádek, který smí
být kratší), a začínaly na pomyslné vodorovné
přímce.

Text, který jste právě přečetli, svým tvarem názorně předvádí popisovanou situaci. Úpravu provede makro `\Rotate` se čtyřmi parametry: prvním z nich je délka řádků (TEXovská dimenze), zbývající tři odpovídají parametrům makra `\rotate`, kterého se v makru `\Rotate` vskutku využívá – k otočení (o úhel ϑ) `\vtop`u s textem, který je přeformátován (příkazem `\parshape`) tak, aby řádky zůstaly stejně dlouhé, ale jejich odsazení (ukládaná do registru `\rowindent`) postupně vzrůstala o vhodně určenou hodnotu (uchovávanou v registru `\rowshift`).

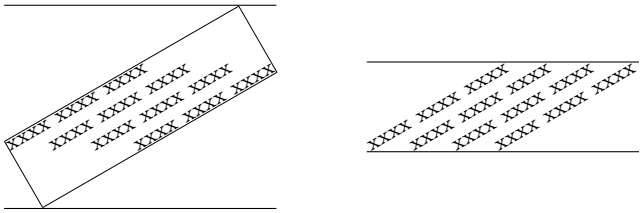
```
\newcount\row       % počitadlo řádků
\newdimen\rowskip   % vzdálenost účaří řádku od účaří \vtop'u
\newdimen\rowshift  % odsazení začátku řádku vzhledem
                    %  k začátku předchozího řádku
\newdimen\rowindent % odsazení začátku řádku vzhledem
                    %  k začátku prvního řádku
\newtoks\rowdata    % parametry příkazu \parshape
%
\def\Rotate#1#2#3#4{% první parametr:  délka řádků
                    % druhý parametr:  text
                    % třetí parametr:  sin(theta)
                    % čtvrtý parametr: cos(theta)
  {\hsize#1
  \setbox1=\vtop{\lineskiplimit=-\maxdimen\noindent#2}%
  \row=1 \rowskip=0pt \rowindent=0pt
  \rowdata{0pt\space \the\hsize}%
  \rowshift=1000sp \dimen3=#3\rowshift \dimen4=#4\rowshift
  % \dimen4 / \dimen3 = cotg(theta)
  \rowshift=\baselineskip
  \multiply \rowshift by \dimen4 \divide \rowshift by \dimen3
  % \rowshift = cotg(theta) * \baselineskip
  \loop \ifdim \rowskip<\dp1
    \advance \row by 1
    \advance \rowskip by \baselineskip
    \advance\rowindent by \rowshift
    % makro \add načítá do \rowdata dvojice dimenzí
    %  "odsazení řádku" a "délka řádku"
    \edef\add{\rowdata=
      {\the\rowdata\space\the\rowindent\space\the\hsize}}\add
  \repeat
  \rotate{\vtop{\noindent\parshape\row\space\the\rowdata #2}}%
    {#3}{#4}%
  }}
```

Šikmo orientovaný ilustrativní text byl vysazen takto:

```
\Rotate{7cm}{Teď se budeme zabývat takovou úpravou ...
 ... začínaly na pomyslné vodorovné přímce.}{0.5}{0.8660}
```

Pozn.: úhel otočení je 30°; $\sin 30° = 0{,}5$; $\cos 30° = 0{,}8660$.

Uvedené makro `\Rotate` má jeden nedostatek: jeho výslednému objektu je přiřazen větší vertikální rozměr. Představu, proč tomu tak je, si uděláte na základě následujícího levého schématu – vidíte, že do tohoto rozměru, jenž je představován vzdáleností mezi oběma vodorovnými linkami, jsou zahrnuty i výšky obou prázdných rohů (otočeného) boxu ohraničujícího text.



Aby se přiřadil skutečný vertikální rozměr, znázorněný vzdáleností mezi vodorovnými linkami na pravém schématu, pro účely makra `\Rotate` upravíte makro `\rotate` modifikací příkazů, které jsou v jeho popisu na šestém a pátém řádku zdola (jsou označeny znaky `%%%` vpravo), tak, aby místo `\ht0` figurovala `\hsize` a místo `\dp0` hloubka posledního řádku `\vtopu`.

2.3.2. Otočení o pravý úhel doleva. K tomuto účelu můžeme použít makro `\rotate`, totiž

```
\rotate{text}{1}{0}
```

Přesto je užitečné zavést specializované (a jednodušší) makro `\turn`, jehož jediným parametrem je otáčený text.

```
\def\turn#1{\setbox0\hbox{\pdfliteral{q 0 1 -1 0 0 0 cm}#1}%
  \dimen0=\wd0 \dimen1=\ht0 \dimen2=\dp0
  \wd0=0pt \ht0=0pt \dp0=0pt
  \hskip\dimen1\vrule height\dimen0 depth0pt width0pt
  \box0\pdfliteral{Q}\hskip\dimen2\relax}
```

Příklad:

```
\turn{Vzhůru} již hlavu, národe,
k nebi své \turn{vzdvihni} oči!
```

Výstup:

Vzhůru již hlavu, národe, k nebi své vzdvihni oči!

Příklad. V tabulkách občas potřebujeme otočit o devadesát stupňů některou položku, např. delší text v záhlaví úzkého sloupce. K tomu lze využít makro `\turn`. Je to ilustrováno následující ukázkou tabulky názvů velikostí písma. Makra `\topstrut` a `\botstrut` definují větší horní a dolní podpěru (pro odsazení vodorovných linek tabulky od textu) a makro `\sstrut` zavádí podpěru trochu menší než `\strut`.

```
\def\topstrut{\vrule height11pt width0pt}
\def\botstrut{\vrule depth4pt width0pt}
\def\sstrut{\vrule height7.75pt depth2.75pt width0pt}
\def\cbox#1{\hbox to 60pt{\hfil\sstrut#1\hfil}}
%
\vbox{%
  \offinterlineskip
  \halign{%
  \strut\vrule width1pt\hfil\ #\ &\vrule\ #\
    &\vrule\ \hfil#\ \vrule width1pt\cr
  \noalign{\hrule height 1pt}%
  \botstrut\turn{Stupeň písma [dd] }\hfil
    &\vbox{\cbox{Pojmenování}
           \cbox{velikosti}
           \cbox{písma}\vskip0pt}%
    &\turn{Šířka čtverčíku [mm] }%
     \hfil\cr
  \noalign{\hrule height 1pt}%
  5&\topstrut perl&1{,}880\cr
  6&nonpareille&2{,}256\cr
  7&kolonel&2{,}632\cr
  8&petit&3{,}009\cr
  9&borgis&3{,}385\cr
  10&garmond&3{,}761\cr
  11&breviář&4{,}137\cr
  12&cicero&4{,}513\cr
  14&střední&5{,}265\cr
  16&tercie&6{,}017\cr
  20&\botstrut text&7{,}521\cr
  \noalign{\hrule height 1pt}}}
```

Výstup ⟶

Stupeň písma [dd]	Pojmenování velikosti písma	Šířka čtverčíku [mm]
5	perl	1,880
6	nonpareille	2,256
7	kolonel	2,632
8	petit	3,009
9	borgis	3,385
10	garmond	3,761
11	breviář	4,137
12	cicero	4,513
14	střední	5,265
16	tercie	6,017
20	text	7,521

2.3.3. Otočení vzhůru nohama. Text se makrem `\upsidedown` otočí vzhůru nohama tak, aby pozice účaří zůstala zachována a nedošlo k horizontálnímu posunu vzhledem k poloze původního textu.

```
\def\upsidedown#1{\setbox0\hbox{%
    \pdfliteral{q -1 0 0 -1 0 0 cm}#1}%
  \dimen0=\wd0 \dimen1\ht0 \dimen2\dp0
  \wd0=0pt \ht0=0pt \dp0=0pt
  \hskip\dimen0\vrule height\dimen2 depth\dimen1 width0pt
    \box0\pdfliteral{Q}}
```

Příklad:

```
\upsidedown{Takhle se píše u protinožců.}
```

Výstup:

Takhle se píše u protinožců.

2.3.4. Změna měřítek. Následující makro `\scale` způsobí natažení resp. stažení textu ve vodorovném a/nebo svislém směru. Jestliže příslušný koeficient (druhý a třetí parametr makra) je větší než jedna, jedná se o natažení, je-li menší než jedna (a kladný), pak se jedná o stažení.

```
\def\scale#1#2#3{% první parametr: text
                 % druhý parametr: koeficient natažení/stažení
                 %                  ve vodorovném směru
                 % třetí parametr: koeficient natažení/stažení
                 %                  ve svislém směru
  \setbox0\hbox{\pdfliteral{q #2 0 0 #3 0 0 cm}#1}%
  \dimen0=#2\wd0 \dimen1=#3\ht0 \dimen2=#3\dp0
  \wd0=0pt \ht0=0pt \dp0=0pt
  \leavevmode\vrule height\dimen1 depth\dimen2 width0pt
  \box0\pdfliteral{Q}\hskip\dimen0\relax}
```

Příklad:

```
\scale{Dlouhý}{0.6}{3.5}, \scale{Široký}{3}{0.75},
\pdfliteral{0.65 g}Bystrozraký\pdfliteral{0 g}
```

Výstup:

Dlouhý, **Široký**, Bystrozraký

2.3.5. Překlopení okolo svislé osy. Kromě příslušné změny měřítka působící překlopení zařídí makro `\hflip` i vhodný horizontální posun překlopeného textu.

```
\def\hflip#1{%
  \setbox0\hbox{\pdfliteral{q -1 0 0 1 0 0 cm}#1}%
  \dimen0=\wd0 \dimen1\ht0 \dimen2\dp0
  \wd0=0pt \ht0=0pt \dp0=0pt
  \hskip\dimen0\vrule height\dimen1 depth\dimen2 width0pt
  \box0\pdfliteral{Q}}
```

Příklad:

```
Cesta tam \dots\ \hflip{a zase zpátky} \dots
```

Výstup:

Cesta tam ... ʞʎʇɐzdɹǝz ǝsɐz ɐ ...

2.3.6. Překlopení okolo účaří.

```
\def\vflip#1{%
  \setbox0\hbox{\pdfliteral{q 1 0 0 -1 0 0 cm}#1}%
  \dimen0=\wd0 \dimen1\ht0 \dimen2\dp0
  \wd0=0pt \ht0=0pt \dp0=0pt
  \vrule height\dimen2 depth\dimen1 width0pt
  \box0\pdfliteral{Q}\hskip\dimen0\relax}
```

Příklad:

```
\vbox{%
  \font\bigger=csr12\bigger
  \hbox{%
    Pověz mi, mé zrcadlo,
    kdo je na světě nejkrásnější?}
    \vskip-0.6\baselineskip
    \hbox{\pdfliteral{0.4 g}%
      \vflip{Pověz mi, mé zrcadlo,
             kdo je na světě nejkrásnější?}%
      \pdfliteral{0 g}}}
```

Výstup:

Pověz mi, mé zrcadlo, kdo je na světě nejkrásnější?
Pověz mi, mé zrcadlo, kdo je na světě nejkrásnější?

2.3.7. Zkosení ve vodorovném směru. Makro `\slant` osu y pootočí o úhel zešikmení ψ, osu x nemění:

```
\def\slant#1#2{% první parametr: text
               % druhý parametr: tg(psi)
  \setbox0\hbox{\pdfliteral{q 1 0 #2 1 0 0 cm}#1}%
  \dimen0=\wd0 \dimen1=#2\ht0 \dimen2=#2\dp0
  \wd0=0pt \ht0=0pt \dp0=0pt
  \ifdim\dimen1>0sp
    \hskip\dimen2
  \else\hskip-\dimen1
  \fi
  \box0\pdfliteral{Q}\hskip\dimen0%
  \ifdim\dimen1>0sp
    \hskip\dimen1
  \else
    \hskip-\dimen2
  \fi}
```

Příklad:

```
{\font\BigR=csr10 at 17pt \BigR
\slant{HAF}{-0.45} \slant{HAF}{-0.3} \slant{HAF}{-0.15}
\slant{HAF}{0} \slant{HAF}{0.15} \slant{HAF}{0.3}
\slant{HAF}{0.45}}
```

Výstup:

HAF HAF HAF HAF *HAF* *HAF* *HAF*

Velikost druhého parametru 0,16667 v makru `\slant` odpovídá sklonu šikmých písem v rodině Computer Modern. Srovnejte: v následující ukázce je první „HAF“ napsáno standardním šikmým písmem (`cssl10 at 17pt`), druhé písmem kolmým (`csr10 at 17pt`), avšak zešikmené makrem `\slant` s oním parametrem 0,16667.

Zdrojový text:

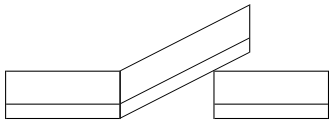
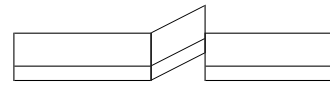
```
{\font\BigR=csr10 at 17pt\font\BigS=cssl10 at 17pt
\BigS HAF \BigR\slant{HAF}{0.16667}}
```

Výstup:

HAF HAF

Nevidíte rozdíl.

2.3.8. Zkosení ve svislém směru. Makro `\slope` pootočí osu x o úhel zešikmení φ, osu y nezmění. Podobně jako v makru `\rotate` (viz odst. 2.3.1) rozlišíme dva případy, schematicky znázorněné takto:

1)  2) 

```
\def\slope#1#2#3{% první parametr: text
                 % druhý parametr: tg(phi)
                 % třetí parametr: ctg(phi)
  \setbox0\hbox{\pdfliteral{q 1 #2 0 1 0 0 cm}#1}%
  \dimen0=#2\wd0 \advance\dimen0 by -\dp0
  \dimen1=\dp0
  \ifdim\dimen0>\ht0  % případ 1):
    \dimen2=\ht0 \advance\dimen2 by \dp0
    \dimen2=#3\dimen2
  \else               % případ 2):
    \dimen2=\wd0
  \fi
  \dimen0=#2\wd0 \advance\dimen0 by \ht0
  \wd0=0pt \ht0=0pt \dp0=0pt
  \vrule height\dimen0 depth\dimen1 width0pt
  \box0\pdfliteral{Q}\hskip\dimen2\relax}
```

Příklad:

```
Stoupejme \slope{stále výš a výše}{0.57735}{1.73205}
a držme se rovně.
```

Výstup:

Stoupejme *stále výš a výše* a držme se rovně.

Pozn.: úhel otočení je 30°; $\operatorname{tg} 30° = 0{,}57735$, $\operatorname{ctg} 30° = 1{,}73205$.

2.4. Globální úpravy tisku

Úpravy tisku, které jsou v některém místě dokumentu zprostředkované příkazem `\pdfliteral`, s kódem PDF pro změnu barvy písma a pro geomet-

rické transformace, nepřežijí stránkový zlom! Potřebujeme-li, aby se tyto úpravy provedly na *každé* straně dokumentu, zahrneme je do výstupní rutiny. V případě formátu plain TEX předefinujeme makra `\plainoutput` ([1], str. 262) a `\pagecontents` ([1], str. 252) a token `\headline`.

2.4.1. Zrcadlový tisk. Ze vztahů $1\,\mathrm{pt} = 2^{16}\,\mathrm{sp} = 65\,536\,\mathrm{sp}$ a $1\,\mathrm{bp} = 1{,}00375\,\mathrm{pt}$ plyne $1\,\mathrm{bp} = 65\,782\,\mathrm{sp}$. Jestliže tedy číslo, vyjadřující TEXovou dimenzi v jednotkách sp prodělíme číslem 65 782, získáme číselné vyjádření této dimenze v postscriptových bodech (bp). Pro zrcadlový tisk dokumentu můžeme použít následující vzor:

```
\catcode`\@=11
\newcount\flipshift \newcount\backshift
% obsahem \flipshift bude celé číslo udávající velikost
%   \hsize - \hoffset v bp:
\flipshift=\hsize \advance\flipshift by -\hoffset
\divide\flipshift by 65782
% obsahem \backshift bude celé číslo udávající velikost
%   \hsize + \hoffset v bp:
\backshift=\hsize \advance\backshift by \hoffset
\divide\backshift by 65782
\def\pagecontents{%
  \ifvoid\topins
  \else
    \pdfliteral{-1 0 0 1 \the\flipshift\space 0 cm}%
    \unvbox\topins
    \pdfliteral{-1 0 0 1 \the\flipshift\space 0 cm}%
  \fi
  \dimen0=\dp255
  \pdfliteral{-1 0 0 1 \the\flipshift\space 0 cm}%
  \unvbox255
  \ifvoid\footins
  \else
    \vskip\skip\footins\footnoterule\unvbox\footins
  \fi
  \ifr@ggedbottom \kern-\dimen0 \vfill \fi}
\catcode`\@=12
\headline{%
  \pdfliteral{-1 0 0 1 \the\flipshift\space 0 cm}%
  Text záhlaví
  \pdfliteral{-1 0 0 1 -\the\backshift\space 0 cm}}
```

2.4.2. Barevný podtisk a barevné písmo. Následující ukázka předvádí způsob úpravy výstupní rutiny tak, aby byly nastaveny barvy různé pro

– podtisk (pozadí)
– text záhlaví
– všechen ostatní text

```
% barva pozadí:
\def\bgcolor{\pdfliteral{0.9 0.8 0.7 rg}}
% barva textu záhlaví:
\def\headcolor{\pdfliteral{0.2 0.9 0.1 rg 0.2 0.9 0.1 RG}}
% barva ostatního textu:
\def\textcolor{\pdfliteral{0.2 0 0.9 rg 0.2 0 0.9 RG}}
\def\BG{% pozadí -- vytvoří se jako \vrule o rozměrech stránky
  \vbox to 0pt{%
    \vskip-\pdfvorigin \vskip-\voffset
    \hbox to 0pt{%
      \hskip-\pdfhorigin \hskip-\hoffset
      \bgcolor
      \vrule height\pdfpageheight depth0mm width\pdfpagewidth
      \hss}%
    \vss}}
\def\plainoutput{%
  \shipout\vbox{%
    \BG \makeheadline \pagebody \makefootline}%
  \advancepageno
  \ifnum\outputpenalty>-20000 \else \dosupereject \fi}
\catcode`\@=11
\def\pagecontents{%
  \ifvoid\topins
  \else
    \textcolor\unvbox\topins
  \fi
  \dimen0=\dp255 \textcolor\unvbox255
  \ifvoid\footins
  \else
    \vskip\skip\footins\footnoterule\unvbox\footins
  \fi
  \ifr@ggedbottom \kern-\dimen0 \vfill \fi}
\catcode`\@=12
  \headline{\headcolor Text záhlaví }
```

3. Úprava okrajů tiskového zrcadla

Začátek této kapitoly je napsán čtyřikrát – pokaždé s jiným formátováním. K tomu za chvíli více. Teď čtěte (a nemusíte čtyřikrát, stačí jednou):

(1) Některé znaky mohou zdánlivě narušovat zarovnání pravého či levého okraje tiskového zrcadla, nacházejí-li se na samém konci nebo začátku řádku. Bývají to znaky o velké světlosti. Zvláště tečka, čárka, středník, apostrof, uvozovky, rozdělovací znaménko apod. Může se zdát, že v takovýchto místech jejich výskytu je okraj tisku jakoby „vykousnut“. Abychom tento optický klam potlačili, můžeme předepsat určité malé posunutí těchto znaků směrem vně bloku.

(2) Některé znaky mohou zdánlivě narušovat zarovnání pravého či levého okraje tiskového zrcadla, nacházejí-li se na samém konci nebo začátku řádku. Bývají to znaky o velké světlosti. Zvláště tečka, čárka, středník, apostrof, uvozovky, rozdělovací znaménko apod. Může se zdát, že v takovýchto místech jejich výskytu je okraj tisku jakoby „vykousnut“. Abychom tento optický klam potlačili, můžeme předepsat určité malé posunutí těchto znaků směrem vně bloku.

(3) Některé znaky mohou zdánlivě narušovat zarovnání pravého či levého okraje tiskového zrcadla, nacházejí-li se na samém konci nebo začátku řádku. Bývají to znaky o velké světlosti. Zvláště tečka, čárka, středník, apostrof, uvozovky, rozdělovací znaménko apod. Může se zdát, že v takovýchto místech jejich výskytu je okraj tisku jakoby „vykousnut“. Abychom tento optický klam potlačili, můžeme předepsat určité malé posunutí těchto znaků směrem vně bloku.

(4) Některé znaky mohou zdánlivě narušovat zarovnání pravého či levého okraje tiskového zrcadla, nacházejí-li se na samém konci nebo začátku řádku. Bývají to znaky o velké světlosti. Zvláště tečka, čárka, středník, apostrof, uvozovky, rozdělovací znaménko apod. Může se zdát, že v takovýchto místech jejich výskytu je okraj tisku jakoby „vykousnut“. Abychom tento optický klam potlačili, můžeme předepsat určité malé posunutí těchto znaků směrem vně bloku.

Velikost posunutí znaků na konci řádku doprava se specifikuje řídicí sekvencí `\rpcode`; velikost posunutí znaků na začátku řádku doleva pak řídicí sekvencí `\lpcode`. Syntaxe:

`\rpcode` *font kód* `=` *přesah*

resp.

`\lpcode` *font kód* `=` *přesah*

Zde *font* je přepínač fontu, jehož se nastavení posunu znaků týká, nebo řídicí sekvence `\font`, která v této souvislosti značí aktivní font; *kód* je ASCII kód příslušného znaku, tj. jedna z možností

- dekadické číslo
- hexadecimální číslo jemuž předchází `"`
- oktalové číslo jemuž předchází `'`
- onen znak jemuž předchází `‘\`
- `\hyphenchar` *font* (je-li uvažovaným znakem rozdělovací znaménko)

a *přesah* je velikost posunu krajního znaku doprava (`\rpcode`) resp. doleva (`\lpcode`) v promilech jednotky em. Zda a jak se tato nastavení uplatní (při zpracování odstavce), se určí příkazem

`\pdfprotrudechars =` *číslo*

přičemž *číslo* je buď `0` ..... specifikovaný posun znaků se neprovede
nebo `1` ..... standardní řádkové zlomy, pak posun znaků natažením řádků
nebo `2` ..... řádkové zlomy berou v úvahu posun znaků

(Místo `0` můžeme zapsat i jakékoliv záporné celé číslo a místo `2` i jakékoliv větší celé číslo.) Uplatní se ta hodnota `\pdfprotrudechars`, která byla přiřazena naposled před zpracováním odstavce.

Příklad. Tři úzké odstavce (1), (2), (3) ze začátku této kapitoly byly vysazeny (po řadě) s hodnotami `\pdfprotrudechars=0`, `\pdfprotrudechars=1` a `\pdfprotrudechars=2`, a s parametry

```
\rpcode\tenrm‘\,=150                  % čárka
\rpcode\tenrm‘\.=180                  % tečka
\rpcode\tenrm\hyphenchar\tenrm=200    % rozdělovací znaménko
\lpcode\tenrm 254=240                 % otevírací uvozovky
\hsize=133.8pt \parindent=0pt \tolerance=700
```

(přičemž v textu byla povolena dělení: `zna\-mén\-ko výsky\-tu může\-me po\-su\-nu\-tí`). Všimněte si odlišného zarovnání hran bloků (v prvním a druhém sloupci) a odlišných řádkových zlomů (ve druhém a třetím sloupci).

Parametr *přesah* v příkazu `\rpcode`, resp. `\lpcode` může být záporný; v takovém případě se příslušný znak posune směrem dovnitř plochy odstavce, tj. doprava, stojí-li na začátku řádku a doleva, stojí-li na konci řádku. Této možnosti můžeme využít k tomu, abychom potlačili přečnívání některých písmen na kon-

cích řádků přes hrany odstavce, což nastává zejména při sazbě kurzívním nebo skloněným písmem.

Příklad. Všimněte si prvního sloupce následujícího (myšlenkově závažného) textu: na začátku druhého řádku písmeno „*V*“ utíká trochu doprava, na konci téhož řádku přečnívá písmeno „*f*“ a na začátku posledního řádku přečnívá písmeno „*j*“; v druhém sloupci je tento jev potlačen:

Mňau mrňau mňau mrňau mňau
Vrr haf haf vrr haf haf vrr haf haf
Mňau mrňau mňau mrňau mňau
je to vysazeno normálně.

Mňau mrňau mňau mrňau mňau
Vrr haf haf vrr haf haf vrr haf haf
Mňau mrňau mňau mrňau mňau
je to vysazeno s korekcí.

A takto vypadá zdrojový text ukázky:

```
\lpcode\tensl‘\V=70 \rpcode\tensl‘\f=-120 \lpcode\tensl‘\j=-40
\hbox{\vbox{\hsize=12pc\noindent\tensl
          Mňau mrňau ..... vysazeno normálně.}%
     \qquad
     \vbox{\hsize=12pc\noindent\tensl\pdfprotrudechars=1
          Mňau mrňau ... vysazeno s korekcí.}%
    }
```

Zvláštním případem popsané techniky je tzv. visící interpunkce, spočívající v tom, že posun krajních znaků bloku se užije jen na interpunkční znaménka, a sice tak, že se posunou vně bloku právě o svoji šířku. K měření velikosti posunu – čísla *přesah* – můžeme užít následující makro `\charwidth`. Poznamenejme, že dimenze `\fontdimen6` *font* má velikost 1 em v písmu *font*.

```
\newcount\overlap  % makro \charwidth do tohoto registru dosadí
                   % šířku znaku v promilech jednotky em
\def\charwidth#1#2{% prvni parametr: přepínač fontu
                   % druhý parametr: znak
  \setbox0=\hbox{#1#2}%
  \overlap=\wd0
  \multiply\overlap by 1000
  \divide\overlap by \fontdimen6#1}
```

Visící interpunkci vidíte ve sloupci (4) na začátku této kapitoly. Ukázka byla vytvořena s nastavením `\protrudechars=1` a s následující specifikací `rpcode` a `lpcode`:

```
\charwidth{\tenrm}{,}\rpcode\tenrm‘\,=\overlap       % čárka
\charwidth{\tenrm}{.}\rpcode\tenrm‘\.=\overlap       % tečka
\charwidth{\tenrm}{\char\hyphenchar\tenrm}%
  \rpcode\tenrm\hyphenchar\tenrm=\overlap             % rozdělovník
\charwidth{\tenrm}{\clqq}\lpcode\tenrm\clqq=\overlap % otevírací uvozovky
```

V ukázce je vidět, že visící interpunkce nemusí působit dojmem dokonalého zarovnání hran bloku. (Jistě je ji však možno chápat jako určitý výtvarný prvek, zejména užívá-li se s většími písmy.)

Hodnoty `\rpcode` a `\lpcode` leží v rozmezí −1000 až 1000; pro praxi to určitě stačí. (Pokud bychom některou z hodnot `\rpcode` či `\lpcode` předepsali menší než −1000, změnila by se na −1000; a kdybychom ji předepsali větší než 1000, změnila by se na 1000.) Implicitní hodnoty `\rpcode` a `\lpcode` jsou nulové.

Mějme nějaký `\vbox` nebo `\vtop`, v němž je zapsán odstavec textu. Žádný z TEXových rozměrů tohoto boxu se nezmění, jestliže na text uplatníme techniku přesahů znaků na okrajích textu při `\pdfprotrudechars=1` (byť by některé znaky přesáhly plochu vymezenou těmi rozměry). Jestliže tuto techniku uplatníme při `\pdfprotrudechars=2`, zůstává šířka boxu; jeho výška či hloubka se ovšem v tomto případě změnit může, jelikož se mohou provést odlišné řádkové zlomy.

Jestliže část textu vložíme do `\hbox`u, pak v případě, že některý z krajních znaků obsahu tohoto boxu se vyskytne na kraji řádku, technika přesahů se na něj uplatní či neuplatní stejně, jako kdyby v boxu zapouzdřen nebyl. Avšak užití boxu může ovlivnit řádkové zlomy.

V závěru této kapitoly se budeme zabývat přesahy znaků na okrajích textu při zvláštních případech sazby.

Při sazbě *do bloku* je odstavcová zarážka nulová (nastavení: `\parindent=0pt` nebo `\noindent`) a východový (tj. poslední) řádek odstavce má plnou délku (nastavení: `\parfillskip=0pt`). Je nutno zodpovědět otázku, zda se uvažovaná technika přesahů uplatní také na první i poslední znak odstavce. Ano, opravdu tomu tak je!

Techniku posunu krajních znaků můžeme použít i při sazbě *na praporek*, tj. při sazbě s jedním okrajem zarovnaným, druhým „otrhaným“. Posun znaků využijeme jen na rovném okraji: v případě pravého „otrhaného“ okraje předepíšeme jen hodnoty `\lpcode`, v případě levého „otrhaného“ okraje jen hodnoty `\rpcode`. Na „otrhaném“ okraji posun znaků nemá význam (a stejně by se neprovedl). Poznamenejme, že s posouvaným znakem se vlastně posune celá řádka, protože mezislovní mezery zde nejsou pružné.

Sazbu na pravý praporek („otrhaný“ pravý okraj) nám zprostředkuje makro `\raggedright` ([1], str. 424), pro sazbu na levý praporek („otrhaný“ levý okraj) si nadefinujeme obdobné makro `\raggedleft`:

```
\def\raggedleft{% otrhaný levý okraj
  \leftskip=0pt plus2em
  \parfillskip=0pt
  \spaceskip=.3333em
  \xspaceskip=.5em
  \relax}
```

Poznamenejme ještě, že posun krajních znaků lze užít i na odstavec, který je *tvarován* příkazem `\parshape` ([1], str. 235).

Intermezzo: PDF versus DVI

Primitivy pdfTEXu, které nejsou obsaženy v TEXu, lze rozdělit do čtyř tříd:

1. Hlavní přepínač `\pdfoutput`. Byl popsán v úvodu tohoto dokumentu. Má zásadní význam, protože určuje, zda výstup pdfTEXu bude ve formátu PDF (`\pdfoutput=1`) nebo DVI (`\pdfoutput=0`).
2. Primitivy, které jsou při nastavení `\pdfoutput=0` pdfTEXem ignorovány. Jsou to všechny primitivy uvedené v kap. 1.
3. Primitivy, které v pdfTEXu fungují i při nastavení `\pdfoutput=0`. Jsou to primitivy uvedené v kap. 3.
4. Primitivy, které při nastavení `\pdfoutput=0` pdfTEX nezná, a tedy při jejich výskytu hlásí chybu. Jsou to všechny primitivy popsané v kap. 2 vyjma `\pdfpageresources`, který náleží do třídy 2. Rovněž primitivy, s nimiž se seznámíte v kapitolách následujících do této třídy patří – kromě `\pdfpagesattr` a `\pdfpageattr` ze sekce 8.2, které náležejí do třídy 2, a `\pdftexversion` a `\pdftexrevision` z konce kap. 9, které jsou z třídy 3.

V TEXovém dokumentu tedy můžete užívat triky se zarovnáním okrajů bloku tisku, které byly popsány v kap. 3: na začátek zdrojového textu přidáte příkaz `\pdfoutput=0` a následně jej zpracujete pdfTEXem.

4. Vkládání externích objektů

V dokumentaci PDF se termínem externí objekt (*XObject*) míní – zhruba řečeno – grafický objekt, jehož obsah je nezávislý na prostředí, ve kterém je použit. Program pdfTEX může pracovat se dvěma typy externích objektů: jsou to formy (*form XObjects*) a obrázky (*image XObjects*). Poznamenejme, že externí objekt použitý pdfTEXem nemusí být nutně vytvořen externím programem.

4.1. Formy

Obsah formy odpovídá obsahu TEXového boxu (ten může obsahovat i obrázky a odkazy na jiné formy). V nejjednodušším případě se forma deklaruje příkazem

`\pdfxform` *číslo_boxu*

což znamená, že

1) obsah registru typu „box“, jemuž je přiřazeno *číslo_boxu*, se bere jako obsah formy,
2) tento obsah se přesune do paměti (a registr se vyprázdní),
3) této formě se přiřadí jisté referenční číslo a to se uloží do registru `\pdflastxform`.

Zapíšeme-li nyní

```
\pdfrefxform\pdflastxform
```

forma se vloží do textu. Při vytvoření další formy příkazem `\pdfxform` je původní hodnota `\pdflastxform` přepsána hodnotou odpovídající této nové formě.

Obecněji je forma deklarována konstrukcí

```
\pdfxform
  transformace
  číslo_boxu
```

kde *transformace* může mít tvar

`attr{/BBox [`x_{ld} y_{ld} x_{ph} y_{ph}`] /Matrix [`a b c d e f`]}`

a má potom následující význam: `/BBox [`x_{ld} y_{ld} x_{ph} y_{ph}`]` vyřízne (a použije) z boxu část tvaru obdélníka, jehož levý dolní roh má souřadnice $x_{\mathrm{ld}}, y_{\mathrm{ld}}$ a pravý horní roh souřadnice $x_{\mathrm{ph}}, y_{\mathrm{ph}}$, a potom `/Matrix [`$a$ b c d e f`]` provede afinní transformaci – tutéž jako „a b c d e f `cm`“ v kontextu sekce 2.3. Souřadnice jsou celá čísla a udávají počet postscriptových bodů (bp).[3] Pokud bezprostředně před `\pdfxform` ... uvedeme `\immediate`, forma se vytvoří hned, jinak až při vytváření stránky.

Příklad:

```
\setbox0\vbox{\hsize8.75cm\noindent Nyní se potěšíme ukázkou
kouzel s formou. Napřed {\tt\char92vbox} s tímto textem
uložíme do registru typu \uv{box}, a ten použijeme k vytvoření
formy. Přitom předepíšeme transformaci, která obsah boxu
trochu roztáhne do šířky a zkosí.}
\pdfxform attr{/Matrix [1.2 0.25 0.3 1 0 0]} 0
\pdfrefxform\pdflastxform
```

Výstup:

Nyní se potěšíme ukázkou kouzel s formou. Napřed `\vbox` s tímto textem uložíme do registru typu „box“, a ten použijeme k vytvoření formy. Přitom předepíšeme transformaci, která obsah boxu trochu roztáhne do šířky a zkosí.

[3] V Adobe Readeru 7.0 se mi podařilo zobrazit formy pouze tak, jako by transformace nebyly předepsány.

Pamatujeme na to, že jako rozměry formy se berou rozměry boxu vstupujícího do její specifikace, bez ohledu na užité transformace. Bývá proto potřeba se postarat o to, aby vytištěná forma byla správně umístěna v textu, který ji obklopuje.

Referenční číslo `\pdfxlastxform` lze uchovat v numerickém registru a později použít, třeba i opakovaně, jak nás navádí následující schéma:

```
\newcount\cislo % pojmenování numerického registru
\immediate\pdfxform... % deklarace formy
\cislo\pdflastxform % uložení jejího referenčního čísla
.........
\pdfrefxform\cislo % a teprve teď se forma vytiskne
.........
\pdfrefxform\cislo % a nyní ještě jednou
.........
\pdfrefxform\cislo % a konečně do třetice
```

Opakovaný tisk obsahu formy tímto postupem šetří objem dokumentu, protože forma je uložena jen jednou. To má význam např. při tisku firemního loga na každé stránce dokumentu.

4.2. Obrázky

Obrázky, které vkládáme do TEXového dokumentu, bývají nejčastěji ve formátu EPS. K jejich zařazení se po překladu zdrojového textu TEXem použije program DVIPS. Výsledný dokument je pak ve formátu PostScript. V pdfTEXu s výstupem do PDF takový postup není možný, a obrázky EPS přímo vkládat nelze. Musíme tedy některým konverzním programem napřed obrázek převést z EPS do formátu, s nímž si pdfTEX poradí: PDF, PNG nebo JPEG. O obrázcích & pdfTEXu pojednávají zajímavé články [3] a [4].

4.2.1. Obrázky ve formátech PNG a JPEG. Pro vložení obrázku v některém z těchto grafických formátů slouží následující konstrukce:

```
\pdfximage
  width dimenze height dimenze depth dimenze
  {jméno.extenze}
\pdfrefximage\pdflastximage
```

Příkaz `\pdfximage` načte obrázek ze souboru *jméno*`.`*extenze*, ale nezpůsobí jeho výstup; obrázku se přiřadí jakési referenční číslo `\pdflastximage` a teprve příkaz `\pdfrefximage\pdflastximage` způsobí výstup obrázku s tímto referenčním číslem. Při dalším příkazu `\pdfximage` ... `{` ... `}` se vytvoří jiné číslo `\pdflastximage` atd. Je-li bezprostředně před `\pdfximage` uvedena sekvence

`\immediate`, obrázek se načítá hned, jinak až při vytváření stránky. Grafickému formátu natahovaného obrázku odpovídá *extenze* souboru: `.png` nebo `.jpg`. Dimenze `width` určuje šířku obrázku, součet dimenzí `height` a `depth` jeho výšku. Jestliže jeden z rozměrů obrázku nelze takto určit, protože chybí buď specifikace dimenze `width` anebo obě specifikace `height` a `depth`, pak se tento rozměr vypočte tak, aby poměr šířky ku výšce obrázku zůstal zachován. Chybějí-li dimenze `width` a `height`, použijí se původní rozměry obrázku (i kdyby byla uvedena dimenze `depth`). Jestliže jsou určeny oba dva rozměry obrázku, v obecném případě se obrázek zdeformuje (v jednom směru bude roztaženější než ve druhém). Dimenze `depth` určuje vertikální polohu obrázku vzhledem k účaří (stejně jako je tomu u boxů). Chybí-li dimenze `depth`, bere se jako nulová:

`height3.5cm`

`height1.5cm depth2cm`

`height0cm depth3.5cm`

Některé z dimenzí `width`, `height` a `depth` mohou být záporné:

- jestliže `width` *dimenze* < 0 a (`height` *dimenze* + `depth` *dimenze*) > 0, dostaneme stranově převrácený obrázek, který bude umístěn od bodu sazby doleva
- jestliže `width` *dimenze* > 0 a (`height` *dimenze* + `depth` *dimenze*) < 0, dostaneme obrázek převrácený ve svislém směru okolo účaří
- jestliže `width` *dimenze* < 0 i (`height` *dimenze* + `depth` *dimenze*) < 0, dostaneme obrázek otočený o 180° okolo bodu sazby
- jestliže `width` *dimenze* < 0 a dimenze `height` není uvedena, dostaneme obrázek otočený o 180° okolo bodu sazby, přičemž vertikální rozměr obrázku se dopočte tak, že poměr stran obrázku zůstává zachován

Pokud soubor s natahovaným obrázkem je v jiném adresáři než zdrojový text dokumentu, musíme v příkazu `\pdfximage` jeho jméno uvést s cestou – relativní nebo absolutní, přičemž jako oddělovače adresářů užíváme lomítka. Např.

```
\pdfximage width5cm {../obr.jpg}
```

nebo

```
\pdfximage width5cm {C:/usr/obr.jpg}
```

Potřebujeme-li zapsat rozměry původního obrázku do souboru `.log`, postupujeme takto:

```
\newbox\obr
\setbox\obr\hbox{\pdfximage {obrazek.pdf}\pdfrefximage\pdflastximage}
\message{^^JSirka obrazku: \the\wd\obr ^^JVyska obrazku: \the\ht\obr ^^J}
```

Při opakovaném tisku obrázku postupujeme obdobně jako při opakovaném tisku formy.

Potřebujeme-li obrázek transformovat, uložíme jej do registru typu „box“, a ten použijeme k vytvoření formy, na kterou již můžeme aplikovat transformace. Např. takto:

```
\setbox0=\hbox{\pdfximage width5cm {obrazek.jpg}
               \pdfrefximage\pdflastximage}
\pdfxform attr{/Matrix [-1 0 0 1 0 0]} 0
\pdfrefxform\pdflastxform
```

Příklad. V další ukázce opravíme obrázek pobřežní krajiny tak, aby horizont byl vskutku horizontální. Obrázek při natažení do formy vhodně pootočíme a následně ořežeme, aby jeho okraje byly rovnoběžné s okraji stránky:

```
\newcount\prvni \newcount\druhy \newcount\treti
\pdfximage width0.3\hsize{krajina.jpg}
\prvni=\pdflastximage
\setbox0=\hbox{\pdfrefximage\prvni}
\pdfxform
  attr{/Matrix [0.99939 0.034899 -0.034899 0.99939 0 0]} 0
  % obrázek je otočen o dva stupně
\druhy=\pdflastxform
\setbox0=\hbox{\pdfrefxform\druhy}
\pdfxform attr {/BBox [0 5 100 78]} 0
  % a nyní je oříznut
\treti=\pdflastxform
  % tisk jednotlivých fází úpravy obrázku:
\centerline{\pdfrefximage\prvni \hfill
            \pdfrefxform\druhy \hfill
            \pdfrefxform\treti}
```

Výstup:

původní obrázek

pootočený

a oříznutý

4.2.2. Obrázky ve formátu PDF. Vše, co bylo v odst. 4.2.1 řečeno o vkládání obrázků ve formátech PNG a JPEG, platí i pro vkládání obrázků ve formátu PDF. Navíc však můžeme předepsat transformaci obrázku. Obrázek může být částí vícestránkového dokumentu ve formátu PDF. V takovém případě můžeme předepsat, která stránka se má natáhnout.

```
\pdfximage
  width dimenze height dimenze depth dimenze
  transformace
  page stránka
  {jméno.pdf}
```

Popis položky *transformace* je týž jako v sekci 4.1. Číslo *stránka* určuje pořadí natahované stránky v dokumentu *jméno*`.pdf`; chybí-li tato specifikace, natáhne se první stránka. Položka *soubor* má stejnou syntaxi jako v odst. 4.2.1, pouze extenze musí být `.pdf`. Pro zařazení obrázku opět použijeme konstrukci

```
\pdfrefximage\pdflastximage
```

Počet stran dokumentu, z něhož je obrázek natahován, je dostupný v registru `\pdflastximagepages`.

Záporné dimenze `width`, `height` a/nebo `depth` působí efekty převrácení resp. otočení obrázku, tak, jak to bylo popsáno v odst. 4.2.1. Zde se tyto transformace skládají s transformacemi určenými položkou *transformace*.

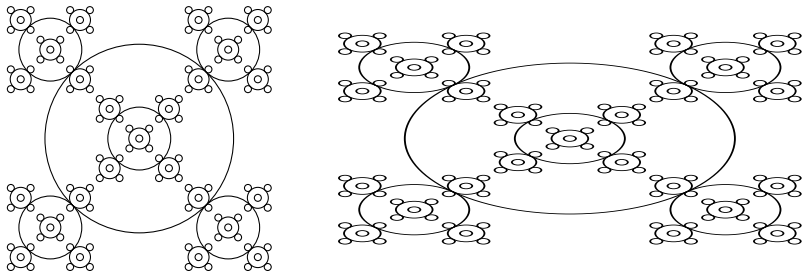
4.2.3. Obrázky vytvořené METAPOSTem. Program METAPOST produkuje obrázky ve formátu EPS, který v tomto případě užívá omezených prostředků. Program pdfTEX dokáže takové obrázky vložit do dokumentu pomocí konverzního programu. K tomu potřebujeme následující soubory z distribuce ConTEXtu: `supp-mis.tex` a `supp-pdf.tex`, a postupujeme takto:

```
\input supp-pdf.tex % mimo jiné natáhne soubor supp-mis.tex
.........
\convertMPtoPDF{jméno.extenze}{a}{b}
```

kde a je zvětšení ve vodorovném směru a b zvětšení ve směru svislém. Oba parametry a & b musí být uvedeny.

Jestliže $a < 0$ a $b > 0$, obrázek se horizontálně překlopí kolem levé strany ohraničujícího boxu (*bounding box*). Jestliže $a > 0$ a $b < 0$, obrázek se vertikálně překlopí kolem spodní strany ohraničujícího boxu. Jestliže $a < 0$ i $b < 0$, obrázek se otočí o 180° kolem levého dolního rohu ohraničujícího boxu. Je to analogické efektům způsobeným záporností (některých z) dimenzí vystupujících v příkazu `\pdfximage`, jak bylo zmíněno v odst. 4.2.1.

Příklad. Po natažení souboru `supp-pdf.tex` byl vytištěn obrázek `rekurze.0`, který byl vygenerován METAPOSTem; vlevo byl vytištěn v původní velikosti příkazem `\convertMPtoPDF{rekurze.0}{1}{1}` a vpravo s deformovaným poměrem šířky ku výšce příkazem `\convertMPtoPDF{rekurze.0}{1.75}{0.8}`.



Čtenář, který se trochu vyzná v syntaxi jazyka METAPOSTu, se možná rád podívá na zdrojový text obrázku; uvidí tam zajímavou ukázku rekurzivního volání makra:

```
def kruznice (expr a, b, c)=
    % a ... střed  kružnice
    % b ... průměr kružnice
    % c ... počet úrovní rekurzí
  pickup pencircle scaled 0.35;
    % nakreslíme kružnici:
  draw fullcircle scaled b shifted a;
    % následuje rekurzivní volání makra pro nakreslení pěti dalších kružnic
    % s třetinovým průměrem, které jsou umístěny vlevo dole, vpravo dole,
    % vpravo nahoře a vlevo nahoře (smyčka for--endfor) od kružnice
    % z předchozí úrovně rekurze a uprostřed ní (příkaz za endfor)
  if c>0:
    for k=(-1,-1),(1,-1),(1,1),(-1,1):
      kruznice (a+1.4142b/3*k, b/3, c-1);
    endfor
    kruznice (a, b/3, c-1);
  fi;
enddef;
%
beginfig(0);
  kruznice ((50,50), 70.71, 3);    % 70.71 = 50 * sqrt(2)
endfig;
%
bye;
```

Obrázek můžeme uložit jako formu a tu užít k opakovanému tisku obrázku:

```
\newbox\mpbox
\newcount\mpref
\setbox\mpbox=\hbox{\convertMPtoPDF{obrazek.12}{1}{1}}
\pdfxform\mpbox
\mpref=\pdflastxform
......
\pdfrefxform\mpref
......
\pdfrefxform\mpref
```

4.3. „Vodotisk“

Nebude se jednat o opravdovou průsvitku, proto ty uvozovky v nadpisu. Oč půjde? O to, aby každá strana dokumentu byla potištěna jedním a týmž grafickým objektem (např. nevtíravým bledým obrázkem) a přes něj byl proveden tisk vlastního dokumentu. K tomuto účelu upravíme výstupní rutinu. Pro ušetření objemu výsledného souboru `.pdf` bude pochopitelně vhodné „vodotisk“ uložit do formy a tu pak opakovaně vyvolávat.

Následuje návrh úpravy výstupní rutiny tak, aby pozadí každé strany tvořil obrázek *jméno*`.`*extenze* vystředěný v ploše tisku.

```
\newcount\refwatermark
\setbox0 =
  \vbox to \vsize{% \vbox o rozměrech tiskové plochy ...
    \vfil
    \hbox to \hsize{% ... a s vystředěným obrázkem:
      \pdfximage width 4cm height 4cm {jméno.extenze}%
      \hfil \pdfrefximage\pdflastximage \hfil}%
    \vfil}
\pdfxform 0 % obsahem formy je onen \vbox
\refwatermark\pdflastxform
\def\BG{% definujeme pozadí jakožto obsah formy
  \vbox to 0pt{\hbox to 0pt{%
    \pdfrefxform\refwatermark\hss}\vss}}
\def\plainoutput{%
  \shipout\vbox{%
    \BG \makeheadline \pagebody \makefootline}%
    \advancepageno
    \ifnum\outputpenalty>-20000 \else\dosupereject\fi}
```

Poznamenejme, že tato sekce je tématicky úzce spjata se sekcí 2.4.

5. Zřetězení článků

Existuje možnost zvolit režim prohlížení dokumentu, při němž jsou za sebou řazeny vybrané objekty – odstavce textu nebo obrázky. Ty přitom nemusí následovat bezprostředně za sebou v dokumentu. Lze tak např. umožnit prohlížení všech ilustrativních příkladů nebo prohlížení všech obrázků apod. Takové seřazení se nazývá zřetězením (*thread*). V dokumentu můžeme definovat několikero zřetězení. Souhrn všech objektů, náležejících do některého zřetězení, se nazývá článek (*article*). Dva možné způsoby sestavení zřetězení budou popsány v sekcích 5.1 a 5.2.

Jestliže se kurzor dostane nad některý objekt náležející do zřetězení (nebo nad plochu, která je objektu přiřazena – upřesníme později), změní se tvar kurzoru, a pokud klepneme myší, objekt se zobrazí v takové velikosti, aby jeho šířka odpovídala šířce okna (případně s dodatečnými okraji – upřesníme později). Dalšími klepnutími myší se postupně zobrazují jednotlivé části zřetězení, dokud z něj nevyjdeme ven. Klepnutí při stisknutém tlačítku Shift provádí totéž v opačném směru.

Nalistujte si příklad z konce kap. 3, týkající se sazby úzkých odstavců šikmým písmem; zajeďte myší nad první z odstavečků (všimněte si přitom změny tvaru kurzoru) a klepněte. Odstaveček se zvětší. Pak klepněte ještě jednou. Tím přejdete na druhý odstaveček (poznáte to podle posledního řádku a podle odlišného zarovnání pravého okraje). A ještě klepněte, a dostanete se do normálního režimu prohlížení a nastaví se původní velikost zobrazení.[4]

5.1. Jednoduché značky

Každý z objektů, které mají být součástí zřetězení, „obalíme“ následujícím způsobem:

```
\vbox{\pdfthread
        width dimenze height dimenze depth dimenze
        informace
        označení
        objekt_zřetězení}
```

Jakožto *označení* uvedeme jednu z položek

```
num přirozené_číslo
name {identifikátor}
```

Přitom všechny objekty náležející do jednoho článku musí mít totéž *označení*. Naopak, různá zřetězení vybraná z téhož dokumentu musí mít *označení* různá.

[4] Při užití Adobe Readeru 7.0 jsem se tímto způsobem nedokázal po výstupu ze zřetězení vrátit k původní velikosti zobrazení, zatímco Acrobat Reader 5.0 fungoval, jak bylo popsáno.

Termín *identifikátor* je buď „nic“, nebo posloupnost znaků, jimiž mohou být písmena (s rozlišením malých a velkých) včetně písmen s diakritickými znaménky, číslice, mezery (přičemž několik mezer za sebou má týž význam jako mezera jedna), TEXové řídicí sekvence (kuriozita, že ano?), a s výjimkou znaku `%` i další znaky z klávesnice (kulaté závorky a rovněž složené závorky musí být ale párovány). Poznamenejme, že „`num 5`“ je něco jiného než „`name {5}`“.

– Nejprve předpokládejme, že (nepovinná) specifikace rozměrů `width`, `height`, `depth` není uvedena. Pak se v prohlížeči změní kurzor (objeví se v něm šipka dolů), jakmile se dostane nad *objekt_zřetězení*. Jestliže klepneme tlačítkem myši, dostaneme se do zmíněného režimu prohlížení zřetězení, přičemž se nastaví takové zvětšení, aby šířka (tohoto zvětšeného) objektu byla rovna šířce okna.
– Jestliže uvedené rozměry jsou specifikovány, přepíší rozměry `\vbox`u a tím se odpovídajícím způsobem předefinuje plocha, nad níž se mění tvar kurzoru, a zvětšení se nastaví tak, aby specifikovaná šířka `width` se zobrazila na celou šířku okna. Není-li některý rozměr specifikován, použije se samozřejmě rozměr původní.

Nepovinná položka *informace* má tvar

```
attr{/I << /Author (jméno_autora)
           /Subject (předmět_článku)
           /Title (název_článku)
           /Keywords (klíčová_slova)
        >>}
```

a stačí ji uvést jen u jednoho objektu zřetězení. Význam je patrný. V textových řetězcích se vyhneme diakritice.

Seznam všech článků prohlíženého dokumentu můžeme vidět ve zvláštním rámečku nebo postranním okénku prohlížeče; vyvolání z menu:

Okno → Články nebo Window → Articles

V tomto rámečku či okénku jsou jako názvy článků uvedeny řetězce specifikované v `/Title(...)`. Další hesla z položky *informace* si tam můžeme vyvolat.

Jestliže je kdesi na počátku zdrojového textu uveden příkaz

`\pdfthreadmargin=`*dimenze*

pak se v režimu prohlížení zřetězení přidají dodatečné okraje o velikosti tohoto rozměru. (Vhodná je hodnota *dimenze* 5–10 pt.)

Tip: Aby se mezi `\vbox`em obsahujícím text a textem, který mu předchází, nenarušilo řádkování, vložíme na začátek `\vbox`u podpěru `\strut`. Tuto podpěru pro tento účel vhodně předefinujeme, pokud neužíváme písmo v základní velikosti, nebo pokud hodnota `\parskip` není nulová.

Pokud zapíšeme konstrukci

```
\pdfthread
  width dimenze height dimenze depth dimenze
  informace
  označení
  objekt_zřetězení
```

dovnitř boxu, jímž v této souvislosti rozumíme nejen `\hbox`, `\vbox` a jeho odvozeniny `\vtop` a `\vcenter`, ale také matematické prostředí `$$` ... `$$`, pak objektem zřetězení je materiál od místa konstrukce `\pdfthread` ... až do konce boxu nejnižší úrovně, který tuto konstrukci obsahuje. Mějme např. schéma

```
\vbox{A B \hbox{C D} E F}
```

v němž *A*–*F* značí části textu, a na některé jeho místo vložíme příkaz `\pdfthread name{1}`. U každé z následujících tří ukázek uvádíme výstup, v němž podtržením je vyznačen příslušný objekt zřetězení:

```
\vbox{A \pdfthread name{1}B \hbox{C D} E F}
```
A <u>*B C D E F*</u>

```
\vbox{A B \hbox{C \pdfthread name{1}D} E F}
```
A B C <u>*D*</u> *E F*

```
\vbox{A B \hbox{C D} \pdfthread name{1}E F}
```
A B C D <u>*E F*</u>

Příklad. Popis sazby dvou úzkých odstavečků psaných šikmým písmem v příkladě na konci kap. 3 nebyl přesný; ve skutečnosti tyto dvě části textu byly zřetězeny zařazením podtržených příkazů:

```
\pdfthreadmargin=10pt % přidaný okraj
\rpcode\tensl`\f=-100 \lpcode\tensl`\j=-40
\hbox{\vbox{\pdfthread num 1 \hsize=12pc\noindent\tensl
            Mňau mrňau ..... vysazeno normálně.}%
      \qquad
      \vbox{\pdfthread num 1 \hsize=12pc\noindent\tensl
                \pdfprotrudechars=1
            Mňau mrňau ..... vysazeno s korekcí.}%
    }
```

5.2. Párové značky

Druhá možnost specifikace objektů zřetězení je užitím dvou značek, počáteční

```
\pdfstartthread
  informace
  označení
```

a koncové

```
\pdfendthread
```

Položky *informace* a *označení* mají týž význam jako v předchozím odst. 5.1.

Způsob použití: Všechny objekty zřetězení jsou vertikální boxy (tj. boxy typu `\vbox`, `\vtop`, popř. `\vcenter`). První z nich bude mít tvar

```
\vbox{%                  \vtop{%                  \vcenter{%
  \pdfstartthread          \pdfstartthread          \pdfstartthread
  informace        nebo    informace        nebo    informace
  označení                 označení                 označení
  tiskový_materiál }       tiskový_materiál }       tiskový_materiál }
```

a poslední z nich tvar

```
\vbox{%                  \vtop{%                  \vcenter{%
  tiskový_materiál         tiskový_materiál         tiskový_materiál
  \par             nebo    \par             nebo    \par
  \pdfendthread}           \pdfendthread}           \pdfendthread}
```

Do zřetězení se pak automaticky zařadí všechny vertikální boxy, které leží mezi těmito dvěma.[5] Ale co uděláme, jestliže tam je také nějaký vertikální box, který do zřetězení nemá být zařazen? Vnoříme jej do `\hbox`u, tzn. užijeme konstrukci `\hbox{\vbox{`*tiskový_materiál*`}}`. Omezení: Vertikální box s `\pdfstartthread` nemůže být vnořen do boxu (vertikálního, horizontálního nebo do matematického módu `$$`...`$$`), který neobsahuje `\vbox` s `\pdfendthread`, a obráceně.

6. Anotace: text, zvuk, video

Anotacemi v dokumentu PDF rozumíme doplňky k dokumentu, jež se vážou k určitým místům v něm. Mohou to být textové poznámky, zvukové nahrávky, videoklipy nebo hypertextové odkazy. Anotace se zpravidla mohou nalézat ve dvou stavech: uzavřeném anebo otevřeném. Uzavřená anotace je představována např. ikonou nebo aktivní plochou; klepnutím, popř. poklepáním myší se stane anotace otevřenou, tzn. podle svého typu se projeví: zobrazí se okénko s textovou poznámkou, spustí se zvuková nahrávka, přehraje se videoklip nebo se přejde na cíl hypertextového odkazu. Tato kapitola bude věnována zmíněným typům anotací s výjimkou hypertextových odkazů, které budou probírány v rámci kapitoly 7.

6.1. Textové anotace

Doplňujícím poznámkám v dokumentu lze dát formu textových anotací. V uzavřeném stavu jsou indikovány ikonami. Po poklepání na ikonu se zobrazí

[5] Má zkušenost: Do zřetězení se zařadily pouze takové z uvažovaných vertikálních boxů, jejichž obsah se při výstupu objevil na téže stránce dokumentu jako obsah vertikálního boxu zahrnujícího počáteční značku zřetězení.

okénko s textem poznámky. Pomocí myši můžete jak ikonu, tak okénko anotace posouvat, a u okénka můžete též měnit jeho rozměry. Uzavřením okénka se anotace opět „schová“ do ikony. Při změně zvětšení dokumentu ikona velikost nemění. Pozor na umístění anotace – dbejte na to, aby ikona nepřekrývala text dokumentu. Při tisku dokumentu se anotace neobjeví.

Ve zdrojovém textu dokumentu se textová anotace specifikuje následující konstrukcí:

```
\pdfannot
  width dimenze height dimenze depth dimenze
  {/Subtype /Text
   /Contents ( obsah )
   /T ( nadpis )
   /C [R G B]
   /Open logická_hodnota
  }
```

Rozměry okénka anotace jsou určeny pomocí TEXových dimenzí `width` ... `height` ... `depth` ... (uvádějte všechny tři). V položce *obsah* je zapsán text anotace, význam nepovinné položky *nadpis* je jasný, a konečně `/C [...]` definuje barvu ikony anotace a pozadí nadpisu pomocí barevných složek v systému RGB, pokud se nespokojíme s implicitní barvou. Dále, je-li uvedeno `/Open true`, je prvotní stav anotace otevřený; je-li tam `/Open false`, nebo chybí-li klíč `/Open` a jeho hodnota, pak prvotní stav anotace je nastaven jako uzavřený.

Poznámka. Texty v anotacích mohou obsahovat písmena, číslice, mezery (několik mezer za sebou má týž význam jako mezera jedna), a s výjimkou znaků `\ ~ %` i další znaky z klávesnice; přitom kulaté závorky a též složené závorky musí být párovány, a znak `#` zapisujeme jako `\#`. Texty v anotacích jsou sázeny fontem určeným konfigurací prohlížeče. Proto tento font nemusí ladit se vzhledem sazby vlastního dokumentu. Abychom se vyhnuli problémům s kódováním diakritiky, je rozumné psát texty anotací „bez hacku, carek a krouzku“. V zápisu textu anotace nelze používat TEXové řídicí sekvence ani matematický mód.

Příklad: Zde je ukázka textové anotace: A takto byla vytvořena:

```
\pdfannot
  width 10cm height 0.5cm depth 1.5cm
  {/Subtype /Text
   /Contents (Zde vidite, jak vypada textova anotace.
              Vyzkousejte si, ze okenko lze posouvat
              a ze lze menit jeho velikost.)
   /T (Ukazka textove anotace)
  }
```

6.2. Zvuk a video jakožto anotace

Video (*movie*) anotace umožňují přehrávat nejen video nahrávky `.mov`, ale také zvukové nahrávky `.wav`. Při práci na zmíněných platformách a při užití níže popsaného postupu se kódy těchto nahrávek do PDF dokumentu nenatahují, ale z dokumentu se na příslušné soubory odkazuje. Video anotace v uzavřeném stavu je představována aktivní plochou. Kurzor myši nad touto plochou změní tvar. Po klepnutí se anotace otevře, což znamená, že se spustí odpovídající nahrávka. Klávesou Esc ji můžeme předčasně ukončit. Základní konstrukce video anotace:

```
\pdfannot
  width dimenze height dimenze depth dimenze
  {/Subtype /Movie
   /Movie << /F ( soubor ) >>
  }
```

Řádka `width` ... `height` ... `depth` ... vymezuje aktivní plochu anotace. Uvádějte všechny tři rozměry. Jestliže soubor s nahrávkou se nachází v témže adresáři jako dokument, má *soubor* syntaxi *jméno*`.`*extenze* (přičemž *extenze* je zde `.wav` nebo `.mov`). Pokud je však tento soubor v jiném adresáři, je třeba uvést i cestu k němu; v tomto případě má tedy *soubor* tvar *cesta*`/`*jméno*`.`*extenze*. Můžeme předepsat jak absolutní cestu, tak i cestu relativní vzhledem k umístění dokumentu. Jako oddělovače adresářů užijeme lomítka. Před označením disku rovněž píšeme lomítko, dvojtečku vynecháváme.

Příklad. Mějme v adresáři `C:\DOC\PDF\` dokument `vylet.pdf`, který obsahuje video anotaci odkazující na nahrávku `pes.mov`. V následující tabulce vidíme, jak zapisujeme soubor `pes.mov` s cestou, v závislosti na tom, kde na disku se nachází. Relativní cesta je míněna vzhledem k umístění souboru `vylet.pdf`.

Umístění souboru `pes.mov` *na disku*	Zápis souboru s absolutní cestou	Zápis souboru s relativní cestou
`C:\DOC\PDF\`	`/F (/C/DOC/PDF/pes.mov)`	`/F (pes.mov)`
`C:\DOC\PDF\ANOT\`	`/F (/C/DOC/PDF/ANOT/pes.mov)`	`/F (ANOT/pes.mov)`
`C:\DOC\VIDEO\MOV\`	`/F (/C/DOC/VIDEO/MOV/pes.mov)`	`/F (../VIDEO/MOV/pes.mov)`

Poznamenejme, že video anotace je bezrozměrnou konstrukcí v tom smyslu, že neposouvá aktuální pozici sazby. Většinou chceme, aby v aktivní ploše anotace byl nějaký text nebo obrázek. Proto je užitečné vytvořit si jednoduché makro `\setannotbox`, které by definovalo příkazy pro specifikaci rozměrů aktivní plochy a pro tisk objektu v ní:

```
\newdimen\annotmargin % nepovinná deklarace dimenze
\annotmargin=dimenze  % nepovinná specifikace okrajů
                      % okolo obsahu aktivní plochy
```

```
\def\setannotbox#1{%
  \ifx\annotmargin\undefined
    \setbox0=\hbox{#1}%
  \else
    \setbox0=\hbox{\kern\annotmargin#1\kern\annotmargin}%
    \dimen1=\ht0 \advance \dimen1 by \annotmargin \ht0=\dimen1
    \dimen2=\dp0 \advance \dimen2 by \annotmargin \dp0=\dimen2
  \fi
  \def\annotboxdim{width \wd0 height \ht0 depth \dp0}%
  \def\annotbox{\box0}}
```

– Použití makra, je-li v aktivní ploše anotace text:

Text před aktivní plochou anotace
`\setannotbox{` *Text v aktivní ploše anotace* `}%`
```
\pdfannot
  \annotboxdim
  {/Subtype /Movie
   /Movie << /F ( soubor ) >>}%
\annotbox
```
Text za aktivní plochou anotace

– Použití makra, je-li v aktivní ploše anotace obrázek:

Text před aktivní plochou anotace
`\immediate\pdfximage width` *dimenze* `{`*jméno.extenze*`}%`
```
\setannotbox{\pdfrefximage\pdflastximage}%
\pdfannot
  \annotboxdim
  {/Subtype /Movie
   /Movie << /F ( soubor ) >>}%
\annotbox
```
Text za aktivní plochou anotace

Konstrukce video anotace může obsahovat další užitečné parametry, čímž se budou zabývat následující dvě podsekce 6.2.1 a 6.2.2. V popisech deklarací těchto anotací nebude bráno v úvahu makro `\setannotbox`, ale čtenář si popisy může snadno upravit pro využití tohoto makra.

Úmluva. V příkladech ve zbytku sekce 6.2 předpokládáme, že máme v dokumentu zavedeno

```
\newdimen\annotmargin
\annotmargin=3pt
\def\setannotbox#1{...}% makro bylo popsáno dříve
```

6.2.1. Zvukové nahrávky. Pro zařazení anotace se zvukovou nahrávkou užijeme konstrukci

```
\pdfannot
  width dimenze height dimenze depth dimenze
  {/Subtype /Movie
   /C [R G B] /Border [0 0 tloušťka]
   /A      << /ShowControls logická_hodnota
              /Mode režim_přehrávání
              /Start [čitatel jmenovatel]
              /Duration [čitatel jmenovatel]
              /Volume hlasitost
           >>
   /Movie << /F (cesta/jméno.wav)
           >>
  }
```

Příkaz `/Border [...]` nakreslí rámeček okolo aktivní plochy, a číslo *tloušťka* udává jeho tloušťku v postscriptových bodech (bp). Chybí-li tento příkaz, rámeček se nenakreslí. Příkazem `/C [...]` je specifikována barva rámečku v systému RGB. Implicitní barva je černá. Rámeček může být proveden i čárkovaně. To se zařídí tím, že do příkazu `/Border [...]` doplníme další dvě číselné hodnoty, které udávají délku čárek a délku mezer (opět v postscriptových bodech) podle vzoru `/Border [0 0` *tloušťka* `[`*čárky mezery*`]]`. Tedy např. zápis `/Border [0 0 2 [5 2.5]]` říká, že rámeček o tloušťce 2 bp bude nakreslen čárkovaně, přičemž se čárky o délce 5 bp budou střídat s mezerami 2,5 bp. Při tisku dokumentu se rámeček neobjeví. Je-li *logická_hodnota* za `/ShowControls` rovna `true`, pod aktivní plochou anotace se zobrazí řídicí lišta přehrávače zvukových záznamů. Pokud specifikujeme *režim_přehrávání* za `/Mode` jako `/Open`, pak tato lišta zůstává i po přehrání, dokud myší neklepneme mimo aktivní plochu. Implicitně je *režim_přehrávání* nastaven jako `/Once`, což znamená, že po přehrání lišta zmizí. Při nastavení `/ShowControls false`, které je implicitní, se řídicí lišta neobjevuje. Klíč `/Start` následovaný dvojicí celých čísel *čitatel* ≥ 0 a *jmenovatel* ≥ 1 říká, že záznam se začne přehrávat v místě vzdáleném od začátku o $\frac{\textit{čitatel}}{\textit{jmenovatel}}$ sekund. Implicitně se přehrává od samého začátku. Klíč `/Duration` následovaný dvojicí celých čísel *čitatel* ≥ 0 a *jmenovatel* ≥ 1 udává délku přehrávání jakožto $\frac{\textit{čitatel}}{\textit{jmenovatel}}$ sekund. Implicitně přehrávání trvá až do konce. Užitím těchto parametrů lze např. nahrávku rozložit na několik částí, každou z nich zařadit do samostatné anotace a ty proložit tištěnými komentáři. Úroveň hlasitosti nastavíme parametrem `/Volume` *hlastitost*, kde *hlasitost* je číslo mezi nulou (= vypnutý zvuk) a jedničkou (= plná hlasitost); větší číslo = větší hlasitost, implicitně nastavena plná hlasitost. Tento parametr může být užitečný tehdy,

potřebujeme-li u několika zvukových anotací nastavit stejnou hladinu hlasitosti, abychom nemuseli znovu a znovu kroutit knoflíkem na reproduktorech.

Příklad:

```
Spusťte nahrávku klepnutím dovnitř rámečku:
\setannotbox{Poslech}%
\pdfannot
  \annotboxdim
  {/Subtype /Movie
   /C [0.8 0 0] /Border [0 0 1]
   /A     << /Duration [6 1] >>
   /Movie << /F (audio.wav) >>
  }%
\annotbox
```

Výstup:

Spusťte nahrávku klepnutím dovnitř rámečku: Poslech

6.2.2. Videoklipy. Jak dále bude popsáno, videoklipy se mohou zobrazit buď v aktivní ploše anotace (po klepnutí na ni), nebo ve zvláštním okénku. V prvním případě se rozměry videoklipu přizpůsobí přesně rozměrům aktivní plochy; abychom se vyhnuli deformaci záběrů, musíme rozměry aktivní plochy zvolit tak, aby byly úměrné rozměrům videa.

```
\pdfannot
  width dimenze height dimenze depth dimenze
  {/Subtype /Movie
   /C [R G B] /Border [0 0 tloušťka]
   /A     << /ShowControls logická_hodnota
             /Mode režim_přehrávání
             /Start [čitatel jmenovatel]
             /Duration [čitatel jmenovatel]
             /Rate tempo
             /Volume hlasitost
             /FWScale [čitatel jmenovatel]
             /FWPosition [x y]
          >>
   /Movie << /F (cesta/jméno.mov)
             /Poster logická_hodnota
          >>
  }
```

Parametry `/C [...]`, `/Border [...]`, `/Showcontrols ...`, `/Start [...]`, `/Duration [...]` a `/Volume ...` mají stejný význam jako v odst. 6.2.1.

Jako *režim_přehrávání* můžeme zvolit jednu z následujících hodnot (klepněte si do rámečků):

`/Once`	.....	přehrává se jen jednou
`/Open`	.....	přehrává se jen jednou, koncový záběr zůstane zobrazen
`/Repeat`	.....	přehrává se opakovaně, dokud přehrávání neukončíme
`/Palindrome`	.....	přehrává se tam a zpátky, tam a zpátky atd., dokud přehrávání neukončíme

Položka *tempo* následující za `/Rate` je reálné číslo s tímto významem:

- směr přehrávání je určen tím, je-li číslo *tempo* kladné či záporné
- rychlost přehrávání je určena absolutní hodnotou čísla *tempo*

Je-li $tempo > 0$, pak přehrávání má normální směr a probíhá od okamžiku, který je určen parametrem `/Start [...]`, a pokud tento chybí, pak od začátku klipu; je-li $tempo < 0$ pak přehrávání má směr obrácený (zpětný) a probíhá od okamžiku určeného parametrem `/Start [...]` (pokud tento chybí, pak se nepřehrává). Přitom $|tempo|$ udává, kolikrát je přehrávání zrychlené, takže

$\|tempo\| = 1$	.....	normální rychlost přehrávání
$\|tempo\| > 1$	.....	zrychlené přehrávání
$\|tempo\| < 1$	.....	zpomalené přehrávání

Parametr `/FWScale [` *čitatel jmenovatel* `]`, kde *čitatel* a *jmenovatel* jsou přirozená čísla, způsobí, že se videoklip zobrazí ve zvláštním okénku, jehož rozměry jsou definovány tak, že skutečné rozměry záběrů videoklipu se vynásobí číslem $\frac{čitatel}{jmenovatel}$; přehrávaný videoklip se přizpůsobí rozměrům okénka. Parametr `/FWPosition [`x y`]`, kde x a y jsou čísla mezi nulou a jedničkou, udává pozici okénka na obrazovce. Číslo x určuje horizontální pozici ($x = 0$ zcela vlevo, $x = 1$ zcela vpravo) a číslo y vertikální pozici ($y = 0$ zcela nahoře, $y = 1$ zcela dole). Implicitně je nastaven parametr `/FWPosition [0.5 0.5]`, což znamená, že okénko je umístěno ve středu obrazovky. Pokud parametr `/FWScale [...]` není uveden, videoklip se zobrazí v aktivní ploše anotace. Jestliže je *logická_hodnota* za `/Poster` rovna `true`, pak v době, kdy anotace je uzavřená, je v její aktivní ploše zobrazen určitý charakteristický záběr videoklipu. Tento jev nenastává, je-li *logická_hodnota* rovna `false`, což je i implicitní nastavení.

Pokud v době přehrávání videoklipu klepneme myší do plochy, kde se zobrazují záběry (ať je to aktivní plocha anotace nebo samostatné okénko), přehrávání se přeruší, a následným poklepáním v téže ploše se pak zase pokračuje.

Příklad:

```
Videoklip se promítne \
\setannotbox{$\vcenter to 6pc{\hsize5.0pc
                        \vfill
                        \centerline{\strut uvnitř}
                        \centerline{\strut tohoto}
                        \centerline{\strut rámečku}
                        \vfill
                       }
             $}%
\pdfannot
  \annotboxdim
    {/Subtype /Movie
     /Border [0 0 2] /C [0.2 0.7 1]
     /Movie << /F (video.mov) >>
    }%
\annotbox
\ \ anebo \
\setannotbox{v okénku}%
\pdfannot
  \annotboxdim
    {/Subtype /Movie
     /Border [0 0 2] /C [0.7 0.3 0.9]
     /A     << /FWScale [3 2] >>
     /Movie << /F (video.mov) >>
    }%
\annotbox
\ \ \dots
```

Výstup:

Videoklip se promítne uvnitř tohoto rámečku anebo v okénku ...

Poznámka. V době přehrávání anotace se zvukem (odst. 6.2.1) zmizí obsah její aktivní plochy. Vysvětlení je prosté, chápeme-li takovouto anotaci jako „videoklip bez videa“ – v aktivní ploše se tedy během přehrávání zobrazí „nic“.

Víme, že zařazením parametru `/Volume 0` do specifikace video anotace vypneme zvuk. Lze naopak přehrát z videoklipu pouze zvuk? Ano, pomocí následujícího špinavého triku:

```
\pdfannot
  width dimenze height dimenze depth dimenze
  {/Subtype /Movie
   /A     << /FWScale [1 číslo] >>
   /Movie << /F (cesta/jméno.mov) >>
  }
```

přičemž *číslo* je přirozené a přiměřeně velké – tak, aby při *číslo*-násobném zmenšení se okénko pro přehrávání videoklipu již nezobrazilo, ale ne zase tak velké, aby poměr $1/$*číslo* byl interpretován jako nula. Je nutno vyzkoušet. Přenositelnost na jiné počítače zřejmě nebude zaručena.

Příklad:

```
Klepnutím dovnitř rámečku spustíte
\setannotbox{neviditelný, ale slyšitelný videoklip}%
\pdfannot
  \annotboxdim
  {/Subtype /Movie
   /Border [0 0 2] /C [0.4 0.5 0.6]
   /A     << /FWScale [1 1000] >>
   /Movie << /F (video.mov) >>}%
  \annotbox
\ .
```

Výstup:

Klepnutím dovnitř rámečku spustíte neviditelný, ale slyšitelný videoklip .

7. Anotace: hypertextové odkazy

Hypertextový odkaz (*hyperlink*) sestává z aktivního odkazu a cíle. Aktivním odkazem bývá obdélníková plocha, zpravidla obsahující text nebo obrázek. Když na plochu klepneme myší, přejde se na cíl: ten se může nacházet přímo v dokumentu – pak se cíl nazývá doskokem – nebo vně dokumentu, např. jako webová stránka. Doskokem je buď specifikované místo v dokumentu anebo určitá stránka dokumentu. V prvním případě je třeba doskok ve zdrojovém textu deklarovat.

7.1. Deklarace doskoku v určitém místě dokumentu

V místě cíle zapíšeme konstrukci

`\pdfdest`
místo
velikost_zobrazení

Jakožto *místo* uvedeme buď

`num` *přirozené_číslo*

nebo

`name {`*identifikátor*`}`

Položky `num` *přirozené_číslo* a `name {`*přirozené_číslo*`}` při témže argumentu *přirozené_číslo* jsou brány jako různé (srv. sekci 5.1). Pokud se formálně shodují označení zřetězení a *místo* doskoku, ke kolizi nedojde; pdfTEX je chápe jako dvě různé položky. Pojem *identifikátor* má zde týž význam jako v kap. 5.1. Jako *velikost_zobrazení* zapíšeme právě jednu položku z následujícího seznamu. Volíme

`fit`	aby se stránka právě vešla do okna
`fith`	aby šířka stránky souhlasila s šířkou okna
`fitv`	aby výška stránky souhlasila s výškou okna
`fitb`	aby se text právě vešel do okna
`fitbh`	aby šířka textu souhlasila s šířkou okna
`fitbv`	aby výška textu souhlasila s výškou okna
`fitr width` *dimenze* `height` *dimenze* `depth` *dimenze*	aby se obdélník se specifikovanými dimenzemi právě vešel do okna
`xyz`	aby se zachovalo aktuální zvětšení
`xyz zoom` *faktor* ...	aby zvětšení bylo dáno faktorem = = promilemi zvětšení

Jestliže je někde na začátku dokumentu uvedena specifikace

`\pdfdestmargin=`*dimenze*

pak se tento rozměr připočte k rozměrům definovaným u `fitr`. (Rozumná hodnota *dimenze*: asi 10 pt.)

Příklad:

```
\pdfdest num 99 fitv
```

Příklad:

```
\pdfdest name {w-4.1} xyz zoom 1500
```

7.2. Aktivní odkaz

Aktivní odkaz v dokumentu je definován následující konstrukcí:

```
\pdfstartlink
  plocha_odkazu
  akce
  obsah_plochy_odkazu
\pdfendlink
```

Zde *plocha_odkazu* definuje jednak velikost plochy, na níž změna tvaru kurzoru indikuje možnost po kliknutí přejít na cíl, a jednak barvu a tloušťku rámečku ohraničujícího tuto plochu. Syntaxe:

```
width dimenze height dimenze depth dimenze
attr{/C [R G B] /Border [0 0 tloušťka] /H zvýraznění}
```

V prvním řádku jsou specifikovány rozměry plochy odkazu. Chybějí-li (některé nebo všechny), použijí se odpovídající rozměry boxu s obsahem plochy odkazu (viz dále). Ve druhém řádku výraz `/C [`*R G B*`]` popisuje barvu rámečku v systému RGB, přičemž *R*, *G*, *B* jsou číselná vyjádření příslušných barevných složek, a *tloušťka* ve výrazu `/Border [ ... ]` je číslo, vyjadřující tloušťku tohoto rámečku v postscriptových bodech (tj. v jednotkách bp). Tloušťka rámečku se nezmění při zvětšení dokumentu v prohlížeči. Při tisku dokumentu rámeček zmizí. Implicitně je barva rámečku černá (chybí-li výraz `/C [ ... ]`) a jeho tloušťka 1 bp (chybí-li výraz `/Border [ ... ]`). Abychom dostali rámeček čárkovaný, specifikujeme `/Border [0 0` *tloušťka* `[`*čárky mezery*`]]`, kde na místě termínů v kurzívě jsou nezáporná čísla, udávající následující rozměry v postscriptových bodech (bp): *tloušťka* = toušťka rámečku, *čárky* = délka každé z čárek, *mezery* = délka každé z mezer mezi čárkami. Chceme-li mít plochu bez rámečku, zapíšeme `/Border [0 0 0]`. Položka *zvýraznění* určuje, zda se změní vzhled plochy odkazu v době, kdy tiskneme tlačítko myši při umístění kurzoru v ploše odkazu. Tato položka může nabývat jedné z hodnot:

`/I` ..... invertují se barvy obsahu i rámečku (implicitně nastaveno, tedy parametr `/H /I` se může vynechat)

`/O` ..... invertuje se pouze barva rámečku

`/P` ..... obsah plochy odkazu se zdánlivě vmáčkne pod úroveň obrazovky

`/N` ..... vzhled plochy odkazu se nezmění

Speciálně, jestliže v konstrukci `\pdfstartlink` ... `\pdfendlink` deklarujeme `attr {/C [1 1 1] /H /O}`, zobrazí se obsah plochy odkazu bez rámečku (přesněji: s rámečkem bílým, a tedy neviditelným na bílém pozadí), ale po dobu tisknutí tlačítka myši s kurzorem nad plochou odkazu se objeví černý rámeček. Jako *obsah_plochy_odkazu* budeme užívat text nebo obrázek. Syntaxe položky *akce* se liší podle typu hypertextového odkazu. Její popis bude předmětem zbytku této

sekce 7.2. Pozor – konstrukci `\pdfstartlink` ... `\pdfendlink` nelze použít ve vertikálním módu! Je-li tedy tato konstrukce na samém začátku odstavce, je nutné přejít do horizontálního módu – např. před `\pdfstartlink` uvést některý z příkazů `\leavevmode`, `\indent`, `\noindent`, `\hskip`..., nebo konstrukci `\pdfstartlink` ... `\pdfendlink` vložit do `\hbox`u či jeho odvozenin, jako `\leftline`.

Aktivní odkazy mohou být deklarovány i ve `\footline` nebo `\headline`, a také v položkách tabulky.

Nepředepíšeme-li rozměry `width`..., `height`... a `depth`... plochy odkazu obsahující text, bude rámeček nalepen těsně k tomuto textu, což je nehezké. Protože určení rozměrů bývá nepohodlné, je snazší rozměry nepředepisovat, ale v textu použít podpěru `\strut` a vložit mezery na začátek a konec textu. Je tu ale ještě jiná možnost: příkaz

`\pdflinkmargin=`*dimenze*

přidá dodatečné okraje velikosti *dimenze* kolem plochy odkazu. Tento příkaz budeme většinou uvádět někde na začátku zdrojového textu, aby se užil pro všechny hypertextové odkazy v dokumentu. Typická hodnota *dimenze* je 1–2 pt.

Vyzkoušejte si nyní různé způsoby zvýraznění plochy aktivního odkazu při klepnutí myší:

/Border [0 0 4] /C [0.25 0.75 0.5]

/H /I	/H /O	/H /P	/H /N
Klepni zde	Klepni zde	Klepni zde	Klepni zde

/Border [0 0 4] /C [1 1 1]

/H /I	/H /O
Klepni zde	Klepni zde

/Border [0 0 0]

/H /I	/H /O	/H /P	/H /N
			

7.2.1. Odkaz na místo v dokumentu. Položka *akce* má tvar

`goto` *místo*

kde specifikace *místo* je táž jako v příslušné deklaraci `\pdfdest`... na začátku sekce 7.1.

Příklad:

(zde předpokládáme, že někde ve zdrojovém textu existuje deklarace doskoku `\pdfdest name{w-4.1} xyz`)

```
Zde je příklad na hypertextový odkaz. Po klepnutí
  \pdfstartlink
    height 10 pt depth 4pt
    attr{/C [0.5 0 0]}
    goto name {w-4.1}%
    \pdfliteral{0.5 0 0 rg}
    na toto místo
    \pdfliteral{0 0 0 rg}%
  \pdfendlink
\ se v nezměněném zvětšení zobrazí stránka s cílem,
který je označen jménem w-4.1.
```

Příklad:

(zde předpokládáme, že někde ve zdrojovém textu existuje deklarace doskoku `\pdfdest num 99 xyz`)

```
A zde je jiný příklad na hypertextový odkaz. Klepnete\=li
na tento obrázek
  \pdfstartlink
    attr{/Border [0 0 0]}
    goto num 99
    \pdfximage width 20mm {obrazek.jpg}%
    \pdfrefximage\pdflastximage
  \pdfendlink
\thinspace, se v nezměněném zvětšení zobrazí stránka s cílem,
označeným číslem 99.
```

7.2.2. Odkaz na stránku s daným pořadím. Položka *akce* má tvar

`goto page` *pořadí_stránky*
`{`*velikost_zobrazení*`}`

Co je *pořadí_stránky* v dokumentu, je jasné. (Uvědomte si, že pořadí stránky se liší od čísla stránky, pokud stránkování nezačíná od jedničky.) A co znamená *velikost_zobrazení*? Je to právě jedna z následujících položek:

```
/Fit        /FitB
/FitH       /FitBH
/FitV       /FitBV
/XYZ
```

Význam je stejný jako v případě analogických položek *velikost_zobrazení* v deklaraci `\pdfdest` doskoku (sekce 7.1). Rozdíl je formální v pozměněném zápisu: místo `fit` nyní máme `{/Fit}` atd.

Některé z těchto položek velikosti zobrazení mohou být doplněny číselnými údaji, které způsobí posun obsahu obrazovky; v následujícím popisu jsou x a y čísla udávající souřadnice v postscriptových bodech (bp). Poznamenejme, že počátek souřadnic leží v levém dolním rohu stránky, resp. textu.

`/FitH` y `/FitBH` y ... vertikální posun, souřadnice y přejde k horní straně okna

`/FitV` x `/FitBV` x ... horizontální posun, souřadnice x přejde k levé straně okna

`/XYZ` x y ..... vertikální + horizontální posun, bod (x, y) přejde do levého horního rohu okna

`/XYZ` x y z ..... totéž, ale zobrazení je v z-násobném zvětšení

A ještě další možnost:

`/FitR` x_{ld} y_{ld} x_{ph} y_{ph} ... stránka se zobrazí v největším možném zvětšení tak, aby se obdélník s levým dolním rohem $(x_{\mathrm{ld}}, y_{\mathrm{ld}})$ a pravým horním rohem $(x_{\mathrm{ph}}, y_{\mathrm{ph}})$ vešel do okna (jestliže jeden z rozměrů bude menší než příslušný rozměr okna, obdélník se v okně vystředí)

Vyzkoušejte si různá zvětšení (postupně klepněte myší do rámečků):[6]

podle výšky textu	podle šířky textu	podle výšky stránky	podle šířky stránky
`/FitBV`	`/FitBH`	`/FitV`	`/FitH`

7.2.3. Odkaz do jiného dokumentu PDF. Odkaz do dokumentu, řekněme, *jméno*`.pdf`, lze snadno vytvořit, je-li cíl v dokumentu *jméno*`.pdf` specifikován pomocí `name` (nikoli `num`!), nebo je-li cílem stránka s daným pořadím v dokumentu *jméno*`.pdf`. V obou případech stačí v konstrukci `\pdfstartlink` ... `\pdfendlink` vložit mezi „`goto`“ a „`name`“, resp. mezi „`goto`“ a „`page`“, slovo `file` následované názvem souboru ve složených závorkách. Položka *akce* má tedy tvar

`goto file {`*jméno*`.pdf} name {`*identifikátor*`}`

resp.

`goto file {`*jméno*`.pdf} page` *pořadí_stránky* `{`*velikost_zobrazení*`}`

Položka *identifikátor* je vymezena v sekci 5.1 a položka *velikost_zobrazení* v odstavci 7.2.2.

Poznámka. Jestliže soubor *jméno*`.pdf` se nachází v jiném adresáři než soubor, z něhož odkazujeme, musíme jeho název uvést i s cestou. Přitom jako oddělovače adresářů musíme užít lomítka, nikoli zpětná lomítka!

[6] Ve skutečnosti tato ukázka nebyla vytvořena odkazy na (aktuální) stránku, ale byla nasimulována odkazy na cíle specifikované příkazy `\pdfdest name {...} fit`...

7.2.4. Odkaz na zřetězení. Položka *akce* má tvar

`thread` *označení*

přičemž *označení* je položka popsaná v kap. 5, a určuje, do kterého zřetězení máme vstoupit. Jestliže odkazujeme na zřetězení v jiném souboru, pak ovšem v *označení* musí být `name` (nikoli `num`! – srv. odst. 7.2.3) a pro položku *akce* použijeme

`thread file {`*jméno*`.pdf} name {`*identifikátor*`}`

I zde platí poznámka z odst. 7.2.3. Výstup ze zřetězení nás vrátí na místo odkazu.[7]

Vyzkoušejte!

7.2.5. Odkaz na akci menu. Dosud jsme brali v úvahu hyperlinky, které odkazovaly na konkrétní místa v dokumentu nebo stránky s konkrétním pořadím. Jsou však i jiné možnosti: Položka *akce* odkazuje na některou položku menu prohlížeče, má-li tvar

```
user{/Type /Action
     /Subtype /Link
     /A << /S /Named
           /N akce_menu
        >>
    }%
```

kde *akce_menu* je jedna z následujících položek:

`/GoBack`	skok na předchozí místo v dokumentu
`/FirstPage`	skok na první stránku
`/LastPage`	skok na poslední stránku
`/PrevPage`	skok na předcházející stránku
`/NextPage`	skok na následující stránku
`/ShowBookmarks`	zobrazí se záložky (viz 7.3)
`/ShowThumbs`	zobrazí se náhledy stránek (viz 7.4)
`/GeneralInfo`	zobrazí se informace o dokumentu (viz 9)
`/Print`	otevře se dialogové okénko pro tisk dokumentu

Tento typ odkazů může být užitečný při celoobrazovkovém režimu prohlížení (bude zmíněn v sekci 8.1), kdy menu není zobrazeno. Poznamenejme, že některé z odkazů definovaných pomocí *akce_menu* uvedených typů lze nahradit odkazy na specifikovanou stránku (viz odst. 7.2.2).

[7] V Adobe Readeru 7.0 se mi uvedeným postupem nepodařilo na místo odkazu se vrátit. V Acrobat Readeru 5.0 ano.

7.2.6. Odkaz na zvukovou nahrávku či videoklip. Abychom na video anotaci mohli odkazovat, musíme ji „pojmenovat“, a to tak, že v její specifikaci uvedeme parametr `/T (`*identifikátor*`)`, přičemž *identifikátor* je textový řetězec:

```
\pdfannot
  width 0pt height 0pt depth 0pt
  {/Subtype /Movie /T (identifikátor) /Border [0 0 0]
   /Movie << /F (cesta/jméno.wav) >>
  }
```

resp.

```
\pdfannot
  width 0pt height 0pt depth 0pt
  {/Subtype /Movie /T (identifikátor) /Border [0 0 0]
   /A << /FWScale [čitatel jmenovatel] /FWPosition [x y] >>
   /Movie << /F (cesta/jméno.mov) >>
  }
```

Význam ostatních parametrů, vyskytujících se v těchto schématech, byl popsán v sekci 6.2. Deklarace anotací mohou pochopitelně obsahovat i další parametry zmíněné v sekci 6.2. Nulové rozměry v druhém řádku uvedených schémat znamenají, že anotace nemají aktivní plochu, a mohou být spuštěny pouze odkazem: Položka *akce* v konstrucki `\pdfstartlink` ... `\pdfendlink` má potom tvar

```
user{/Subtype /Link
     /A << /Type /Action
           /S /Movie /T (identifikátor)
           /Operation běh
        >>
   }%
```

přičemž *identifikátor* zde ve specifikaci *akce* odkazu i ve specifikaci anotace je tentýž. Položka *běh* za `/Operation` určuje, co se má dít se záznamem. Může mít jednu z těchto hodnot:

```
/Start  ..... spuštění
/Pause  ..... přerušení
/Remote  ..... pokračování po předchozím přerušení
/Stop  ..... ukončení
```

Příklad:

```
\pdfannot
  width0pt height0pt depth0pt % nulová aktivní plocha
  {/Subtype /Movie /T (zvuk) /Border [0 0 0]
   /Movie << /F (audio.wav) >> }%
```

```
{Zde
\pdfstartlink
  height10pt depth4pt
  attr{/C [0 0.8 0.1] /Border [0 0 2]}
  user{/Subtype /Link
       /A << /Type /Action
             /S /Movie /T (zvuk) /Operation /Play >> }%
  \ spouštím
\pdfendlink
\ nahrávku \dots\ teď ji
\pdfstartlink
  height10pt depth4pt
  attr{/C [1 0.9 0] /Border [0 0 2]}
  user{/Subtype /Link
       /A << /Type /Action
             /S /Movie /T (zvuk) /Operation /Pause >> }%
  \ přeruším
\pdfendlink
\ \dots\ \par\hskip60pt a zase
\pdfstartlink
  height10pt depth4pt
  attr{/C [0 0.9 0] /Border [0 0 2]}
  user{/Subtype /Link
       /A << /Type /Action
             /S /Movie /T (zvuk) /Operation /Resume >> }%
  \ pokračuji
\pdfendlink
\ \dots\ a zde nahrávku
\pdfstartlink
  height10pt depth4pt
  attr{/C [1 0 0] /Border [0 0 2]}
  user{/Subtype /Link
       /A << /Type /Action
             /S /Movie /T (zvuk) /Operation /Stop >> }%
  \ ukončím
\pdfendlink
\ .\advance\baselineskip by 3pt\par}
```

Výstup:

Zde spouštím nahrávku ... teď ji přeruším ...

a zase pokračuji ... a zde nahrávku ukončím .

Obsah rámečku v době přehrávání nemizí – na rozdíl od situace zmíněné v poznámce z odst. 6.2.2.

Pokud bychom v deklaraci anotace videoklipu `.mov` (uvedena na začátku tohoto odst. 7.2.6) vynechali slovník `/A <<... >>`, pak při odkazu by se přehrál pouze zvuk videoklipu.

7.2.7. Odkaz na URL. Odkazovat můžeme též na zdroj na internetu – na webovou stránku, e-mailovou adresu, na diskusní skupinu apod. Položka *akce* má v tomto případě tvar

```
user{/Subtype /Link
     /A << /Type /Action
           /S /URI
           /URI (URL)
        >>
    }%
```

Zde *URL* je adresa na odkazovaný zdroj na Internetu.

Příklad:

```
\noindent Podívejte se
\pdfstartlink
  height10pt depth4pt
  attr{/C [0.5 0 0]}
  user{/Subtype /Link
       /A << /Type /Action
             /S /URI
             /URI (http://www.cstug.cz/stanovy/index.html)
          >>
      }%
  {\pdfliteral{0 1 1 0 k}\bf\ na stanovy
   \pdfliteral{0 0 0 1 k}}%
\pdfendlink
\ Československého sdružení uživatelů \TeX{}u.
```

Výstup (vyzkoušejte):

Podívejte se **na stanovy** Československého sdružení uživatelů TEXu.

Jiná varianta tohoto typu odkazu spočívá v zadání základní URL (např. doménové adresy serveru, na němž je umístěn cíl odkazu) příkazem

```
\pdfcatalog{/URI << /Base (základní_URL) >>}
```

a v odkazování pomocí relativní URL (v tomto případě jména cílového souboru s cestou):

```
user{/Subtype /Link
     /A << /Type /Action
           /S /URI
           /URI (relativní_URL)
        >>
    }%
```

Zadání základní URL je platné pro celý dokument a může být zapsáno kdekoli v něm. Předchozí příklad můžeme tedy přeformulovat, při zachování výstupu, takto:

Příklad:

```
\pdfcatalog{/URI << /Base (http://www.cstug.cz/) >>}
.........
\noindent Podívejte se
\pdfstartlink
  height10pt depth4pt
  attr{/C [0.5 0 0]}
  user{/Subtype /Link
       /A << /Type /Action
             /S /URI
             /URI (stanovy/index.html)
          >>
      }%
  {\pdfliteral{0 1 1 0 k}\bf\ na stanovy
   \pdfliteral{0 0 0 1 k}}%
\pdfendlink
\ Československého sdružení uživatelů \TeX{}u.
```

Základní URL může obsahovat i část cesty, relativní URL pak je tvořena zbývající částí cesty; tak např. výstup předchozího příkladu se nezmění, deklarujeme-li základní URL příkazem

```
\pdfcatalog{/URI << /Base (http://www.cstug.cz/stanovy/) >>}
```

a užijeme-li v konstrukci `\pdfstartlink`... `\pdfendlink` tuto relativní URL:

```
......
user{/Subtype /Link
     /A << /Type /Action /S /URI
           /URI (index.html) >>}
......
```

Speciálně může základní URL již být sama adresou některého zdroje na Internetu: V takovém případě je ale relativní URL prázdným řetězcem, takže

parametr `/URI (...)` lze vynechat; např., je-li v dokumentu deklarováno

```
\pdfcatalog
  {/URI << /Base (http://www.cstug.cz/stanovy/index.html) >>}
```

pak můžeme předchozí příklad pozměnit takto:

```
......
user{/Subtype /Link
     /A << /Type /Action /S /URI >>}
......
```

Specifikace základní URL příkazem `\pdfcatalog{...}` nám umožňuje odkazovat stručněji pomocí URL, na druhé straně nám ale nijak nezabraňuje, abychom v odkazech užívali kompletní URL, chceme-li.

Podobně můžeme též odkazovat na soubory na lokálním disku, mají-li takový formát, aby šly zobrazit internetovým prohlížečem. Oddělovači adresářů jsou lomítka. Názvy cest a souborů nesmí obsahovat mezery (v adresovém okénku internetového prohlížeče by totiž zmizely a soubor by pro prohlížeč nebyl k nalezení). Položka *akce* může například vypadat takhle:

```
user{/Subtype /Link
     /A << /Type /Action
           /S /URI
           /URI (file://C:/dokumenty/typo.html)
        >>
    }%
```

Nebo, při specifikaci

```
\pdfcatalog{/URI << /Base (file://C:/dokumenty/) >>}
```

může *akce* být zapsána jako

```
user{/Subtype /Link
     /A << /Type /Action /S /URI
           /URI (typo.html) >>}
```

Pokud v dokumentu nespecifikujeme základní URL, pak se relativní URL vztahuje k adresáři obsahujícímu dokument, z něhož odkazujeme.

Problémy jsou při přenesení dokumentu na jiný počítač: musíme pamatovat na to, že soubor, na nějž odkazujeme, musí být umístěn na disku tak, aby cesta k němu byla stejná jako ta původní.

Poznamenejme, že příkaz `\pdfcatalog{...}` může obsahovat i další parametry (nesouvisející s právě probíraným tématem). Více o tom bude v sekcích 7.4 a zejména 8.1.

7.2.8. Spuštění programu nebo otevření dokumentu. Má-li se při klepnutí na plochu odkazu spustit určitý program nebo otevřít některý dokument, specifikujeme položku *akce* takto:

```
user{/Subtype /Link
     /A << /Type /Action
           /S /Launch
           /F (soubor)
        >>
    }%
```

Zde *soubor* je jedna z následujících dvou možností:

– jméno souboru, kterým se spouští program
– jméno souboru, obsahujícího dokument, který má být otevřen

Jméno souboru se uvádí s extenzí. Pokud soubor není v témže adresáři jako dokument, z něhož je volán, musí se jeho jméno uvést s cestou – buď absolutní nebo relativní; oddělovači adresářů jsou lomítka, před označení disku se též zapisuje lomítko a dvojtečka za označením disku se nepíše.

V položce *soubor* nelze uvést parametry. Pokud spuštění programu parametry vyžaduje, pomůžeme si tak, že příkaz ke spuštění programu (s parametry) zapíšeme do dávkového souboru, a ten spouštíme.

Pokud souborem je dokument, pak jeho typ musí být přidružen některému programu pro jeho otevření. Jestliže tomu tak není, pak příkaz k otevření programu zapíšeme do dávkového souboru, a ten spouštíme.

Přenositelnost na jiné počítače je problematická, pokud program, který má být spuštěn, není součástí distribuce dokumentu, nebo pokud nepředpokládáme, že cílový počítač je nakonfigurován tak, aby uměl otevírat soubory příslušných typů.

7.3. Záložky

Záložky (*outlines*, *bookmarks*) jsou hypertextové odkazy se speciálním umístěním – v okénku vedle zobrazeného dokumentu nebo ve zvláštním rámečku. Klepneme-li na záložku, prohlížeč nalistuje stránku, která obsahuje místo, na něž záložka odkazuje. Je zvykem záložky uspořádat do hierarchie sledující strukturu dokumentu (tak, aby odkazovaly na jednotlivé nadpisy v něm), a představující tak vlastně obsah. Podobně jako kapitola může mít několik podkapitol, záložka může mít několik potomků (podřazených záložek). Záložka může mít počáteční nastavení jako rozvinutá, kdy jsou vidět její přímí potomci (podřazené záložky na nejblíže nižší úrovni), nebo svinutá, kdy potomky můžeme spatřit až poté, co ji rozvineme.

Záložku vytvoříme následujícím způsobem:

```
\pdfoutline
  akce
  count počet_potomků
  {text_záložky}
```

Položka *akce* je typu `goto` (viz 7.2.1, 7.2.2, 7.2.3) nebo `thread` (viz 7.2.4). Absolutní hodnota celého čísla *počet_potomků* udává počet přímých potomků deklarované záložky. Je-li číslo *počet_potomků* kladné, záložka je otevřená, je-li záporné, záložka je uzavřená. Nemá-li záložka potomky, pak *počet_potomků* je 0; v tomto případě můžeme deklaraci `count`... vynechat. Poznámka v sekci 6.1, týkající se textů anotací, platí také pro *text_záložky*.

Záložky můžeme deklarovat kdekoli ve zdrojovém textu dokumentu, avšak v tom pořadí, v jakém se mají zobrazit. Jejich deklarace tedy nemusí nutně následovat bezprostředně jedna za druhou, jako je tomu zde:

Příklad:

```
\pdfoutline goto name {U} {Uvod}
\pdfoutline goto name {1} {1. Parametry dokumentu}
\pdfoutline goto name {2} count 3 {2. Uziti kodu PDF pro upravy tisku}
  \pdfoutline goto name {2.1} count -4 {2.1. Zvyraznovani barvou}
    \pdfoutline goto name {2.1.1} {2.1.1. Zvyrazneni tisku}
    \pdfoutline goto name {2.1.2} count 1 {2.1.2. Prepinace barev}
      \pdfoutline thread name {cmyk} {Tabulka preddefinovanych barev}
    \pdfoutline goto name {2.1.3} {2.1.3. Zvyrazneni pozadi}
    \pdfoutline goto name {2.1.4} {2.1.4. Barevna matematika}
  \pdfoutline goto name {2.2} count -3 {2.2. Smesovani barev}
    \pdfoutline goto name {2.2.1} {2.2.1. Prekryvani ploch}
    \pdfoutline goto name {2.2.2} {2.2.2. Prekryvani pismen}
    \pdfoutline goto name {2.2.3} {2.2.3. Prusvitne barvy}
  \pdfoutline goto name {2.3} count -8 {2.3. Geometricke transformace}
    \pdfoutline goto name {2.3.1} {2.3.1. Otoceni o ostry uhel doleva}
    ......................
    \pdfoutline goto name {2.3.8} {2.3.8. Zkoseni ve svislem smeru}
  \pdfoutline goto name {2.4} count 2 {2.4. Globalni upravy tisku}
    \pdfoutline goto name {2.4.1} {2.4.1. Zrcadlovy tisk}
    \pdfoutline goto name {2.4.2} {2.4.2. Barevny podtisk a barevne pismo}
\pdfoutline goto name {3} {3. Uprava okraju tiskoveho zrcadla}
......................
```

Někdo dá přednost deklaracím záložek spolu s tiskem nadpisů kapitol (anebo jiných úseků textu). Abychom mohli nadpisy kapitol použít přímo jako texty záložek, je užitečné vytvořit makro, které vysází číslo a název kapitoly, deklaruje doskok v místě tohoto nadpisu (jako *jméno* v argumentu `name` použije číslo kapitoly) a deklaruje příslušnou záložku (přičemž jejím textem je název kapitoly bez

diakritických znamének). Nejprve ale definujeme pomocné makro `\noaccents`, které odřezává diakritická znaménka v textovém řetězci. Činnost tohoto makra spočívá v aplikaci příkazu `\lowercase` po předchozím předefinováním příkazu `\lccode` tak, aby mapoval písmena s diakritikou na táž písmena bez ní (při zachování velikosti písmen) a velká písmena bez diakritiky na sebe. Toto předefinování je lokální v makru `\noaccents`.

```
\newcount\ASCIIcode
\def\noaccents#1{{%
  \ASCIIcode=65
  \loop \lccode\ASCIIcode=\ASCIIcode
    \ifnum\ASCIIcode<90 \advance\ASCIIcode by 1 \repeat
  \lccode‘Á=‘A \lccode‘É=‘E \lccode‘Í=‘I \lccode‘Ő=‘O
  \lccode‘Ű=‘U \lccode‘Ý=‘Y \lccode‘Ě=‘E \lccode‘Ů=‘U
  \lccode‘Ä=‘A \lccode‘Ö=‘O \lccode‘Ü=‘U \lccode‘Č=‘C
  \lccode‘Ř=‘R \lccode‘Š=‘S \lccode‘Ž=‘Z \lccode‘Ď=‘D
  \lccode‘Ť=‘T \lccode‘Ň=‘N \lccode‘Ō=‘O \lccode‘Ľ=‘L
  \lccode‘Ĺ=‘L \lccode‘Ŕ=‘R \lccode‘á=‘a \lccode‘é=‘e
  \lccode‘í=‘i \lccode‘ó=‘o \lccode‘ú=‘u \lccode‘ý=‘y
  \lccode‘ě=‘e \lccode‘ů=‘u \lccode‘ä=‘a \lccode‘ö=‘o
  \lccode‘ü=‘u \lccode‘č=‘c \lccode‘ř=‘r \lccode‘š=‘s
  \lccode‘ž=‘z \lccode‘ď=‘d \lccode‘ť=‘t \lccode‘ň=‘n
  \lccode‘ô=‘o \lccode‘ľ=‘l \lccode‘Í=‘l \lccode‘ŕ=‘r
  \lowercase{#1}}}
```

A nyní slíbené makro pro tvorbu nadpisů kapitol & příslušných záložek (předpokládáme, že máme definován přepínač `\BF` fontu pro názvy kapitol):

```
\def\head#1 #2 #3 {%
  % první par.: číslo kapitoly = jméno doskoku
  % druhý par.: název kapitoly = text záložky
  % třetí par.: počet podkapitol = počet podřazených záložek
  \vskip2\baselineskip
  \pdfdest name{#1} xyz
  \centerline{\BF #1. \ #2}%
  \noaccents{\pdfoutline goto name{#1} count #3 {#2}}%
  \nobreak \vskip\baselineskip}
```

Toto makro nelze použít, pokud název kapitoly obsahuje TEXové řídicí sekvence nebo vzorec zapsaný v matematickém módu.

Příklad:

```
\head 4 {Vkládání externích objektů} 3 % 3 sekce v kap. 4
```

7.4. Náhled stránek

Dalším prostředkem pro snadný pohyb v zobrazeném dokumentu jsou miniatury stránek (*thumbnails*), které jsou podobně jako záložky umístěny ve zvláštním okénku nebo rámečku. Při klepnutí na miniaturu se odpovídající originální stránka zobrazí v prohlížeči. Pod miniaturami stránek jsou uvedena jejich čísla. Jsou-li stránky v dokumentu číslovány arabskými číslicemi od jedné, pak se čísla pod miniaturami s čísly stránek shodují. Jinak je zapotřebí číslování pod miniaturami předefinovat. K tomu užijeme konstrukci

```
\pdfcatalog
  {/PageLabels << /Nums [seznam_tříd_stránkování] >>}
```

Zde předpokládáme, že dokument je rozčleněn na několik částí, z nichž každá má jiný způsob číslování stránek. Popis každého z těchto stránkování jazykem PDF tvoří třídu stránkování. Jejich souhrn, `[`*seznam_tříd_stránkování*`]`, má tvar

```
[pozice << číslování >>
 pozice << číslování >>
   .........
 pozice << číslování >> ]
```

Položka *pozice* je číslo udávající pozici počáteční stránky konkrétní části dokumentu vzhledem k začátku dokumentu. Pro první stránku dokumentu je *pozice* rovna 0, pro stránku dokumentu v pořadí n-tou je *pozice* rovna $n - 1$. Položka *číslování* specifikuje způsob stránkování příslušné části dokumentu, a může mít až tři složky

```
/S  styl
/P  (prefix_čísel_stránek)
/St číslo_počáteční_stránky
```

U položky *styl* máme na výběr následující styly stránkování:

```
/D ..... arabskými číslicemi
/r ..... malými římskými číslicemi
/R ..... velkými římskými číslicemi
/a ..... malými písmeny (a–z, aa–zz, ...)
/A ..... velkými písmeny (A–Z, AA–ZZ, ...)
```

Dále *prefix_čísel_stránek* je text, který se sází před každým číslem stránky v uvažované části dokumentu. Kladné celé *číslo_počáteční_stránky* se musí předepsat, pokud stránkování v dané části dokumentu nezačíná od jedné. (Vždy je to číslo zapsané arabskými číslicemi, bez ohledu na styl stránkování.)

Do dialogového okénka prohlížeče

Jít na stránku nebo Go To Page

vpisujeme označení stránky rovněž ve tvaru, který je užit pod miniaturou stránky.

Příklad. Mějme dokument, sestávající z oddílů uvedených v prvním sloupci následující tabulky, přičemž stránkování dokumentu je předepsáno ve druhém sloupci:

Oddíl dokumentu	Předpis stránkování	Počet stránek oddílu	Pozice počáteční stránky	Číslo počáteční stránky
Předmluva	stránky i, ii, ..., iv	6	0	1
Stať – 1. díl	stránky 1, 2, ..., 19	19	6	1
Obrázky	4 nečíslované stránky	4	25	–
Stať – 2. díl	stránky 20, 21, ..., 35	16	29	20
Dodatky	stránky D-1, D-2, D-3	3	45	1
Literatura	místo čísel stránek název „Literatura“		48	–

Shodu stránkování u miniatur se stránkováním dokumentu docílíme zařazením následujícího příkazu do zdrojového souboru dokumentu:

```
\pdfcatalog
  {/PageLabels << /Nums [ 0 << /S /r >>
                          6 << /S /D >>
                         25 <<       >>
                         29 << /S /D /St 20  >>
                         45 << /S /D /P (D-) >>
                         48 << /P (Literatura) >> ]
               >>}
```

8. Způsob zobrazování dokumentu

Poté, co je dokument vytvořen, můžeme ještě předepsat, jakým způsobem jej zobrazit. A právě tím se bude zabývat tato kapitola. Sekce 8.1 bude věnována zobrazení dokumentu jako celku, sekce 8.2 pak zobrazení jednotlivých stránek.

8.1. Katalog

S příkazem `\pdfcatalog{...}` jsme se setkali v souvislosti s definováním základní URL pro daný dokument (v podsekci 7.2.7) a při předpisu stránkování pod miniaturami stránek (v sekci 7.4). Pomocí dalších parametrů tohoto příkazu můžeme ještě předepsat počáteční nastavení prohlížeče při zobrazení dokumentu. Syntaxe je následující:

```
\pdfcatalog
  {/PageMode okno
   /PageLayout stránky
   /ViewerPreferences << /DisplayDocTitle logická_hodnota
                         /HideToolbar logická_hodnota
                         /HideMenubar logická_hodnota
                         /HideWindowUI logická_hodnota
                      >>
   /PageLabels << /Nums [ seznam_rozsahů_stránkování ] >>
   /URI << /Base (základní_URL) >>
  }
  openaction počáteční_akce
```

Při specifikaci položky *okno* máme následující možnosti:

`/UseOutlines`	s dokumentem se zobrazí i záložky
`/UseThumbs`	s dokumentem se zobrazí i miniatury stránek
`/UseNone`	zobrazí se jen dokument
`/FullScreen`	zobrazí se dokument na celé obrazovce

Při specifikaci položky *stránky* si můžeme vybrat z možností:

`/SinglePage`	zobrazí se jednotlivé stránky
`/OneColumn`	zobrazí se stránky za sebou v pásu
`/TwoColumnLeft`	zobrazí se dvojice stránek v pásu: (1,2) (3,4) ..., tj. liché vlevo
`/TwoColumnRight`	zobrazí se dvojice stránek v pásu: (–,1) (2,3) (4,5) ..., tj. sudé vlevo

Jestliže *logická_hodnota* za klíčem `/DisplayDocTitle` je `false` – což je nastaveno implicitně, na titulním panelu prohlížeče se zobrazí jméno prohlíženého PDF souboru. Jestliže *logická_hodnota* je `true`, na titulním panelu se zobrazí *název_dokumentu* uvedený za `/Title` ve specifikaci `\pdfinfo` (viz kap. 9). Jestliže specifikujeme `/HideToolbar true`, resp. `/HideMenubar true`, resp. `/HideWindowUI true`, pak u prohlížeče se nezobrazí panel s nástroji, resp. menu, resp. stavový řádek a posuvníky. Implicitní nastavení jsou `/Hide..... false`, při nichž příslušná zobrazení nejsou potlačena. Význam `/PageLabels` byl objasněn v sekci 7.4 a význam `/URI` v podsekci 7.2.7. Je-li uveden parametr `openaction` *počáteční_akce*, pak při otevření dokumentu v prohlížeči se provede *počáteční_akce*; ta musí být typu `goto` (viz 7.2.1, 7.2.2, 7.2.3) nebo `thread` (viz 7.2.4). Jinak se dokument otevře na první stránce v implicitním zvětšení.

Tip: Pokud na některé místo ve zdrojovém textu dokumentu vložíme příkaz

```
\pdfcatalog {} openaction goto page \the\pageno {velikost_zobrazení}
```

pak po zpracování pdfTEXem započne prohlížení vzniklého souboru `.pdf` na té straně, která obsahuje místo, v němž byl příkaz uveden. (Možné hodnoty parametru *velikost_zobrazení* byly

popsány v odst. 7.2.2.) Pokud v naší TEXové instalaci spouštíme pdfTEX z editoru, který je patřičně konfigurovatelný a je vybaven dostatečně bohatým makrojazykem, pak můžeme zařídit, aby po překladu zdrojového souboru naskočilo prohlížení výsledného dokumentu na té straně, která obsahuje místo, v němž se nacházel textový kurzor v okamžiku spuštění překladu. K tomu je pouze zapotřebí, aby se do zdrojového souboru bezprostředně před překladem v místě kurzoru automaticky vložil uvedený příkaz `\pdfcatalog...` a po návratu do editoru se tento příkaz automaticky smazal. V dokumentu může ještě na jiném místě být užita konstrukce `\pdfcatalog`, ta však již nesmí obsahovat povel `openaction` – jinak by pdfTEX ohlásil chybu.

Akci, která se má provést při otevření dokumentu, můžeme též specifikovat za klíčem `/OpenAction` mezi parametry ve složených závorkách v příkazu `\pdfcatalog` (v tom případě se položka `openaction...` za `\pdfcatalog{...}` neuvádí):

```
\pdfcatalog
  {...
   /OpenAction počáteční_akce
  }
```

Zde musí být počáteční *počáteční_akce* zapsána přímo v kódu PDF.

Příklad. Při otevření dokumentu se má spustit program (srv. odst. 7.2.8).

```
\pdfcatalog
  {/PageMode ...
   /PageLayout ...
   /ViewerPreferences << ... >>
   /PageLabels << ... >>
   /URI << ... >>
   /OpenAction << /Type /Action
                  /S /Launch
                  /F ( soubor )
               >>
  }
```

kde *soubor* je jméno toho souboru, jímž se spouští program; uvádí se s extenzí. Není-li spouštěcí soubor v témže adresáři jako dokument, jeho jméno se uvádí s (absolutní nebo relativní) cestou; oddělovače adresářů jsou lomítka, před označení disku se též píše lomítko, dvojtečka za označením disku se vynechává.

Příklad. Při otevření dokumentu se má spustit zvuková nahrávka `.wav` nebo videoklip `.mov` (srv. odst. 7.2.6). V předchozím příkladu nahradíme parametr `/OpenAction << ... >>` podle následujícího vzoru:

```
/OpenAction << /Type /Action
               /S /Movie
               /T ( identifikátor )
            >>
```

přičemž parametr `/T (`*identifikátor*`)` je pojmenování video anotace, ve které je definována příslušná nahrávka; syntaxe této anotace je táž jako na začátku odst. 7.2.6.

8.2. Atributy stránek

V kapitole 1 bylo popsáno, jak definovat velikost stránek a umístění tiskového materiálu na nich. S vytvořenými stránkami můžeme však dále manipulovat předepsáním tzv. atributů stránek. Ty umožňují stránky otáčet, ořezávat, nebo je doprovázet přehráváním zvukových či video nahrávek apod.

Globálně se atributy stránek nastaví příkazem

```
\pdfpagesattr{...}
```

kde mezi složenými závorkami jsou uvedeny příslušné atributy. Tak např. pro otočení stránek o 90° doprava, resp. doleva, zapíšeme

```
\pdfpagesattr{/Rotate 90}   resp.   \pdfpagesattr{/Rotate -90}
```

A pro (obdélníkové) oříznutí stránek slouží příkaz

```
\pdfpagesattr{/CropBox [x_ld  y_ld  x_ph  y_ph]}
```

kde $x_{\mathrm{ld}}, y_{\mathrm{ld}}$ jsou souřadnice levého dolního bodu výřezu a $x_{\mathrm{ph}}, y_{\mathrm{ph}}$ souřadnice jeho pravého horního bodu v postscriptových bodech (bp); počátek souřadnic je v levém dolním rohu stránky.

Kromě příkazu `\pdfpagesattr{...}` je k dispozici též příkaz

```
\pdfpageattr{...}
```

který se od předešlého liší v tom, že se netýká všech stran dokumentu, ale ovlivní pouze aktuální stranu a strany následující. Atributy, které jsou uvedeny ve složených závorkách příkazu `\pdfpageattr` se přidají k atributům ve složených závorkách příkazu `\pdfpagesattr`, popř. je přepíší (přesněji: atributy v příkazu `\pdfpageattr` přepíší ty z atributů v příkazu `\pdfpagesattr`, které jsou téhož typu). Vliv příkazu `\pdfpageattr{...}` zrušíte zapsáním `\pdfpageattr{}`; tím obnovíte funkci příkazu `\pdfpagesattr{...}` s atributy v něm předepsanými. Chcete-li, aby příkaz `\pdfpageattr{...}` ovlivnil vždy *pouze* aktuální stranu, předefinujte výstupní rutinu – v případě plainu takto:

```
\output={\plainoutput\global\pdfpageattr{}}
```

Tip: Je-li v dokumentu tiskový materiál na některé stránce otočen o 90° doleva (např. tabulka, která by se v normální pozici na stránku nevešla), je vhodné pro tuto stránku specifikovat `\pdfpageattr{/Rotate 90}`, aby čtenář viděl její obsah v normální pozici a nemusel pokládat monitor na bok. Přitom tisk dokumentu ovlivněn nebude.

Předepsáním atributů stránky podle následujícího schématu docílíme automatické spuštění určitého programu při otevření, resp. uzavření stránky.

```
\pdfpageattr
  {/AA << kdy  << /Type /Action
                  /S /Launch
                  /F ( soubor )
               >>
       >> }
```

Položka *kdy* nabývá jedné z hodnot:

`/O` ..... program se spustí při otevření stránky
`/C` ..... program se spustí při uzavření stránky

a položka *soubor* má týž význam a syntaxi jako v prvním příkladu na str. 79.

Předpokládejme nyní, že máme definovánu video anotaci pojmenovanou pomocí parametru `/T (` *identifikátor* `)` (viz začátek odst. 7.2.6). Následující specifikace atributů některé stránky způsobí, že tato anotace se při otevření, resp. uzavření dotyčné stránky (položka *kdy*) automaticky začne přehrávat nebo se naopak přehrávání ukončí (položka *běh*):

```
\pdfpageattr
  {/AA << kdy  << /Type /Action
                  /S /Movie
                  /T ( identifikátor )
                  /Operation  běh
               >>
       >> }
```

Přitom položka *kdy* má týž význam jako v předchozím příkladu a položka *běh* je táž jako v odst. 7.2.6.

Příklad. Makro `\accomp` deklaruje video anotaci se zvukovou nahrávkou a specifikuje atributy stránky pro její přehrání při otevření stránky.

```
\def\accomp#1{%
  \pdfannot
    width0pt height0pt depth0pt
    {/Subtype /Movie /T (#1) /Border [0 0 0]
     /Movie << /F (#1.wav) >> }%
  \pdfpageattr
    {/AA << /O << /Type /Action /S /Movie /T (#1) >> >>}%
  }
```

Zapíšeme-li pak ve zdrojovém textu dokumentu příkaz `\accomp{abcd}`, pak otevření příslušné stránky bude doprovázeno nahrávkou `abcd.wav` (pokud je tento soubor v témže adresáři jako dokument).

9. Vlastnosti dokumentu

Vlastnosti dokumentu najdeme v prohlížeči:

Soubor → Vlastnosti dokumentu → Popis

popř.

File → Document properties → Summary

Zobrazí se tam pouze ty údaje, které se prohlížeči podaří zjistit. Potřebujeme-li je změnit nebo doplnit o další, ve zdrojovém textu uvedeme následující příkaz s některými z uvedených položek mezi složenými závorkami:

```
\pdfinfo
  {/Author (jméno_autora)
   /Subject (předmět_dokumentu)
   /Title (název_dokumentu)
   /Keywords (klíčová_slova)
   /Creator (...)  % implicitně: TeX
   /Producer (...) % implicitně: pdfTeX & verze
   /CreationDate (D:datum_a_čas) % implicitně: aktuální datum
   /ModDate (D:datum_a_čas)
  }
```

přičemž *datum_a_čas* má následující tvar:

RRRRMMDDhhmmss

kde na místě kurzívních písmen jsou číslice tvořící následující číselné položky:

RRRR	rok	*hh*	hodina
MM	měsíc	*mm*	minuty
DD	den	*ss*	sekundy

V textových řetězcích se vyhneme diakritice.

Příklad: Na konci zdrojového textu tohoto dokumentu by mohlo být uvedeno:

```
\pdfinfo
  {/Author (Frantisek Chvala)
   /Subject (Navody k uzivani pdfTeXu)
   /Title (O moznostech pdfTeXu)
   /Keywords (PDF, externi objekty, zretezeni clanku, anotace)
   /CreationDate (D:20050506153000)
     % dokument dokončen 6. května 2005 v půl čtvrté odpoledne
   /ModDate (D:20050513120000)
     % poslední revize dokumentu 13. května 2005 v pravé poledne
  }
```

Poznamenejme, že verzi užívaného pdfTEXu můžeme vytisknout v dokumentu. K tomu účelu si definujeme makro `\writeversion`, které užívá řídicí sekvence `\pdftexversion` (stonásobek čísla verze) a `\pdftexrevision` (písmenko označující revizi):

```
\newcount\intpart \newcount\fracpart
\def\writeversion{%
  \fracpart=\pdftexversion \intpart=0
  \loop
    \ifnum\fracpart>100 \advance\fracpart by -100 \advance\intpart by 1
  \repeat
  \the\intpart.\ifnum\fracpart<10 0\fi\the\fracpart\pdftexrevision}
```

Příklad:

```
Pro sazbu tohoto dokumentu byla užita verze \writeversion\ pdf\TeX{}u.
```

Výstup:

Pro sazbu tohoto dokumentu byla užita verze 1.21a pdfTEXu.

Dodatek: konfigurace pdfTEXu

Některé řídicí sekvence pdfTEXu jsou pojmenováními vnitřních registrů, ve kterých jsou uloženy údaje určující vzhled dokumentu a další jeho charakteristiky. Ve zdrojovém textu dokumentu můžeme tyto údaje specifikovat příslušnými pdfTeXovými příkazy *\řídicí_sekvence* `=` *hodnota*. Tato přiřazení ale mohou být realizována již při generování pdfTEXového formátu, a určují tak implicitní nastavení. Pokud některé takové přiřazení chybí, je příslušný implicitní údaj definován programem pdfTEX (druhý sloupec tabulky). Pozor – některé registry při nulové hodnotě mají jiný význam (třetí sloupec tabulky).

Pojmenování registru	Implicitní hodnota	Význam při nulové implicitní hodnotě
`\pdfoutput`	0	
`\pdfcompresslevel`	9	
`\pdfdecimaldigits`	4	
`\pdfpkresolution`	0	72
`\pdfprotrudechars`	0	
`\pdfhorigin`	1 in	
`\pdfvorigin`	1 in	
`\pdfpagewidth`	0 pt	`\hsize` + 2 (`\pdfhorigin` + `\hoffset`)
`\pdfpageheight`	0 pt	`\vsize` + 2 (`\pdfvorigin` + `\voffset`)
`\pdfdestmargin`	0 pt	
`\pdflinkmargin`	0 pt	
`\pdfthreadmargin`	0 pt	

Významy všech těchto registrů byly objasněny v článku. Co bylo dosud řečeno v tomto dodatku, platí pro pdfTEXu verze 1.21a. Ve verzi 1.11b se implicitní

nastavení parametrů dokumentu neurčuje při generování formátu, ale načítá se při startu programu pdfTEX z jistého konfiguračního souboru. Dodefinování implicitních hodnot při chybějící specifikaci se liší od údajů v předchozí tabulce v následujících položkách:

Pojmenování registru	Implicitní hodnota
`\pdfcompresslevel`	0
`\pdfpkresolution`	600
`\pdfhorigin`	0 pt
`\pdfvorigin`	0 pt

Nic nezkazíte, nebudete-li spoléhat na implicitní nastavení parametrů dokumentu a uvedete-li jejich hodnoty ve zdrojovém textu.

Literatura

[1] Petr Olšák: *TEXbook naruby*. Konvoj, Brno 1997.
Elektronická verze: ftp://math.feld.cvut.cz/pub/olsak/tbn/tbn.pdf

[2] Hàn Thế Thành, Sebastian Rahtz, Hans Hagen, Hartmut Henkel: *The pdfTEX user manual*, 2004.
http://www.tug.org/applications/pdftex/pdftex-a.pdf

[3] Vít Zýka: *Používáme pdfTEX: vkládání obrázků*. Zpravodaj Československého sdružení uživatelů TEXu 11 (2001), č. 4, 181–186.
Elektronická veze čísla: http://bulletin.cstug.cz/pdf/bul_014.pdf

[4] Vít Zýka: *Používáme pdfTEX II: prezentace fotografií aneb jak na hypertext*. Zpravodaj Československého sdružení uživatelů TEXu 12 (2002), č. 1, 13–21.
Elektronická veze čísla: http://bulletin.cstug.cz/pdf/bul_021.pdf

[5] Vít Zýka: *Používáme pdfTEX III: video a zvuk v prezentaci.* Zpravodaj Československého sdružení uživatelů TEXu 12 (2002), č. 2, 47–55.
Elektronická veze čísla: http://bulletin.cstug.cz/pdf/bul_022.pdf

[6] Vít Zýka: *TEX a PDF*. SLT 2002 (J. Kasprzak a P. Sojka, editoři), Brno 2002, str. 69–77.
Elektronická verze sborníku: http://www.cstug.cz/slt/02/slt02.pdf

[7] Vít Zýka: *Používáme pdfTEX IV: mikrotypografické rozšíření.* Zpravodaj Československého sdružení uživatelů TEXu 14 (2004), č. 2, 47–53.
Elektronická veze čísla: http://bulletin.cstug.cz/pdf/bul_042.pdf

[8] *PDF Reference*.
http://partners.adobe.com/asn/tech/pdf/specifications.jsp

Rejstřík

Rejstřík uvádí především řídicí a klíčová slova pdfTEXu (která nejsou obsažena v TEXu), jež jsou popsána v článku; nejsou zahrnuty operátory, klíče a hodnoty jazyka PDF.