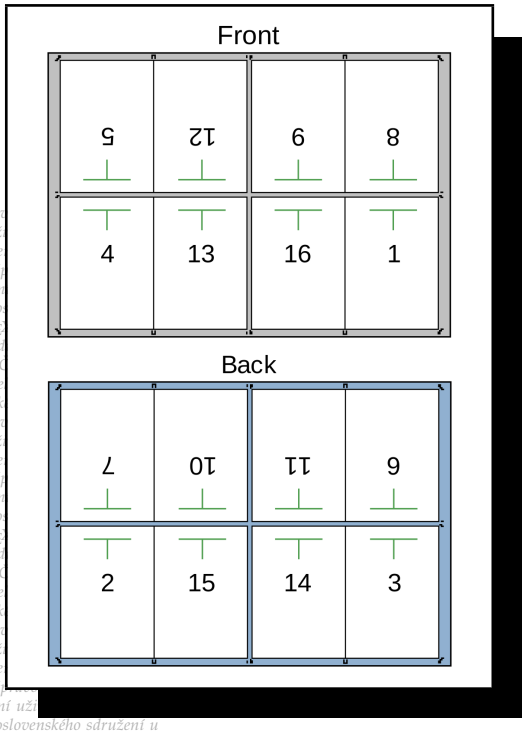
CSTUG



Zpravodaj Československého sdružení uživatelů TEXu Zprav
sdružení uživatelů TEXu Zpravodaj Československého sdružení už
j Československého sdružení uživatelů TEXu Zpravodaj Českoslove
elů TEXu Zpravodaj Československého sdružení uživatelů TEXu Zp
ého sdružení uživatelů TEXu Zpravodaj Československého sdružen
vodaj Československého sdružení uživatelů TEXu Zpravodaj Česko
živatelů TEXu Zpravodaj Československého sdružení uživatelů TE
venského sdružení uživatelů TEXu Zpravodaj Československého sd
Zpravodaj Československého sdružení uživatelů TEXu Zpravodaj
ení uživatelů TEXu Zpravodaj Československého sdružení uživate
skoslovenského sdružení uživatelů TEXu Zpravodaj Československ
TEXu Zpravodaj Československého sdružení uživatelů TEXu Zprav
sdružení uživatelů TEXu Zpravodaj Československého sdružení už
j Československého sdružení uživatelů TEXu Zpravodaj Českoslove
elů TEXu Zpravodaj Československého sdružení uživatelů TEXu Zp
ého sdružení uživatelů TEXu Zpravodaj Československého sdružen
vodaj Československého sdružení uživatelů TEXu Zpravodaj Česko
živatelů TEXu Zpravodaj Československého sdružení uživatelů TE
venského sdružení uživatelů TEXu Zpravodaj Československého sd
Zpravodaj Československého sdružení uživatelů TEXu Zpravodaj
ení uživatelů TEXu Zpravodaj Československého sdružení uživate
skoslovenského sdružení uživatelů TEXu Zpravodaj Československ
TEXu Zpravodaj Československého sdružení uživatelů TEXu Zprav
sdružení uživatelů TEXu Zpravodaj Československého sdružení už
j Československého sdružení uživatelů TEXu Zpravodaj Českoslove
elů TEXu Zpravodaj Československého sdružení uživatelů TEXu Zp
ého sdružení uživatelů TEXu Zpravodaj Československého sdružení už
vodaj Československého sdružení uživatelů TEXu Zpravodaj Československého sdružení u
živatelů TEXu Zpravodaj Československého sdružení uživatelů TEXu Zpravodaj Českoslo
venského sdružení uživatelů TEXu Zpravodaj Československého sdružení uživatelů TEXu
Zpravodaj Československého sdružení uživatelů TEXu Zpravodaj Československého sdruž
ení uživatelů TEXu Zpravodaj Československého sdružení uživatelů TEXu Zpravodaj Če
skoslovenského sdružení uživatelů TEXu Zpravodaj Československého sdružení uživatelů
TEXu Zpravodaj Československého sdružení uživatelů TEXu Zpravodaj Československého
sdružení uživatelů TEXu Zpravodaj Československého sdružení uživatelů TEXu Zpravoda

ZPRAVODAJ

Československého sdružení uživatelů TEXu

ení uživatelů TEXu Zpravodaj Československého sdružení uživatelů TEXu Zpravodaj Če
skoslovenského sdružení uživatelů TEXu Zpravodaj Československého sdružení uživatelů
elů TEXu Zpravodaj Československého sdružení uživatelů TEXu Zpravodaj Československ
ého sdružení uživatelů TEXu Zpravodaj Československého sdružení uživatelů TEXu Zpra
vodaj Československého sdružení uživatelů TEXu Zpravodaj Československého sdružení u
živatelů TEXu Zpravodaj Československého sdružení uživatelů TEXu Zpravodaj Českoslo
venského sdružení uživatelů TEXu Zpravodaj Československého sdružení uživatelů TEXu
Zpravodaj Československého sdružení uživatelů TEXu Zpravodaj Československého sdruž
ení uživatelů TEXu Zpravodaj Československého sdružení uživatelů TEXu Zpravodaj Če
skoslovenského sdružení uživatelů TEXu Zpravodaj Československého sdružení uživatelů
TEXu Zpravodaj Československého sdružení uživatelů TEXu Zpravodaj Československého
sdružení uživatelů TEXu Zpravodaj Československého sdružení uživatelů TEXu Zpravoda
j Československého sdružení uživatelů TEXu Zpravodaj Československého sdružení uživat
elů TEXu Zpravodaj Československého sdružení uživatelů TEXu Zpravodaj Československ
ého sdružení uživatelů TEXu Zpravodaj Československého sdružení uživatelů TEXu Zpra

1–4

2022

ISSN 1211-6661 Tištěná verze

ISSN 1213-8185 Online verze

Ročník 32

OBSAH

Petr Sojka: Úvodník 1

Petr Olšák: `cropmarks.tex` – makra na tvorbu ořezových značek 4

Tereza Vrabcová: Digitální archivace Zpravodaje CSTUGu 11

Denis Roegel: Romantika v METAPOSTu po francouzsku: líbající se kružnice 18

Vít Novotný: Vysokoúrovňové jazyky pro TEX 35

Max Chernoff: Automatically Removing Widows and Orphans with lua-widow-control 49

Peter Wilson: Mělo by to fungovat XII 77

Zpravodaj Československého sdružení uživatelů TEXu je vydáván v tištěné podobě a distribuován zdarma členům sdružení. Vydaná čísla Zpravodaje v elektronické podobě (PDF) jsou bezodkladně veřejně vystavena na webové adrese `https://www.cstug.cz/`.

Své příspěvky do Zpravodaje můžete zasílat v elektronické podobě, nejlépe jako jeden archivní soubor (`.zip, .arj, .tar.gz`), na e-mailovou adresu `bulletin@cstug.cz`. Nezapomeňte přiložit všechny soubory, které dokument načítá (s výjimkou standardních součástí TEX Live).

ISSN 1211-6661 (tištěná verze)
ISSN 1213-8185 (online verze)

Úvodník

Petr Sojka

Milé čtenářky a čtenáři, TEXistky a TEXisté!

Je mi potěšením vám představit články tohoto čísla, a seznámit vás s informacemi ze světa našeho oblíbeného sázecího systému.

První článek čísla na straně 4 z pera kolegy Olšáka připomíná téma, před kterým stál každý, kdo někdy dělal předtiskovou přípravu tiskovin: téma ořezových značek a archové montáže. První makra pro ořezové značky na archívu CTAN pro plain TEX jsou z let 1992 [1] a 1994 [2]. Že je problematika živá a potřebná ukazuje i makrobalík `zwpagelayout` [3]. Článek zmiňuje i další makrobalíčky vytvořené nejen pro „intelektuální rozptýlení“ autora, ale i jako „námět pro hlubší rozpracování“ TEXovými programátory.

Článek studentky Fakulty informatiky Masarykovy univerzity Terezy Vrabcové na straně 11 je záznamem její přednášky na jarním přednáškovém odpoledni CSTUG [4]: máme úplný digitální archiv celé historie našeho Zpravodaje od roku 1990, v digitální matematické knihovně dml.cz! Včetně metadat obsahujících bibliografické záznamy, včetně přidělených persistentních identifikátorů DOI pro jednoznačné citování. To umožňuje jednak dohledávat publikace téměř dvou stovek autorů publikujících ve Zpravodaji, ale také hledat souvislosti a podobné články v Evropské digitální matematické knihovně eudml.org, kam je Zpravodaj dále exportován a archivován. Um celých generací českých a slovenských TEXistů je tak zachován!

Mohou se kružnice líbat? V libovolné pozici, ve třech a rekurzivně? To je téma článku Denise Roegela na straně 18, který se věnuje METAPOSTu a implementaci nedávno objevených možností sestrojení Soddyho kružnice v něm. Geometrovo oko nad deseti obrázky v článku zaplesá, a doufám, že zaplesá i vaše nad ukázanými možnostmi METAPOSTu.

TEX s výhodou používají programátoři jako Vítek Novotný, který se na jarním přednáškovém odpoledni CSTUG [4] a na straně 35 v tomto čísle podělil o svůj pohled na možnosti vysokoúrovňového programování v TEXovém ekosystému. Nic vám neříká LuaMetaTEX, expl3, YAML nebo LyLuaTEX? Chcete využít svých informatických schopností přepínání mezi různými úrovněmi abstrakce a oslovuje vás výzva Donalda Knutha v TEXbooku *“Go forth now and create masterpieces of the publishing art!”*? Pak je článek právě pro vás!

Kvalita algoritmu odstavcového zlomu, kdy TEX řeší speciální variantu NP-úplného optimalizačního problému, byla vždy klíčovou výhodou sazby prováděné TEXem. Kromě mikrotypografických rozšíření, které typografické komunitě přinesl pionýrsky Hàn Thế Thành v pdfTEXu, je podstatné *automatizované* řešení

DOI: 10.5300/2022-1-4/1

nežádoucích parchantů, tedy vdov a sirotků. To je téma článku Maxe Chernoffa na straně 49 postaveného na možnostech programovacího jazyka Lua dostupného v LuaTEXu. Další potěšení pro programátory a zároveň typografy mezi námi!

Číslo rámuje překlad dvanáctého pokračování řady praktických doporučení Petera Wilsona *Glisterings* na straně 77 věnovaný rámečkování.

V den psaní tohoto úvodníku proběhlo již druhé přednáškové odpoledne organizované ČSTUGem. Odezněla, byla nahrána a na webu sdružení bude vystavena čtvrtá a pátá letošní přednáška [5] objednaná sdružením, po třech jarních přednáškách [4]. V našich krajinách se děje mnoho důležitého vývoje pro československou i celosvětovou TEXovou komunitu, a ČSTUG se snaží tyto aktivity podporovat.

Daří se vydávat tento Zpravodaj, jehož všechny články od roku 1990 se podařilo s přispěním sdružení digitalizovat. Po doplnění a kontrolách metadat včetně bibliografie a přidělení DOI u CrossRef jsou vystaveny a oindexovány v české digitální matematické knihovně dml.cz. Doufám, že s podporou sazby příspěvků šablonou na Overleaf, redakční rady a technické redakce je publikování v časopise již nyní potěšením a svátkem ☺.

Valné shromáždění na návrh výboru po více než *dvou dekádách*[1] [7] navýšení příspěvků pro kolektivní členy od roku 2023 *mírně* zvedlo členské příspěvky takto:

Typ členství	příspěvek
Individuální členství základní	400 Kč
Individuální členství slevněné (studenti, důchodci)	250 Kč
Kolektivní členství	2500 Kč
Kolektivní členství slevněné (střední školy)	750 Kč

Tato výše členských příspěvků umožní aspoň částečně pokrýt zvýšené výdaje dané dvěma dekádami navyšování nákladů spojených s chodem sdružení, které byly doposud kryty z finančních rezerv sdružení a dobrovolnickou činností několika nadšenců. Umožní to rozdmýchat naději, že dosavadní poměrně minimalistický chod sdružení bude růst a rozšiřující se benefity vyplývající z členství povedou následně k růstu členské základny a komfortu uživatelů TEXu.

Členstvím ve sdružení podporujete:

- budování komunity uživatelů TEXu: Zpravodaj (členství v CrossRef, web);
- péči o infrastrukturu: archívy CTAN, distribuce TEXlive, diskusní listy a skupiny;
- TEXové projekty (TEXGyre, československé vzory dělení,...).

Platba kolektivního členství ČSTUGu v TUG nám přináší mnohá dobrodiní:

[1] Opomíjíme zde navýšení příspěvku pro kolektivní členy o 150 Kč v roce 2005 [6].

- Podporujeme dobrou věc a další vývoj kvalitní typografie systémy TEX a METAFONT.
- Přinášíme osm čísel TUGboatu do knihoven v Československu.
- Členové sdružení uživatelů CSTUG mohou kolektivním členstvím CSTUGu čerpat i výhody členství v TUGu jako slevy na konferenčních poplatcích, fontech, knihách, viz `https://tug.org/aims_ben.html`.
  Pro členy připravujeme federovaný přístup i k nejnovějším elektronickým číslům časopisu TUGboat.

Závěrem chci poděkovat aktivním TEXistům, autorům, členům sdružení uživatelů TEXu. Bez nich, převážně členů CSTUGu a TUGu a jejich „labour of love“ by to nebylo možné.

Odkazy

1. UNGAR, Peter. *cropmark-pu: Cropmark macros for Plain TEX* [online]. CTAN, 1992-02-10 [vid. 2022-11-12]. Dostupné z: `https://www.ctan.org/pkg/cropmark-pu`.
2. TAYLOR, Philip. *cropmrks: Add crop marks to a Plain TEX document* [online]. CTAN, 1994 [vid. 2022-11-12]. Dostupné z: `https://www.ctan.org/pkg/cropmrks`.
3. WAGNER, Zdeněk. *zwpagelayout: Page layout and crop-marks* [online]. CTAN, 2022-04-18 [vid. 2022-11-12]. Dostupné z: `https://www.ctan.org/pkg/zwpagelayout`.
4. *Valná hromada CSTUG 2021* [online]. 2022-04-01. [vid. 2022-11-15]. Dostupné z: `https://www.cstug.cz/informace/zpravy/2022-04-01-valna-hromada-2022/`.
5. *Valná hromada CSTUG 2022* [online]. 2022-10-24. [vid. 2022-11-17]. Dostupné z: `https://www.cstug.cz/informace/zpravy/2022-10-24-valna-hromada-2022/`.
6. KUBEN, Jaromír. Úvodník. 2005, roč. 15, s. 1–2. ISSN 1211-6661. Dostupné také z: `https://dml.cz/handle/10338.dmlcz/149977`.
7. *Členství v CSTUGu* [online]. 2001-03-27. [vid. 2001-04-29]. Dostupné z: `http://www.cstug.cz/clenstvi/`. Vizte např. webový archiv web.archive.org.

Summary: Introductory Word

Go forth and participate in CSTUG to make the bright future of TEX & Friends a reality! *You can!*

Masarykova univerzita, Fakulta informatiky, Botanická 68a, 602 00 Brno
sojka@fi.muni.cz

cropmarks.tex – makra na tvorbu ořezových značek

Petr Olšák

Existuje jistě celá řada maker umožňujících přidávat do dokumentu ořezové značky. Zde prezentované nové řešení je postaveno na makrech plainTEXu a umožní přidat ořezové značky do jakéhokoli PDF dokumentu nezávisle na tom, jakým způsobem byl tento dokument vytvořen. Vytvoříme tedy nejprve v něčem (v LaTEXu, v OpenOffice, v OpTEXu, ...) dokument a pak použijeme makra z `cropmarks.tex` k přidání ořezových značek.

Soubor `cropmarks.tex` je součástí balíčku `olsak-misc` [1]. Ten obsahuje další užitečná makra pro plainTEX. Na konci tohoto článku je přidána stručná informace i o těchto dalších nástrojích. Balíček `olsak-misc` je sice součástí TEXlive, ale je tam bohužel umístěn poněkud nešťastně v adresáři pro dokumentaci. Než tedy začnete experimentovat, umístěte si soubor `cropmarks.tex` někam, kde jej Váš TEX najde.

Klíčová slova: plainTEX, ořezové značky

Obvyklý způsob použití

Předpokládejme, že máme PDF soubor `ukazka.pdf` a chceme mu přidat ořezové značky. Vytvoříme soubor (například se jménem `ukazka-crop.tex`).

```
\input cropmarks
\def\document{ukazka} % jméno PDF souboru bez přípony .pdf
\docropmarks          % vytvoří ořezové značky
```

a dále můžeme spustit příkaz `pdftex ukazka-crop` nebo `optex ukazka-crop`. Příkaz přečte původní dokument stránku po stránce a přidá k němu ořezové značky. Není nutné psát závěrečné `\bye`, protože činnost TEXu ukončí makro `\docropmarks`. Vstupní PDF dokument musí mít správně nastaveny rozměry papíru tak, jak předpokládáme, že to bude vypadat po provedení ořezu. Jde tedy o konečné rozměry papíru dokumentu.[1]

V souboru `crop-dokument.tex` můžete mít také konfigurační údaje:

```
\input cropmarks
\def\document{jmeno} % jméno vstupního PDF souboru
\def\info{}          % text přidaný do každého horního okraje
\hmargin=10mm        % velikost přidaného horního+dolního okraje
```

[1] Výjimka je možná, okraje vstupního dokumentu mohou být případně větší (chceme-li mít např. obrázky na spadávku). Pak je třeba nastavit kladný parametr `\overlap`.

 DOI: 10.5300/2022-1-4/4

```
\vmargin=10mm % velikost přidaného pravého+levého okraje
\mlenght=5mm  % délka čar ořezových značek, jsou v přidaném okraji
\gap=2mm      % vzdálenost čar od místa řezu
\mthick=.2pt  % šířka čar ořezových značek
\overlap=0mm  % překryv okraje dokumentu do přidaného okraje
\hmiddle=1    % počet středových značek v levém+pravém okraji
\vmiddle=1    % počet středových značek v horním+dolním okraji
\docropmarks  % čte \document.pdf a vytvoří ořezové značky
```

Ve výše uvedené ukázce jsou přímo uvedeny implicitní hodnoty jednotlivých parametrů. Máte možnost si je ve svém souboru změnit podle potřeby.

Údaj `\info` se přidá do horního okraje, který podléhá ořezu, a opakuje se na každém archu. Můžete tam třeba mít:

```
\def\info{\tt [\document.pdf] -- \today\ -- sheet: \folio}
```

Pak se v okraji vytiskne jméno dokumentu, datum a číslo archu.

Nemusíte vytvářet nový soubor. Ořezové značky lze přidat také přímo vhodně voleným příkazem z příkazového řádku, například:

```
pdftex cropmarks '\def\document{jmeno} \docropmarks'
```

Můžete mít konfigurační údaje třeba v souboru `cropmarks.cfg` a pak přímý příkaz na vytvoření ořezových značek zní:

```
pdftex cropmarks \
   '\input cropmarks.cfg \def\document{name} \docropmarks'
```

Ořezové značky po archové montáži

Pokud sestavujeme více stránek do archů, pak chceme mít ořezové značky v okrajích celých archů a využijeme „středové značky“, jejichž počet je dán pomocí `\hmiddle` a `\vmiddle`. Ty nyní budou značit místa ohybu. Je tedy potřeba nejprve provést jiným softwarem archovou montáž a na výsledné PDF z této fáze přípravy dokumentu teprve aplikovat `cropmarks`.

Ukážeme si to na dvou příkladech. První (jednodušší) bude mít dvě stránky na archu a druhý (komplikovanější) vytvoří ořezové značky pro tzv. „impozici“ osmi stránek na archu, což je metoda rozmístění stránek, u níž se předpokládá, že se nakonec na oboustranně tištěné archy (každý arch pak nese dohromady 16 stránek) použije automatický skládací a řezací stroj, který z toho udělá svazečky po šestnácti stránkách, viz [2, 3].

V obou příkladech předpokládáme, že již máme dokument vytvořený nějakým softwarem. Podstatné je, že dokument obsahuje jednotlivé stránky se správně nastavenými konečnými rozměry papíru, jak je chceme vidět po ořezu.

V prvním příkladě budeme mít na začátku dokument `mytext.pdf`. Navíc předpokládáme, že stránky jsou formátu mírně menšího než A5, aby se vešly dvě

vedle sebe na archy A4 a abychom nakonec viděli ořezové značky, když bychom to chtěli tisknout. Uděláme si brožurku, tj. uspořádáme všechny stránky dokumentu na archy tak, aby po vytištění na duplexové tiskárně stačilo celý vytištěný svazek přehnout v půli, oříznout, a bylo hotovo.

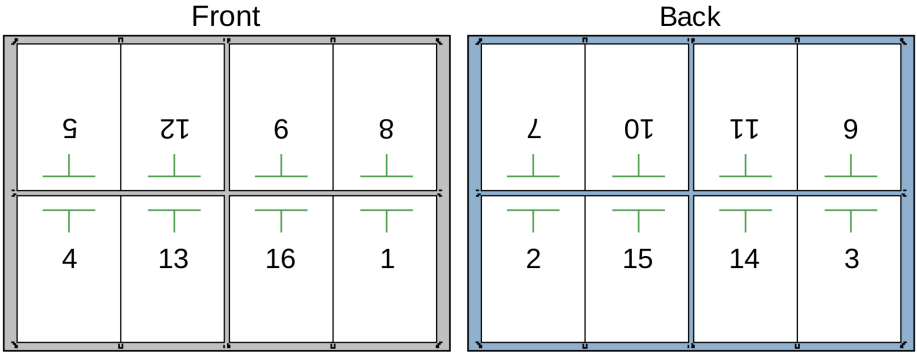
Provedeme tedy sestavení archů (například pomocí programu `pdfjam` [4], `paperjam` [5] nebo pomocí OpTEX triku 0089 [6] nebo makrem `booklet.tex` z balíčku `olsak-misc` [1]. Tím získáme nový dokument, nazveme ho třeba `mytext-archy.pdf`. Nakonec si připravíme soubor `mytext-crop.tex` s tímto obsahem:

```
\input cropmarks
\def\document{mytext-archy}
\hmiddle=1
\vmiddle=0
\docropmarks
```

a dostáváme konečný výsledek `mytext-crop.pdf`. Důležité je, že ty středové značky nejsou složeny ze dvou běžných ořezových značek a tedy nemají svou druhou čárku rovnoběžnou s okrajem. Kdyby měly, tak takové čárky svou polovinou tloušťky (při nepřesném ořezu třeba i více) zasahují už do čisté stránky po ořezu, což je nežádoucí.

Pokud to budeme chtít vytisknout na duplexové tiskárně, je potřeba zvolit médium větší než stránky dokumentu `mytext-crop.pdf` včetně ořezových značek a dále nastavit v ovladači tisku požadavek zákazu změny velikosti dokumentu a centrování dokumentu na použité médium.

Pusťme se do druhého příkladu. **Impozice** je daleko zábavnější. Jednotlivé archy vypadají takto:



Je třeba každý arch zvlášť nejprve přehnout podle svislé osy, pak podle vodorovné a nakonec ještě podle svislé. Poslední ohyb vytváří hřbet svazečku, ostatní tři strany je nutno oříznout, aby se v tom vůbec dalo listovat. Můžete si

udělat zmenšený model archu z nějakého menšího papíru, napsat na jeho rub a líc čísla podle obrázku a dále to poskládat a promyslet si, co se stalo. Všimněte si, že závěrečný ohyb se provede podle první a třetí svislé linky v obrázku a ten nesmí mít ořez (celé by se to rozpadlo). Naopak všechny ostatní linky v obrázku mají nakreslený prostor ořezu, tj. jednotlivé stránky na sebe v těch místech těsně nenavazují. Provede se tam v ose prostoru ořezu přehyb a pak se to ořízne.

Z internetu jsem si stáhl PDF soubor `bnemoc.pdf` [7], který byl vytvořen softwarem v ničem nesouvisejícím s TEXem. Ale to nevadí. Prvním úkolem je archová montáž. Asi bych po delším studiu manuálových stránek přišel na to, jak to provést například v programu `paperjam` [5], ale mě bavilo daleko víc si tuto úlohu vyřešit v OpTEXu, což jsem zveřejnil v triku 0088 [8]. Po zkopírování příslušných maker z tohoto triku do souboru `trick-0088.tex` pak stačí vytvořit soubor `bnemoc-archy.tex` s tímto obsahem:

```
\input trick-0088
\def\document{bnemoc}
\vspacing=18mm  % Mezera mezi dvěma archovými řádky je 18mm
\sheet 1 {
  [ v5 | v12 |14mm|  v9 | v8 ]  % prostřední mezera je 14mm
  [ p4 | p13 |14mm| p16 | p1 ]
}
\sheet 2 {
  [ v7 | v10 |14mm| v11 | v6 ]
  [ p2 | p15 |14mm| p14 | p3 ]
}
\printsheets
```

Pomocí symbolu `v` jsou značeny stránky otočené hlavou dolů, pomocí symbolu `p` jsou pak neotočené stránky. `\sheet 1` je deklarace lícové strany archu a `\sheet 2` rubové. Má-li dokument více stránek, než je deklarováno, další stránky pokračují na dalších arších modulo maximální číslo deklarované stránky, zde tedy modulo 16.

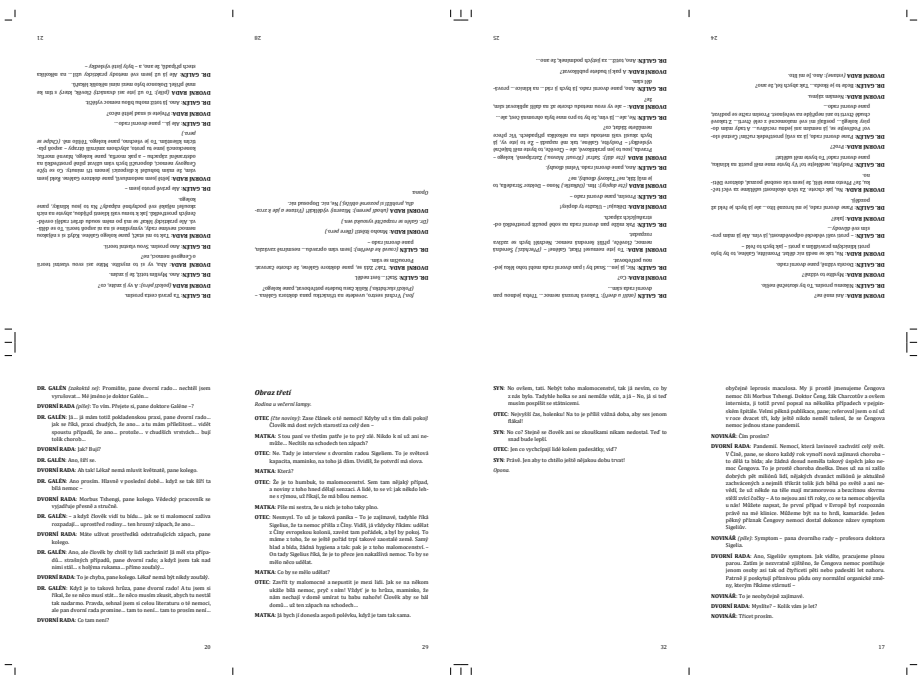
Po spuštění `optex bnemoc-archy` vzniká dokument `bnemoc-archy.pdf` s rozmístěnými stránkami po arších. Pro první ohyb dle svislé osy je vymezena mezera 14 mm, takže po přehybu se má v odpovídajícím místě odříznout sedmimilimetrový okraj. Chceme tam mít nejen značku pro přehyb (přesně uprostřed), ale i dvě značky pro ořez. Tedy lidově řečeno tam budou tři svislé čárky vedle sebe ve vzdálenosti 7 mm jedna od druhé. V ohybu podél vodorovné osy máme 18 mm místa, tedy odřezáváme po ohybu 9 mm. Odpovídající značka bude mít tři vodorovné čárky nad sebou ve vzdálenosti 9 mm. Vytvořím si tedy soubor `bnemoc-crop.tex` s následujícím obsahem:

```
\input cropmarks
\def\document{bnemoc-archy}
\hmiddle=1 \middlecrop h1: {18mm}
```

```
\vmiddle=3 \middlecrop v2: {14mm}
\docropmarks
```

V ukázce je použita deklarace `\middlecrop`, která vymezuje, které středové značky budou „široké“ (tj. budou se skládat ze tří čáreček) a jak budou široké (od první čárky k poslední). První značka v levém a pravém okraji bude široká 18 mm a dále druhá značka v horním a dolním okraji bude široká 14 mm. Ostatní nedeklarované středové značky zůstávají „úzké“, tedy sestávají z jediné čárky.

Zpracujeme-li soubor `bnemoc-crop.tex` příkazem `optex bnemoc-crop` nebo `pdftex bnemoc-crop`, dostáváme dokument se střídavými lícovými a rubovými stranami archů včetně ořezových značek. Pro ilustraci zmenšeninu lícové strany druhého archu vidíte na následujícím obrázku. Můžete si vzít lupu a začít zkoumat, co tam je. Text tohoto dokumentu je zřejmě všeobecně známý a navíc v dnešní době znovu silně aktuální, ačkoli byl napsán před 85 lety.



Další makra z balíčku `olsak-misc`

Soubory s plainTEXovými makry z balíčku `olsak-misc` mají skoro vždy na svém konci za příkazem `\endinput` podrobnější dokumentaci, v níž se píše, k čemu

slouží a jak se používají. V této sekci proto uvedu jen stručné shrnutí, zájemci se pak mohou podívat přímo do jednotlivých souborů.

Tyto soubory vznikaly v různých časových etapách mého přátelství s TEXem. V závorce vedle názvu souboru proto uvádím také rok vzniku souboru.

Poznamenávám, že za přímo použitelné v praxi považuji jen některé soubory, které jsou uvedeny v prvním úseku následujícího seznamu.

- `booklet.tex` (2016) umožní stránkovou montáž do svazečku, jak je popsáno v prvním příkladu v předchozí sekci.
- `cropmarks.tex` (2022) přidává ořezové značky. Hlavně o tom je tento článek.
- `qrcode.tex` (2015) generuje QR kódy. Funguje úplně stejně jako LATEXový balíček `qrcode.sty`, jen s méně chybami a hlavně funguje i v plainTEXu.
- `scanbase.tex` (2002) je parser na textové výstupy z databází MySQL či podobných.
- `scancsv.tex` (2005) čte soubory v CSV formátu, tj. např. výstupy z programů Excel a podobných.
- `xmlparser.tex` (2016) umožní transformovat XML vstup na soubor se stejně vymezenou strukturou, ale s TEXovsky přátelštější syntaxí.

Následující úsek obsahuje soubory, které vznikly spíše pro intelektuální rozptýlení nebo jako námět pro další hlubší rozpracování programátorem maker.

- `cnv.tex` (2005) je makro na konverzi token listů v TEXu dle deklarovaných pravidel (konverzní tabulky).
- `cnv-pu.tex` (2005) je konverzní tabulka pro `cnv.tex` umožňující dávat Unicode texty do PDF záložek (tam se totiž používá Unicode octal strings).
- `cnv-word.tex` (2005) je příklad na užití `cnv.tex` pro konverzi úseků slov.
- `eparam.tex` (2014) expanduje parametr v době jeho čtení a končí na explicitní `}` nebo na deklarované sekvenci (například na `\stop`).
- `fun-coffee.tex` (2015) Vytvoření kávové skvrny v dokumentu. Je to reakce na internetovou diskusi, kde totéž někdo udělal pomocí Ti*k*Z. Chtěl jsem předvést, že bez Ti*k*Z to jde taky.
- `openclose.tex` (2014) Text mezi `\Open` a `\Close` je opraven tak, aby to byl vždy balancovaný text.
- `seplist.tex` (2014) umožní použít makro s inteligentně separovaným parametrem: čtení parametru se zastaví na prvním separátoru z deklarovaného seznamu separátorů. Takže parametr může mít více než jediný separátor.

Odkazy

1. OLŠÁK, Petr. *olsak-misc: Collection of plain TEX macros* [online]. CTAN, 2022-08 [vid. 2022-11-10]. Dostupné z: `https://ctan.org/pkg/olsak-misc`.
2. *Imposition* [online]. Wikipedia: The Free Encyclopedia, 2022-09-16 [vid. 2022-11-10]. Dostupné z: `https://en.wikipedia.org/wiki/Imposition`.

3. ANN. *Tiskový arch, archová montáž* [online]. Studijni-svet.cz [vid. 2022-11-10]. Dostupné z: `https://studijni-svet.cz/tiskovy-arch-archova-montaz/`.
4. THOMAS, Reuben; FIRTH, David. *pdfjam: Shell scripts interfacing to pdfpages* [online]. CTAN [vid. 2022-11-10]. Dostupné z: `https://ctan.org/pkg/pdfjam`.
5. MAREŠ, Martin. *PaperJam PDF Transformer* [online]. 2022-08-21. [vid. 2022-11-10]. Dostupné z: `http://mj.ucw.cz/sw/paperjam/`.
6. OLŠÁK, Petr. *Printing booklets* [online]. 2022-06-17. [vid. 2022-11-10]. Dostupné z: `https://petr.olsak.net/optex/optex-tricks.html#booklet`.
7. ČAPEK, Karel. *Bílá nemoc*. 1. vyd. František Borový, 1937. Dostupné také z: `https://cs.wikisource.org/wiki/B%C3%ADl%C3%A1_nemoc`.
8. OLŠÁK, Petr. *More pages on single sheet* [online]. 2022-06-14. [vid. 2022-11-10]. Dostupné z: `https://petr.olsak.net/optex/optex-tricks.html#frontsheet`.

Summary: `cropmarks.tex` – Macros for Creating Crop Marks

The package `cropmarks.tex` from `olsak-misc` [1] bundle of packages is presented here. It enables to add configurable crop marks to any PDF file (irrelevant what software created it). The macro is based on Plain TEX and works in OpTEX too. The special crop marks for impositions [2] are available too and this example is shown in the article in detail.

The summary about other `olsak-misc` macro files is added in the last section here. Moreover, the macros are documented at the end of each file.

Keywords: plain TEX, crop marks

Petr Olšák
`http://petr.olsak.net`

Digitální archivace Zpravodaje CSTUGu

Tereza Vrabcová

Je-li člověk hrdý na svůj výtvor, chce jej sdílet s co největším množstvím lidí. A tak nemůže být divu, že CSTUG sáhl po příležitosti digitálně archivovat Zpravodaj a zpřístupnit své články širší odborné veřejnosti v rámci České digitální matematické knihovny (DML-CZ). Tento článek představuje technické řešení a výsledky této digitalizace společně se statistickými zajímavostmi, na které se přišlo v jejím průběhu. Jedná se o interpretaci autorčiny přednášky na valném shromáždění CSTUGu 14. května 2022 [1].

Klíčová slova: digitalizace, DML-CZ, statistika

Zpravodaj Československého sdružení uživatelů TEXu (CSTUG) tímto vydáním počíná 32. ročník své publikace. Během této doby v něm bylo publikováno velké množství článků pojednávajících o složitostech a krásách TEXu a mnoha dalších nadstavbách a odvozeninách tohoto programu pro počítačovou sazbu, kolem kterého bylo toto sdružení utvořeno. Mnoho hodin práce bylo věnováno jejich napsání, editaci a přípravě pro jejich publikování. Je tedy pochopitelné, že během tohoto procesu vyvstala touha zachovat informace a detaily uložené v těchto stránkách, stejně tak, jako pokusit se rozšířit o nich povědomí v rámci širší odborné veřejnosti našich dvou zemí.

Česká digitální matematická knihovna (DML-CZ) je evropský projekt (2005 až 2009) zajišťující akvizici, digitalizaci, archivaci a vyhledávání nad československými matematickými texty. V roce 2019 tým DML-CZ nabídl vzhledem k významu TEXu pro matematiku Zpravodaj CSTUG digitálně archivovat a zpřístupnit v rámci knihovny, která usnadňuje dosažitelnost těchto článků pro širší veřejnost. Na valném shromáždění 2020 bylo Vítem Novotným v prezentaci *Konverze Zpravodaje pro Českou digitální knihovnu* [2] navrženo řešení tohoto problému.

Průběh archivace

Nejprve bylo potřeba připravit články ve formátu přijatelném pro DML-CZ. Každé číslo bylo rozděleno na jednotlivé články, pro které pak byly připraveny následující dokumenty:

- `meta.xml`: Tento soubor obsahuje informace o daném článku: jméno článku a autora, jazyk článku, typ článku, rozsah stránek, abstrakt a mnoho dalších.
- `source.pdf`: Tento soubor obsahuje článek jako takový.
- `references.pdf`: Tento soubor obsahuje bibliografické odkazy článku. Cílem bylo vytvoření co nejbohatších metadat pro usnadnění vyhledávání a jednoznačnosti odkazů.

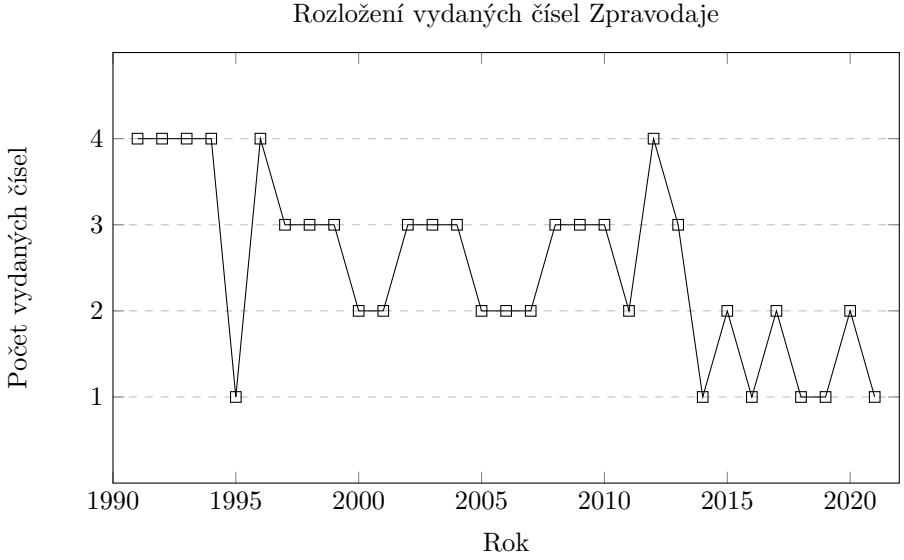
DOI: 10.5300/2022-1-4/11

Ukázky XML souborů je možné si prohlédnout na čtvrté straně mé prezentace z valného shromáždění CSTUGu 14. května 2022 [1]. Výsledky této archivace je již v tuto chvíli možno procházet na stránkách DML-CZ [3].

Statistiky Zpravodaje

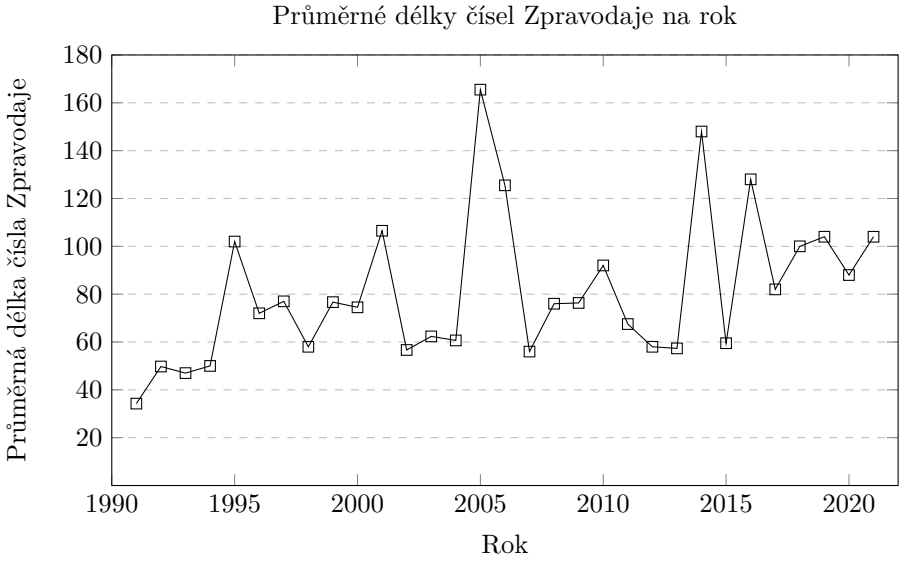
Knihovna DML-CZ na svých stránkách aktuálně nezobrazuje všechna metadata připravená při archivaci tohoto časopisu. Prohlédněme si tedy tato data formou statistik a zajímavostí objevených při archivaci. Pokud nebude stanoveno jinak, grafy v tomto článku zobrazují statistiky odborných článků na rozdíl od grafů v prezentaci z valného shromáždění, které zobrazovaly statistiky všech článků. Zobrazovaná data pocházejí z let 1991–2021.

Během let 1991–2021 bylo ve 31 ročnících celkově vydáno 78 čísel Zpravodaje. Dle mediánu i modu se počet čísel pohyboval na třech číslech každý rok, průměr se blížil spíše ke dvěma a půl (2,52) číslům. V posledních letech se pak počet čísel pohyboval mezi jedním až dvěma čísly.

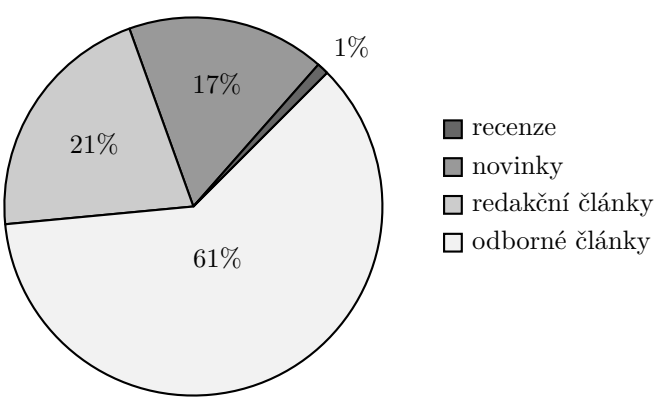


Průměrná délka čísla Zpravodaje naopak v průběhu let rostla. Medián i modus čísla se rovná 64 stránkám, průměrný počet stránek všech čísel pak je 72,55[1]. Nejdelším vydáním je číslo 2–4 z roku 2005 [4] s úctyhodnými 239 stránkami.

[1] Budeme-li pokládat dvojčísla, trojčísla a čtyřčísla za dvě, tři a čtyři separátní čísla, průměrný počet stran je pak 59,57, medián se rovná 57 stránkám a modus zůstává na 64 stránkách.

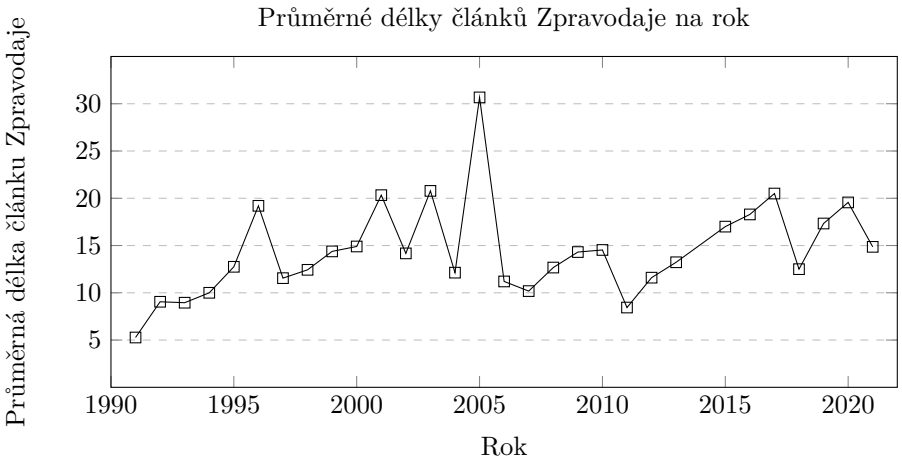


Víme tedy, kolik máme čísel a jak jsou dlouhá, ale co na nás čeká uvnitř? Každý digitalizovaný článek spadá do jedné ze čtyř kategorií – redakční článek, novinky, recenze, či odborný článek. V množství vítězí kategorie odborných článků s více než třemi pětinami z celkového počtu 645 článků (391 článků), na posledním místě se pak nachází kategorie recenze se sedmi články.

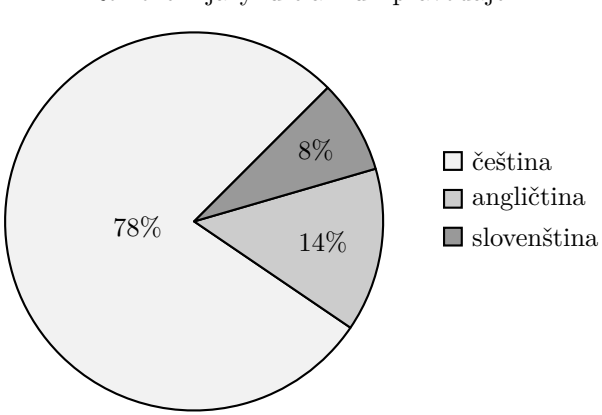


Zaměřme se nyní na kategorii odborných článků. Nejčastější délka jednoho takového článku se pohybuje na sedmi stránkách, hodnota mediánu pak na

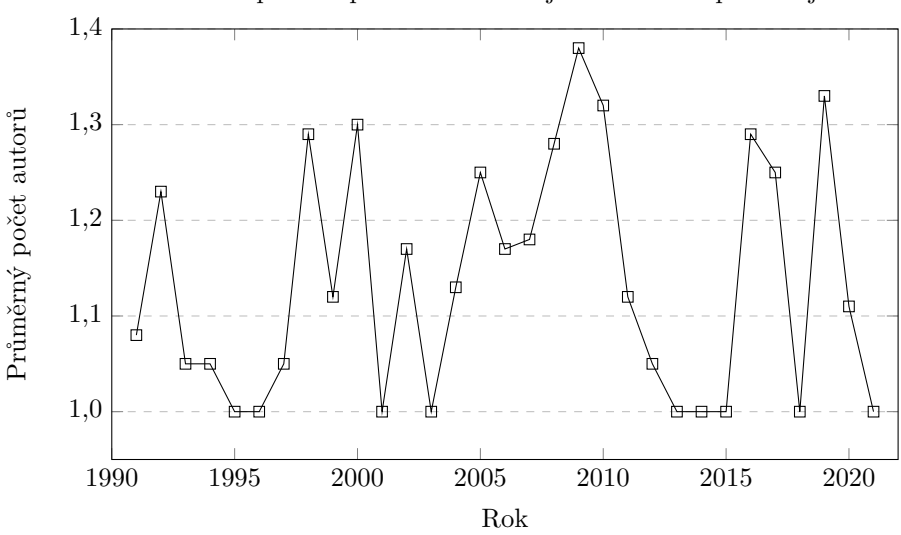
devíti. Pro výpočet střední hodnoty radši odeberme neobvykle dlouhé články, jako například článek publikovaný v roce 2014, *TEX pro pragmatiky* [5] od Petra Olšáka se 148 stránkami, či *CSTUG FAQ* [6] z roku 2005 s 228 stránkami. Pak průměrný článek nabývá délky 9,48 stránky.



Jazykové rozložení článků může být pro některé překvapující. Na prvním místě s nejvyšší četností sedí čeština s více než třemi čtvrtinami článků (307), na druhém angličtina (53) a slovenština přichází na řadu poslední (31).



Na druhou stranu překvapující snad nebude průměrný počet autorů jednoho článku, kde se medián a modus drží na jedničce, průměrná hodnota pak na 1,14.



Na psaní odborných článků ve Zpravodaji se podílelo celkem 180 unikátních osob: 165 mužů a 15 žen.

Bibliografické zajímavosti

Množství bibliografických odkazů využitých v článku závisí na uvážení autora, kvantifikovat je pouze průměrem přes všechny články se zdá nedostatečné. Podívejme se tedy radši na články s nejbohatšími bibliografiemi z našeho výběru:

1. *TEX na školách? Samozřejmě ano! Příklad užití TEXu na Fakultě informatiky Masarykovy university* (2017, 50 odkazů)
2. *Sadzba bibliografie podla normy ISO 690 v systéme LATEX* (2016, 42 odkazů)
3. *E-TEX: Sprievodca budúcimi rozšíreniami TEXu* (1994, 35 odkazů)

Informace ve článcích Zpravodaje nebyly čerpány pouze z těch nejnovějších zdrojů v době jejich psaní, jak je možné vidět při procházení nejstarších citovaných bibliografických odkazů:

1. *The Abacus, in Its Historic and Scientific Aspects* (1886)
2. *On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling* (1900)
3. *A History of Japanese Mathematics* (1914)
4. *Nauka o sazbě obyčejné, tabulkové, matematiky a chemie* (1925)
5. *Tibetan Yoga And Secret Doctrines* (1935)

Z důvodu tematického zaměření Zpravodaje je přirozené, že určité zdroje informací jsou citovány častěji než ostatní. Ukažme si, které odkazy byly citovány nejčastěji napříč odbornými články:

1. *The TEXbook* (30 citací)
2. *TEXbook naruby* (29 citací)
3. *Typografický systém TEX* (16 citací)
4. *The pdfTEX user manual* (14 citací)
5. *TEX: The Program* (11 citací)

Závěr

Díky digitalizaci Zpravodaje CSTUGu je procházení a nalézání článků jednodušší než kdy předtím. Proč tedy nezavítat na stránky DML-CZ a neobjevit polozapomenuté moudrosti z let předešlých? (Samozřejmě nejlépe až po dočtení tohoto čísla.)

Odkazy

1. VRABCOVÁ, Tereza. *Digitální archivace Zpravodaje CSTUG* [online]. 2022. [vid. 2022-09-30]. Dostupné z: `https://www.cstug.cz/informace/zpravy/2022-04-01-valna-hromada-2022/files/vrabcova-dmlcz-2022-05-14.pdf`.
2. NOVOTNÝ, Vít. *Konverze Zpravodaje CSTUGu pro Českou digitální matematickou knihovnu (DML-CZ)* [online]. 2021. [vid. 2022-09-30]. Dostupné z: `https://www.cstug.cz/informace/zpravy/2021-02-06-valne-shromazdeni-2020/files/novotny-konverze-2021-03-06.pdf`.
3. *Zpravodaj Československého sdružení uživatelů TEXu* [online]. Česká digitální matematická knihovna. [vid. 2022-09-30]. Dostupné z: `https://www.dml.cz/handle/10338.dmlcz/148724`.
4. Zpravodaj Československého sdružení uživatelů TEXu [online]. 2005, roč. 15, č. 2–4 [vid. 2022-09-30]. ISSN 1211-6661. Dostupné z: `https://www.dml.cz/handle/10338.dmlcz/149988`.
5. OLŠÁK, Petr. TEX pro pragmatiky. *Zpravodaj Československého sdružení uživatelů TEXu* [online]. 2014, roč. 24, č. 1–4 [vid. 2022-09-30]. ISSN 1211-6661. Dostupné z DOI: `10.5300/2014-1-4/1`.
6. CSTUG FAQ. *Zpravodaj Československého sdružení uživatelů TEXu* [online]. 2005, roč. 15, č. 2–4, s. 94–331 [vid. 2022-09-30]. ISSN 1211-6661. Dostupné z DOI: `10.5300/2005-2-4/94`.

Summary: Digital Archival of the GSTUG Bulletin

When a person is proud of their creation, they want to share it with the largest amount of people possible. Therefore, it cannot be a surprise that GSTUG took the opportunity to digitally archive the GSTUG Bulletin and make its articles accessible to the general public with the help of the Czech Digital Mathematics Library (DML-CZ). This article introduces the technical solution and the results of this digitization, along with some interesting statistics that have been discovered during the process. It is an interpretation of the author's talk at the general assembly of GSTUG on May 14, 2022.

Keywords: digitization, DML-CZ, statistics

Tereza Vrabcová, xvrabcov@fi.muni.cz

Romantika v METAPOSTu po francouzsku: líbající se kružnice

Denis Roegel

Když se kružnice potkají, políbí se. Pokud se líbají tři, další se pokusí přidat a políbit je všechny naráz. V tomto článku se podíváme na tuto situaci podrobně z pohledu METAPOSTu a poradíme jim, jak se mají líbat, v libovolné pozici a velikosti. Naučíme je také líbat se rekurzivně.

Klíčová slova: METAPOST, Apolloniova úloha, navzájem dotýkající se kružnice, vnitřní a vnější Soddyho kružnice, Eppsteinova konstrukce, Apolloniův fraktál

1. Úvod

Apollonios z Pergy (3. století před naším letopočtem) byl řecký geometr, mimo jiné také autor díla *Conica* – Pojednání o kuželosečkách. Je mu připisováno, že jako první použil termíny *elipsa*, *parabola* a *hyperbola*. Jeho kniha *De Tactionibus* (*O dotycích*), citovaná Papposem Alexandrijským, definuje problém tečen jako problém nalezení kružnice dotýkající se třech dalších objektů, v libovolné kombinaci bodů, přímek a kružnic. Apollonius ukázal, jak lze tento problém vyřešit pomocí pravítka a kružítka. Nyní tyto úlohy známe pod označením Apolloniovy úlohy. Pokud jsou ony objekty tři kružnice, existuje až osm různých řešení [2].[1]

Za situace, že se tři kružnice dotýkají zvnějšku, se těchto osm řešení redukuje jen na dva případy, a to na vepsanou (vnitřní) a opsanou (vnější) dotýkající se kružnici, známé jako vnitřní a vnější Soddyho kružnice (viz Obrázek 1).

René Descartes našel jednoduché analytické řešení. Křivosti $e_1 = \frac{1}{r_1}$, $e_2 = \frac{1}{r_2}$, $e_3 = \frac{1}{r_3}$ tří dotýkajících se kružnic jsou ve vztahu ke křivosti e_4 Soddyho kružnice zapsané pomocí rovnice

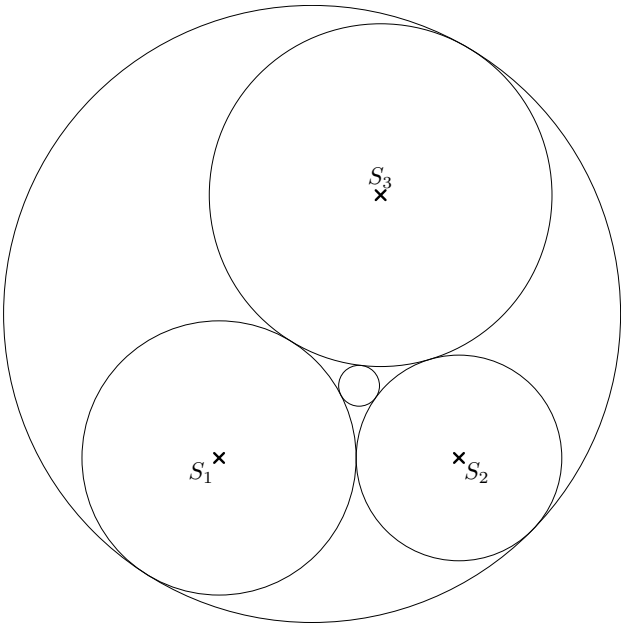
$$2(e_1^2 + e_2^2 + e_3^2 + e_4^2) = (e_1 + e_2 + e_3 + e_4)^2 . \tag{1}$$

V rovnici jsou e_1, e_2 a e_3 zadány, řešení e_4 jsou dvě. Kladné řešení představuje vnitřní Soddyho kružnici a záporné řešení vnější Soddyho kružnici, jejíž poloměr je dopočítán pomocí $-\frac{1}{e_4}$. Analytické řešení může být použito k iterování nákresu, musíme si však dát pozor na přetečení u aritmetických operací. Vnější Soddyho

Z anglického originálu *Kissing Circles: A French Romance in METAPOST* [1] přeložil Pavel Stříž.

[1] Soddyho kružnice buď nezahrnuje žádnou ze tří kružnic (1 řešení), jednu z nich (3 řešení), dvě z nich (3 řešení), nebo všechny tři (1 řešení). Celkem je tedy osm řešení; jejich grafické vyobrazení viz [2, str. 16]. (pozn. překl.)

 DOI: 10.5300/2022-1-4/18



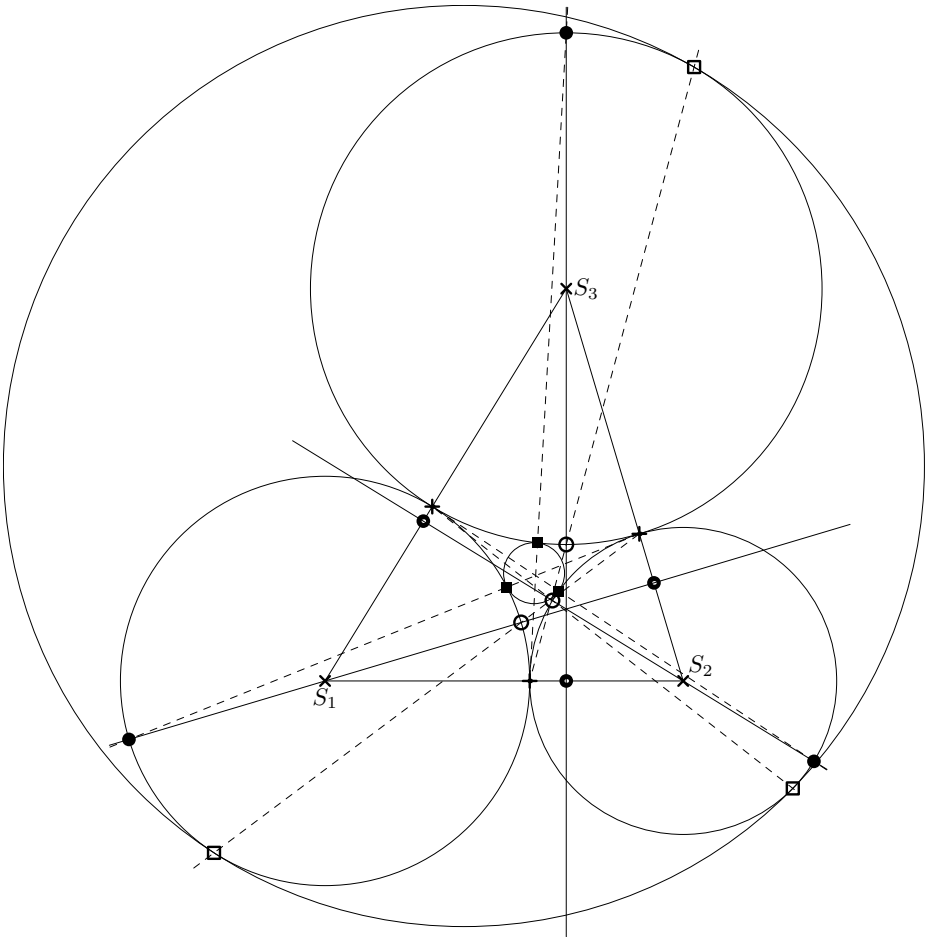
Obrázek 1: Vnitřní a vnější Soddyho kružnice ke kružnicím se středy S_1, S_2 a S_3.

kružnice obsahuje uvnitř další kružnice, a proto má malou hodnotu křivosti. Jak vměstnáváme stále menší a menší kružnice, dostáváme stále větší hodnoty křivosti. METAPOST sám o sobě není příliš vhodný na zpracování příliš malých ani příliš velkých čísel, v takovém případě je výhodnější využít geometrický přístup bez nutnosti výpočtů.

2. Konstrukce Davida Eppsteina

David Eppstein publikoval v roce 2001 nový způsob sestrojení vnitřní a vnější Soddyho kružnice [3]. Naším cílem nebude ověřit, zda je tato konstrukce správná, ale zjistit, jak nejlépe ji lze využít v METAPOSTu, a to zejména co nejobecněji.

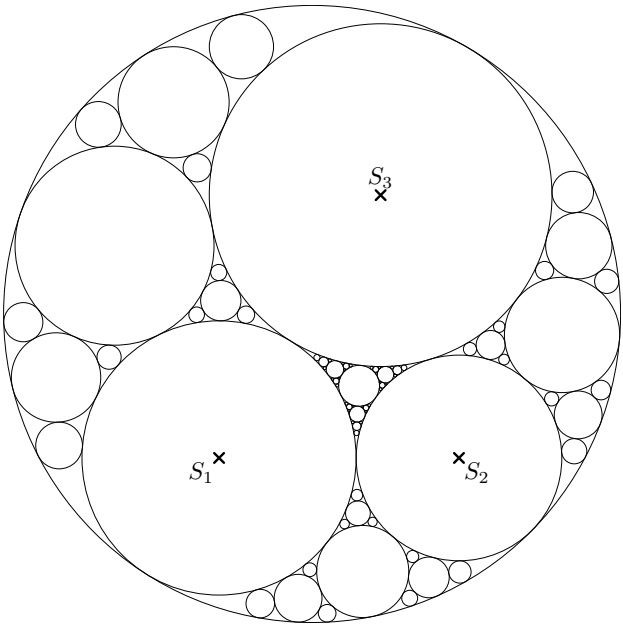
Eppsteinův postup si nyní popíšeme. Zadány máme tři navzájem dotýkající se kružnice se středy S_1, S_2 a S_3 (Obrázek 2). Z každého středu je spuštěna kolmice na protější stranu trojúhelníku. Paty získaných výšek jsou označeny puntíkem s průhlednou tečkou uprostřed (⦿). Tato protažená výška protne kružnici, jejímž středem prochází, na dvou místech. Bližší k průsečíku těchto výšek označíme značkou prstence (○), ty vzdálenější označíme puntíkem (●). Nyní každá z těchto



Obrázek 2: Eppsteinova konstrukce.

dvou značek může být spojena s bodem dotyku zbylých dvou kružnic, který označíme vertikálním křížkem (✚). Přímka procházející přes body označené puntíkem a vertikálním křížkem protne původní kružnice ještě v bodech, které označíme plnými čtverci (■). A tyto tři body označené plnými čtverci tvoří body dotyku vnitřní Soddyho kružnice.

Podobně přímka procházející přes body označené prstencem a vertikálním křížkem protne původní kružnice ještě v bodech, které označíme prázdnými čtverci (□). Tyto tři body tvoří body dotyku vnější Soddyho kružnice.



Obrázek 3: Apolloniův fraktál hloubky 3 s 83 kružnicemi.

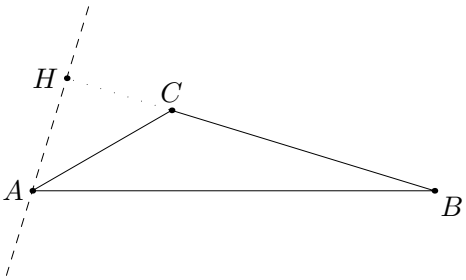
Tento postup lze použít k nalezení vnitřních dotýkajících se kružnic k vnější Soddyho kružnici a například kružnicím se středy S_1 a S_3. Takto lze sestrojit i Apolloniův fraktál (Obrázek 3).

3. Problémy se sestrojením a přípravy v METAPOSTu

Nalezení Soddyho kružnic je poměrně přímočaré, ačkoliv již v této fázi musíme dávat pozor na speciální případy. Skutečné problémy k vyřešení se objeví tehdy, jakmile budeme konstrukci iterovat, jinými slovy ve chvíli, kdy se rozmístění kružnic mění a objevuje se více situací. Hlavním zdrojem potíží je opakování se identických kružnic. Eppsteinovým postupem získáme šest bodů, musíme si však dát velký pozor na to, jak tyto body roztřídíme do dvou skupin po třech. Navíc musíme vyřešit situaci, která trojice bodů patří k té kružnici, kterou si zrovna přejeme nakreslit.

Začneme naši práci přípravou série robustních maker na typické dílčí úlohy. Některé ze zmíněných maker lze snadno využít v jiných programech.

• $A + \overrightarrow{CB}$ rotated 90

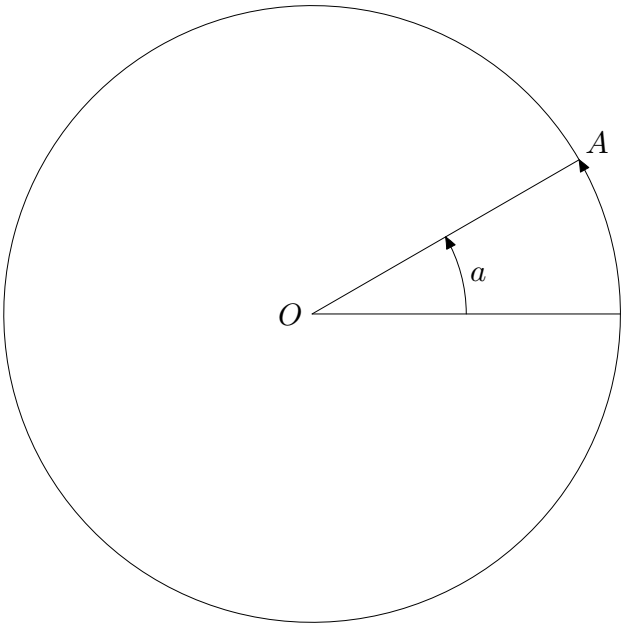


Obrázek 4: Nalezení výšky trojúhelníka ABC.

3.1. Výška trojúhelníku

Naše první makro (Obrázek 4) spočte výšku trojúhelníku ABC procházející bodem A. Výška nemusí protnout protější stranu trojúhelníku přímo, proto je dobrý plán použít konstrukci `whatever`. Makro tedy říká, že průsečík H leží *někde* ve směru úsečky $[B, C]$ a *někde* na protažené úsečce $[A, H]$, kde H je sestrojený bod za použití vektoru $\overrightarrow{BC}$. Makro nevrací průsečík H samotný, ale cestu (trajektorii) jdoucí dál za body A a H alespoň o délku r, což je hodnota, kterou makru poskytneme. Nápad spočívá v tom, že použijeme tyto cesty, abychom našli průsečík s původními kružnicemi, proto je potřeba cesty rozšířit alespoň o poloměr kružnice r a ještě o kousek víc, abychom si byli jisti, že průsečík bude nalezen. Jde o to, že dvě cesty, které mají společný bod, a ten by zároveň byl počátkem jedné z cest, by nemusel být METAPOSTem identifikován z důvodu chyb v zaokrouhlování.

```
vardef triangle_height(expr A,B,C,r)=
  save H,d;
  hide(
    pair H,d;
    H=whatever[B,C]
     =whatever[A,A+((B-C) rotated 90)];
    d=unitvector(H-A);
    )
  ((A-1.1r*d)--(H+r*d)) % výška spuštěná z vrcholu A
enddef;
```



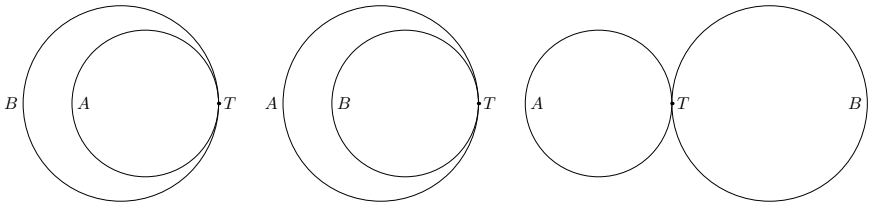
Obrázek 5: Kružnice otočená o a stupňů. Bod A je začátek i konec cesty (křivky) kružnice se středem O.

3.2. Kružnice

Kružnice můžeme získat pomocí makra `fullcircle` a zároveň bude toto makro použito k nalezení průsečíků. Kružnice však mohou být problém, neboť jejich průsečík s přímkou není obvykle jediný, a pokud si přejeme jen jediný průsečík, musíme zajistit, že je to ten, který potřebujeme. Další související problém je nespojitost kružnice jako křivky v METAPOSTu. Ačkoliv to není okem viditelné, kružnice vykreslená METAPOSTem má svůj začátek a konec. Je poměrně rozumné se této nespojitosti vyvarovat, neboť je to zdroj řady potíží.

Jedna konvenční cesta, jak se vyvarovat numerickému problému hledání průsečíku funkcí `instersectionpoint` u nespojité kružnice je otočit tuto kružnici tak, aby její nespojitost byla tam, kde to nebude vadit (Obrázek 5). Vypadá to, jako kdyby tento krok neměl smysl, ale využívá se toho, že funkce vrací první průsečík, který minimalizuje parametry cesty. Pokud tedy zajistíme, že kružnice bude otočena tak, že průsečík s minimálním parametrem je ten chtěný, budeme sladce odměněni.

Z tohoto důvodu si naprogramujeme makro pro kružnici se středem O, poloměrem r a otočenou o úhel a.



Obrázek 6: Tři obecné případy dotyků.

```
def circle(expr O,r,a)=
  (fullcircle scaled 2r
              rotated a shifted O)
enddef;
```

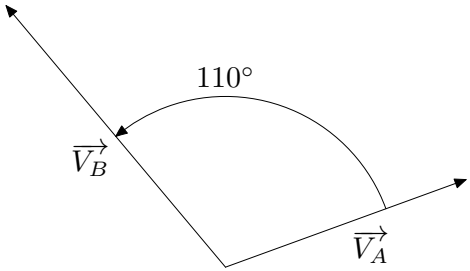
3.3. Body dotyku

Připravíme makro na výpočet bodu dotyku mezi dvěma kružnicemi, o kterých víme, že se dotýkají právě v jednom bodu (Obrázek 6). Makro navíc musí vyřešit situaci, kdy jedna kružnice leží uvnitř druhé. To nastane, když je vzdálenost mezi středy kružnic menší než oba poloměry. Potom kružnice s menším poloměrem leží uvnitř té druhé. Bod dotyku leží na přímce spojující středy kružnic. Například, když kružnice se středem S_a leží uvnitř kružnice se středem S_b, pak bod dotyku dostaneme vztahem

$$S_a + r_a \cdot \overrightarrow{S_aS_b} \,/\, \|\overrightarrow{S_aS_b}\| \,.$$

```
def tangency(expr Sa,ra,Sb,rb)=
 (if (arclength(Sa--Sb)<rb) or  % a uvnitř b nebo
     (arclength(Sa--Sb)<ra):    % b uvnitř a
      if ra<rb:                 % a uvnitř b
        Sa+ra*unitvector(Sa-Sb)
      else:                     % b uvnitř a
        Sb+rb*unitvector(Sb-Sa)
      fi
  else: circle(Sa,ra,0) intersectionpoint (Sa--Sb)
  fi)
enddef;
```

Když nahlédneme pozorně na zdrojový kód tohoto makra, zjistíme, že nám makro nezkolabuje, pokud není nalezen společný bod kružnic kvůli chybám v zaokrouhlování.



Obrázek 7: Úhel mezi dvěma polopřímkami definovaný vektory $\overrightarrow{V_A}$ a $\overrightarrow{V_B}$. Je upraven tak, aby patřil do intervalu $\langle 0, 180° \rangle$.

3.4. Úhel mezi dvěma polopřímkami

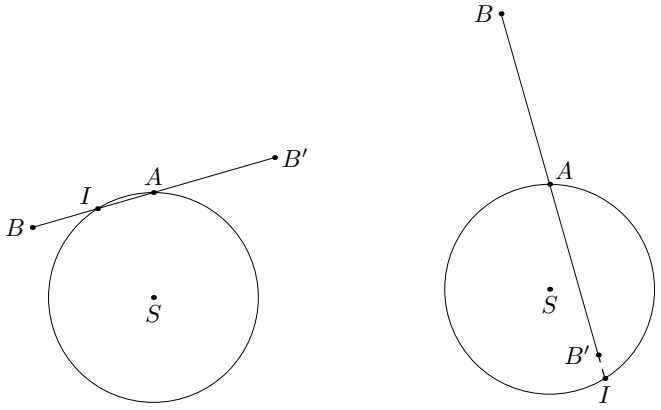
Úhel svíraný dvěma polopřímkami definovanými pomocí vektorů $\overrightarrow{V_A}$ a $\overrightarrow{V_B}$ lze získat tak, že použijeme konstrukci `angle` a zajistíme, aby byl v intervalu $\langle 0, 180° \rangle$ (Obrázek 7).

```
vardef angleof(expr Va,Vb)=
  save a;
  hide(
    a=angle(Vb)-angle(Va);
  forever:
    if a>=180:a:=a-180;fi;
    if a<0:a:=a+180;fi;
    exitif ((a<180) and (a>=0));
  endfor;
  )
  a % vypočtený úhel
enddef;
```

3.5. Průsečík přímky a kružnice

Pro Eppsteinovu konstrukci potřebujeme k danému průsečíku přímky a kružnice najít jejich druhý průsečík. Vymyslet makro pro tento požadavek, aby fungovalo pro všechny nastalé situace, není triviální záležitostí. Jedna ze situací, na kterou si musíme dát pozor, je tečna ke kružnici. Pak může nastat případ, kdy žádný průsečík není nalezen vinou zaokrouhlování.

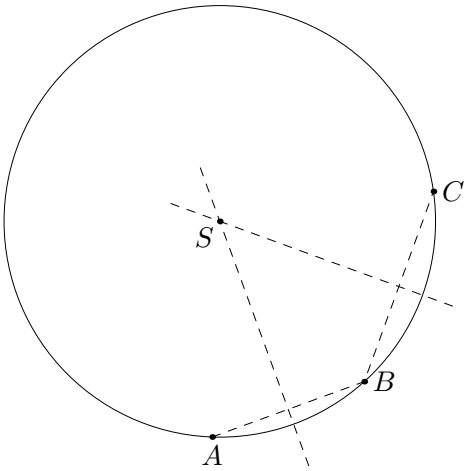
Když píšeme takové makro, musíme zároveň zajistit, že nám nezkolabuje, když budou dosti blízko sebe dva průsečíky (Obrázek 8). Máme-li bod A ležící na kružnici a jiný bod B, pak náš postup bude prvně nalézt úhel mezi vektorem $\overrightarrow{SA}$ a vektorem $\overrightarrow{AB}$. Pokud je tento úhel mezi 89 a 91 stupni, můžeme předpokládat, že přímka AB je tečnou kružnice. I v případě, kdy to tečna není, budou průsečíky dosti blízké a můžeme je ignorovat. V takovém případě makro vrací bod A.



Obrázek 8: Průsečík přímky a kružnice. Kružnice, bod na kružnici A a další bod B jsou zadány. Hledáme další průsečík I. Kratší ze vzdáleností $|SB|$ a $|SB'|$ určuje, na které straně poloroviny s hraniční přímkou SA, resp. na které z polopřímek AB, AB' leží I.

Pokud tomu tak není, najdeme druhý průsečík tak, že porovnáme vzdálenost středu kružnice se dvěma body B a B' ležícími na přímce AB symetricky vzhledem k bodu A. Druhý průsečík pak leží na polopřímce $\overrightarrow{AE}$, kde E je ten z bodů B a B', který je blíže ke středu kružnice. Mírně otočíme kružnicí proti směru hodinových ručiček tak, abychom se vyhli situaci, kdy funkce `intersectionpoint` vrátí průsečík A namísto průsečíku I. To by odpovídalo A s příliš velkým parametrem.

```
vardef intersection_circle_line(expr S,r,A,B)=
  save a,BP,I,uv;
  hide(
    pair BP,I,uv;
    a=abs(angleof(B-A,A-S)-90);
    if a<1: I=A;
    else:
      BP=A+(A-B);
      if arclength(S--B)<arclength(S--BP):
        uv=unitvector(A-B);
        I=circle(S,r,angle(A-S)+1) intersectionpoint
           ((B-2.1r*uv)--(1.1[B,A]));
      else:
        uv=unitvector(A-BP);
        I=circle(S,r,angle(A-S)+1) intersectionpoint
          ((BP-2.1r*uv)--(1.1[BP,A]));
```



Obrázek 9: Kružnice procházející třemi zadanými body A, B a C.

```
      fi;
    fi;)
  I % výsledný průsečík
enddef;
```

3.6. Kružnice procházející třemi body

Závěrečná fáze Eppsteinovy konstrukce bere na vstupu tři body a vykreslí kružnici procházející těmito body (Obrázek 9).

Slouží k tomu makro `circle_through`, které spočítá střed kružnice jako průsečík os dvou úseček. Využívá k tomu pomocné makro `whatevermedian`, které vrací neznámý bod na ose zadané úsečky.

```
def whatevermedian(expr A,B)=
  whatever[.5[A,B],
           .5[A,B]+(B-A) rotated 90]
enddef;
def circle_through
      (expr A,B,C)(text S)(text r)=
  S=whatevermedian(A,B)
   =whatevermedian(B,C);
  r=arclength(A--S);
  draw fullcircle scaled 2r shifted S;
enddef;
```

3.7. Existence průsečíku dvou úseček

Bude užitečné si připravit makro, které zjistí, zdali existuje, průsečík dvou úseček (ne přímek). Relativně snadná cesta k dosažení tohoto cíle je připravit si booleovskou verzi makra `intersectionpoint`. Vrátí `true` (pravda) místo průsečíku a `false` (nepravda) místo chybového hlášení METAPOSTu.

```
secondarydef p intersectionpoint_b q =
  begingroup save x_,y_;
    (x_,y_)=p intersectiontimes q;
    if x_<0:
      false
    else: true
    fi
  endgroup
enddef;
```

3.8. Vnitřní a vnější dotyky kružnic

Připravíme si ještě následující makro, které načte čtyři různé středy kružnic. Kružnice b a c jsou vně tečné a kružnice d je tečná k těmto dvěma vnitřně. Kružnice a je tečná vně k b a c, ale je tečná vnitřně vůči d. Jinými slovy řečeno je kružnice d vnější Soddyho kružnicí ke kružnicím a, b a c. (Obrázek 1)

A naopak, když zadáme kružnice b, c a d, pak Eppsteinova konstrukce spočte dvě kružnice, z nichž jedna je kružnice a. Ukazuje se, že určení bodu na konkrétní kružnici je záležitostí existence průsečíku úseček $[S_a, S_d]$ a $[S_b, S_c]$, kde S_a, S_b, S_c a S_d jsou středy příslušných kružnic. Tyto dvě kružnice tečné vně ke kružnicím b a c, ale tečné vnitřně ke kružnici d, nazveme *vnitřní*, pokud bude dříve zmíněný průsečík existovat, a *vnější*, pokud ne. Odpovídá to středu vnější Soddyho kružnice tehdy, když neexistuje průsečík dříve zmíněných úseček.

Z hlediska praktického rozhodování to vypadá tak, že vnitřní kružnice ze zvažovaných dvou je ta menší.

```
def is_inner(expr Sa,Sb,Sc,Sd)=
  ((Sb--Sc) intersectionpoint_b (Sa--Sd))
enddef;
```

3.9. Směrnice úsečky

Makro `slope` spočte směrnici zadané úsečky vyjádřenou jako úhel. Tohle se nám bude hodit ve chvíli, kdy budeme otáčet kružnici o potřebný úhel.

```
def slope(expr p)=
  angle((point 1 of p)-(point 0 of p))
enddef;
```

4. Hlavní makro

Vstupem pro hlavní část makra jsou středy čtyř kružnic, jejich poloměry a hloubka rekurze. První tři kružnice se navzájem vně dotýkají a hledáme u nich obě Soddyho kružnice. Čtvrtá kružnice nemá přímý význam a přiřadíme jí zápornou délku poloměru.

Během iteračního algoritmu nastanou dvě situace. První případ je ten, kdy kružnice mají navzájem vnější dotyk a hledá se vnitřní Soddyho kružnice. To znamená, že čtvrtá kružnice nebude mít pro nás význam i nadále.

Druhý případ je hraniční situací, kdy první kružnice je vnější Soddyho kružnice, druhá a třetí kružnice mají spolu vnější dotyk a zároveň k Soddyho kružnici vnitřní dotyk. Čtvrtá kružnice má vnější dotyk k druhé a třetí a zároveň vnitřní dotyk ke kružnici první. Čtvrtá kružnice již byla nakreslena, ale musíme ještě zjistit, která je ta poslední kružnice, která ještě nebyla nakreslena.

4.1. Výpočet bodů dotyku a výšek trojúhelníka

Zjištění bodů dotyku a výšek trojúhelníka je přímočará záležitost s využitím dříve nadefinovaných vztahů.

```
vardef tangent_circles(expr Sa,Sb,Sc,So,
           ra,rb,rc,ro,n)=
  save S,T,r,ht,Ihc,Ilc;
  pair S[];     % středy kružnic
  pair T[][];   % body dotyku
  numeric r[];  % poloměry
  path ht[];    % výšky trojúhelníka
  pair Ihc[][]; % průsečíky výšek a kružnic
  pair Ilc[];   % výsledné souřadnice průsečíků
  S1=Sa; S2=Sb; S3=Sc;
  r1=ra; r2=rb; r3=rc;
  T[1][2]=tangency(S1,r1,S2,r2);
  T[2][3]=tangency(S2,r2,S3,r3);
  T[3][1]=tangency(S3,r3,S1,r1);
  ht1=triangle_height(S1,S2,S3,r1);
  ht2=triangle_height(S2,S1,S3,r2);
  ht3=triangle_height(S3,S1,S2,r3);
```

Abychom zjednodušili některé výrazy, nadefinovali jsme makro `next`.

```
def next(expr i)=
  (if i+1<4:i+1 else: 1 fi)
enddef;
```

4.2. Průměry kružnic a další průsečíky

Průsečíky výšek s kružnicemi je snadné získat; klíčem k úspěchu je správně je seskupit. V našem případě prvně seskupíme průsečíky, které získáme ve směru paty výšky tak, že drobně otočíme kružnice ve směru hodinových ručiček. Tím získáme body `Ihc[i][1]`, které jsou na obr. 2 označeny značkou prstenců.

Druhou skupinu bodů tvoří body na opačné polopřímce, které jsou zaznačeny puntíkem.

Poté jsou značky prstenců a puntíků spojeny přímkou s bodem dotyku protějších kružnic. Na původní kružnici díky vzniklé sečně získáme body označené prázdným a plným čtverečkem.

```
for i:=1 upto 3:
  % prstenec
  Ihc[i][1]
     =circle(S[i],r[i],slope(ht[i])-5)
      intersectionpoint ht[i];
  % puntík
  Ihc[i][2]-S[i]=S[i]-Ihc[i][1];
  % čtverec
  Ilc[i]=intersection_circle_line(
        S[i],r[i],
        Ihc[i][1],
        T[next(i)][next(next(i))]);
  % plný čtverec
  Ilc[3+i]=intersection_circle_line(
        S[i],r[i],
        Ihc[i][2],
        T[next(i)][next(next(i))]);
endfor;
```

4.3. Závěrečná fáze

Na závěr postupu, který je v nejobecnějším tvaru rekurze zmíněn níže, musíme zjistit, jestli je makro voláno poprvé, či nikoliv. Pokud je voláno poprvé (větev první), vykreslíme jen vnitřní a vnější Soddyho kružnice.

```
if firststep: % větev 1
  firststep:=false;
  % vnější Soddyho kružnice
  circle_through(Ilc1,Ilc2,Ilc3)
                (S4)(r4);
  % vnitřní Soddyho kružnice
  circle_through(Ilc4,Ilc5,Ilc6)
                (S5)(r5);
```

```
    % zahájení rekurze
    if n>0: % okolní kružnice
      tangent_circles(S4,S1,S2,S3,
                      r4,r1,r2,r3,n-1);
      tangent_circles(S4,S2,S3,S1,
                      r4,r2,r3,r1,n-1);
      tangent_circles(S4,S3,S1,S2,
                      r4,r3,r1,r2,n-1);
    fi;
  else:
    if ro<0: % větev 2
      circle_through(Ilc4,Ilc5,Ilc6)
                    (S5)(r5)
    else:    % větev 3
      if is_inner(So,S2,S3,S1):
        circle_through(Ilc1,Ilc2,Ilc3)
                      (S4)(r4);
      else:
        circle_through(Ilc4,Ilc5,Ilc6)
                      (S4)(r4);
      fi;
    fi;
  fi;
  % zahájení rekurze
  if n>0:    % větev 4
    if ro>0: % větev 5
      tangent_circles(S1,S2,S4,S3,
                      r1,r2,r4,r3,n-1);
      tangent_circles(S1,S3,S4,S2,
                      r1,r3,r4,r2,n-1);
      tangent_circles(S2,S3,S4,origin,
                      r2,r3,r4,-1,n-1);
    else:    % větev 6
      tangent_circles(S1,S2,S5,origin,
                      r1,r2,r5,-1,n-1);
      tangent_circles(S1,S3,S5,origin,
                      r1,r3,r5,-1,n-1);
      tangent_circles(S2,S3,S5,origin,
                      r2,r3,r5,-1,n-1);
    fi;
  fi;
enddef;
```

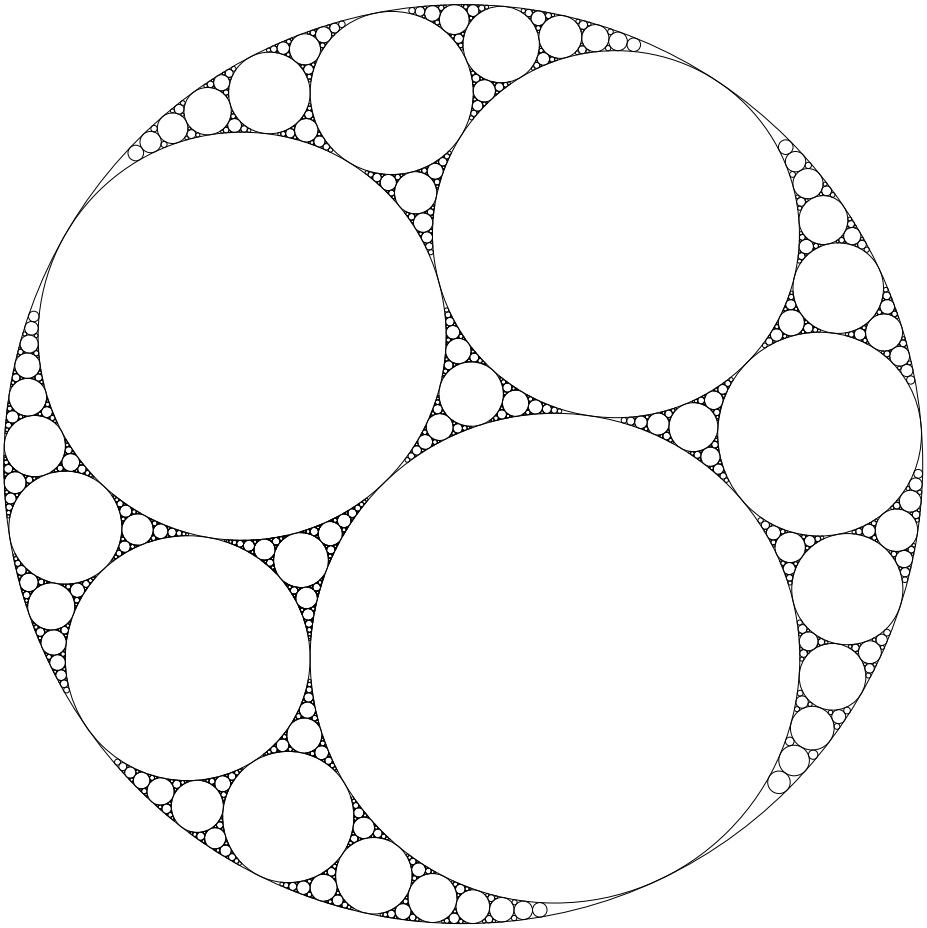
Jakmile máme vnější Soddyho kružnici se středem S_4, zavoláme makro na okolní kružnice. Každé takové iteraci poskytneme vnější Soddyho kružnici jako parametr první, poté dvě ze tří vnitřních kružnic, poté i třetí vnitřní kružnici. V průběhu dalšího volání makra bude aktivována třetí větev a nová kružnice bude vložena mezi dvě vnitřní kružnice a vnější Soddyho kružnici užitím dříve zmíněného kritéria.

Druhá větev se použije tehdy, když má být nalezena vnitřní Soddyho kružnice ze tří kružnic s vnějším dotykem.

Bez ohledu na to, co se již v programu událo, jakmile dosáhneme čtvrté větve, tak jsme již nalezli kružnici a nastanou dvě nové možnosti: buď se dotýkáme původní vnější Soddyho kružnice se středem S_1 (pátá větev), nebo je kružnice vnitřní Soddyho kružnicí tří kružnic s vnějším dotykem (šestá větev). Rozbor páté větve nás navede ke dvěma hraničním případům a jednomu vnitřnímu (vnitřní Soddyho kružnice). Rozbor šesté větve nás navede ke třem novým vnitřním případům (obvyklé vnitřní Soddyho kružnice). Všimněme si, že parametr `origin` chod makra neovlivní, když je poloměr záporný.

Hlavní makro se spouští následujícím kódem, jehož výsledkem je Apolloniův fraktál hloubky 7 (Obrázek 10).

```
pair S[];    % středy kružnic
numeric r[]; % poloměry kružnic
numeric n;   % hloubka fraktálu, konkrétně rekurze
n=7;         % hloubka rekurze, konkrétně fraktálu
S1=origin;
r1=4cm;
r2=3cm;
r3=6cm;
% nalezení středu kružnice S2
S2-S1=(r1+r2,0);
% dostáváme dvě řešení u kružnice se středem S3,
% ukládáme si jen jedno z nich
S3=circle(S1,r1+r3,0)
    intersectionpoint circle(S2,r2+r3,0);
draw circle(S1,r1,0);
draw circle(S2,r2,0);
draw circle(S3,r3,0);
firststep:=true;
tangent_circles(S1,S2,S3,origin,
                r1,r2,r3,-1,n);
```



Obrázek 10: Apolloniův fraktál hloubky 7 s celkem 6563 kružnicemi. Počet kružnic lze vypočítat ze vzorce $5 + 3 \cdot (3^n - 1)$, kde n udává hloubku rekurze. Pro $n = 0$ výpočtem zjistíme, že dostáváme pět kružnic, což jsou tři zadané, vnitřní a vnější Soddyho kružnice.

5. Závěr

Naše malá procházka napříč líbajícími se kružnicemi uvedla detaily geometrické konstrukce v METAPOSTu. Konkrétně jsme podrobně prošli Eppsteinovu konstrukci. Konečná verze kódu je jednoduchá, ale tato jednoduchost nebyla získána hned a bezprostředně. Navíc je kód podpořen robustností každého makra, které se navíc snaží být maximálně obecné. Zdrojový kód není nutné použít pouze pro uvedené hodnoty parametrů a měl by fungovat ve všech případech, kdy nedojde k překročení mezí METAPOSTu.

Odkazy

1. ROEGEL, Denis. Kissing Circles: A French Romance in METAPOST. *TUGboat*. 2005, roč. 26, č. 1, s. 10–17. Dostupné také z: `https://tug.org/TUGboat/Articles/tb26-1/tb82roegel.pdf`.
2. GISCH David; Ribando, Jason M. Apollonius' Problem: A Study of Solutions and Their Connections. *American Journal of Undergraduate Research*. 2004, roč. 3, č. 1, s. 15–25. ISSN 1536-4585. Dostupné také z: `http://www.ajuronline.org/uploads/Volume%203/Issue%201/31D-GischArt.pdf`.
3. EPPSTEIN, David. Tangent Spheres and Triangle Centers. *American Mathematical Monthly*. 2001, roč. 108, č. 1, s. 63–66. ISSN 0002-9890. Dostupné také z: `https://arxiv.org/abs/math.MG/9909152v1`.

Summary: Kissing Circles: A French Romance in METAPOST

When circles meet, they kiss. If three of them kiss, others can try to join and kiss all of them at once. In this article, we look at this problem from the METAPOST point of view, and we try to tell circles how to kiss, no matter their position and size. Recursive kissing will also be attempted.

Keywords: METAPOST, kissing circles, inner and outer Soddy circles, David Eppstein's construction, Apollonius' problem, Apollonius' gasket.

Denis Roegel, roegel@loria.fr
http://www.loria.fr/~roegel
LORIA – Campus Scientifique, BP 239
F-54506 Vandœuvre-lès-Nancy Cedex, France

Vysokoúrovňové jazyky pro TEX

Vít Novotný

TEX je strojový kód světa digitální sazby, který od spisovatelů a grafiků vyžaduje netriviální programátorské dovednosti a programátorům poskytuje minimum vysokoúrovňových abstrakcí. V článku představuji vybrané značkovací, programovací a stylové jazyky pro TEX, které umožňují dělbu práce mezi spisovatele, vývojáře a grafiky a usnadňují proces přípravy elektronických dokumentů. Článek je přepis mé přednášky na valném shromáždění CSTUGu 14. května 2022 [1].

Klíčová slova: vysokoúrovňové jazyky, programovací jazyky, značkovací jazyky, stylové jazyky, ε-TEX, pdfTEX, LuaTEX, LuaMetaTEX, LATEX 2ε, LATEX3, expl3, XML, DocBook, TEI, XHTML, XSLT, CSS, CSL, ConTEXt, HTML, markdown, YAML, Pandoc, Ti*k*Z, BibLATEX, BibLATEXML, LyLuaTEX

Kreativnímu jedinci je TEX pozvánkou k tomu, aby se na chvíli stal spisovatelem, grafikem, ilustrátorem, sazečem a vývojářem v jedné osobě. Rozsáhlým dokumentům však prospívá, když obsah, stylopisy a programová výbava vznikají do jisté míry nezávisle.[1] To umožňuje dělbu práce mezi doménové experty, aniž by neúnosně rostly náklady na vzájemnou koordinaci. Kreativec, který dokáže zastoupit několik rolí, se může v případě potřeby omezit jen na jednu z nich a věnovat jí svou plnou a ničím nerušenou pozornost.

TEXový dokument můžeme rozdělit na obsah, stylopisy a programovou výbavu jednoduše tak, že vytvoříme tři samostatné soubory:

- text dokumentu označkovaný TEXovými makry
- stylopis s nastavením délkových registrů, písem, výstupní rutiny, apod.
- program s definicí značkovacích maker

Při dělbě práce ale můžeme narazit na to, že doménoví experti nejsou schopni nebo ochotni používat při své práci TEX, což má několik důvodů:

[1] Čtenář může namítnout, že rozdělení dokumentu na nezávislé části je fikce: Jsou obrázky v textu doménou spisovatele, nebo ilustrátora? Může spisovatel měnit písmo a barvu textu či způsob číslování seznamů, nebo jde o úkol grafika? Kde jsou hranice mezi rolemi grafika, sazeče a vývojáře?

Tyto námitky jsou na místě především u akcidenčních a nízkonákladových dokumentů, jako jsou plakáty, básnické sbírky nebo jídelní a nápojové lístky. U takových dokumentů může být vhodnější zvolit takový přístup, při kterém jsou role volnější a mezi účastníky procesu přípravy dokumentu probíhá soustavná komunikace. Autor článku má zkušenosti převážně s přípravou odborných a technických dokumentů, u kterých jsou jednotvárnost a kompozicionalita úmyslnými prvky stylu. Takové dokumenty lze snadno rozdělit na dílčí části bez újmy na celku.

DOI: 10.5300/2022-1-4/35

- Moderní značkovací jazyky jako markdown minimalizují poměr značek vůči textu pro snadný zápis a zvýšenou čitelnost, zatímco TEX bývá při větším množství značek upovídaný a nepřehledný.[2]
- Moderní stylovací jazyky jako CSS jsou deklarativní a nepožadují programátorské dovednosti, zatímco TEX je imperativní programovací jazyk.
- Moderní programovací jazyky jako Python nabízí vysokoúrovňové abstrakce a bohaté základní knihovny vestavěných funkcí, zatímco TEX nabízí pouze primitivní datové typy a operace nad nimi.

Nicméně existuje mnoho vysokoúrovňových jazyků pro TEX, na které se výše uvedená omezení nevztahují a které můžeme použít pro přípravu obsahu, stylopisů a programové výbavy namísto nízkoúrovňového TEXu.

V sekci 1 shrnuji základní pojmy související s TEXem, jako jsou stroje, makrobalíky a formáty. V sekcích 2 až 5 nabízím přehled programovacích, značkovacích, stylových a dalších doménově specifických vysokoúrovňových jazyků pro TEX. V sekci 6 shrnuji poznatky z tohoto článku. V sekci 7 se zamýšlím nad dalším směřováním TEXu a vysokoúrovňových jazyků jako takových.

1. Přehled základních pojmů

TEX je nízkoúrovňový programovací jazyk pro digitální sazbu [2]. Referenční implementací TEXu je *stroj* TEX90 od Donalda Knutha [3]. Moderní TEXové stroje jako ε-TEX [4], pdfTEX [5] a LuaTEX [6] rozšiřují TEX90 o dodatečné primitivní příkazy, které zvyšují vývojářský komfort.

Makrobalíky jako plain [7], LATEX [8] a ConTEXt [9] staví z primitivních příkazů TEXových strojů vysokoúrovňové značkovací a programovací jazyky pro spisovatele a vývojáře. *Formáty* odpovídají kombinaci stroje a makrobalíku, např. LuaTEX + LATEX = LuaLATEX[3] a LuaTEX + ConTEXt = ConTEXt MkIV. Pomocí formátů připravuje sazeč dokumenty z příkazové řádky s příkazy jako `lualatex` a `context`.

2. Programovací jazyky pro vývojáře

V této sekci popisuji, jaké možnosti nabízí moderní TEXové stroje a makrobalíky vývojářům. Veškeré ukázky v této sekci vypíší text $1 + 2 = 3$.

[2] Čitelnost textu označkovaného v TEXu se zlepší, pokud značky sestavíme z interpunkčních znamének. Pokud se však pro usnadnění zápisu omezíme na znakovou sadu ASCII, skončíme s pouhými 32 znaky, z nichž 10 už je využíváno TEXem. Při větším množství značek by proto jednotlivá interpunkční znaménka odpovídala několika různým značkám. Vyřešení nejednoznačností by vyžadovalo buďto složitou lexikální a syntaktickou analýzu textu, pro kterou TEX nemá vhodné primitivní datové typy ani příkazy, nebo bychom museli interpunkci ve značkách použít způsobem, který usnadňuje analýzu na úkor čitelnosti. Proto se při větším množství značek typicky místo interpunkce používají upovídaná TEXová makra.

[3] Většina TEXových formátů poskytuje vlastní značkovací jazyky. Formát LATEX 2_ε je však zdaleka nejznámější a také nejdůslednější v odstínění spisovatele od programování a stylování.

Stroj ε-TEX a jeho následovníci jako pdfTEX a LuaTEX nabízí primitivní příkaz `\numexpr`, který vyhodnocuje celočíselné aritmetické výrazy:

```
$ 1 + 2 = \numexpr 1 + 2 \relax $
```

Příkaz `\numexpr` zvyšuje komfort oproti primitivním příkazům TEXu 90:

```
$ 1 + 2 = \newcount\x \x=1 \advance\x by 2 \the\x $
```

Pro další primitivní typy má ε-TEX příkazy `\dimexpr`, `\glueexpr` a `\mugluexpr`.

Stroje LuaTEX a LuaMetaTEX [10] nabízí primitivní příkaz `\directlua`, který umožňuje zadávat a spouštět programy v jazyce Lua:

```
$ 1 + 2 = \directlua { tex.print(1 + 2) } $
```

Kromě základní knihovny jazyka Lua mohou vývojáři interagovat s TEXovým strojem a instalací TEXu [6, kapitoly 5–10; 10, kapitoly 4–10] a využívat rozšiřující softwarové knihovny pro práci se soubory a zpracování textu [6, sekce 4.3].

Stroje pdfTEX a LuaTEX rozšiřují primitivní příkaz `\input` o variantu, která spouští libovolné programy pomocí příkazové řádky operačního systému:

```
$ 1 + 2 = \input|" echo 1 + 2 | bc " $
```

Rozšířená varianta příkazu `\input` umožňuje integrovat TEXový kód s širším ekosystémem programové výbavy mimo instalaci TEXu.[4]

Součástí experimentálního formátu LATEX3 je makrobalík *`expl3-generic`*, který poskytuje vysokoúrovňový programovací jazyk expl3 [11; 12; 13]:

```
\input expl3-generic
\ExplSyntaxOn
$ 1 + 2 = \int_eval:n { 1 + 2 } $
\ExplSyntaxOff
```

Expl3 nabízí vysokoúrovňové datové typy pro obecné programování (seznamy a hašové tabulky) a typografické programování (rakve[5] a barvy) a bohatou základní knihovnu funkcí pro řízení toku programu a TEXové expanze, celočíselnou i reálnou aritmetiku a zpracování TEXových tokenů a unicodového textu. Jazyk expl3 využívá primitivní příkazy strojů TEX90, ε-TEX a pdfTEX; funguje ale i s novějšími stroji, jako jsou XETEX, LuaTEX a LuaMetaTEX [12, sekce 9].

[4] Jednou z nevýhod rozšířené varianty příkazu `\input` je vazba na konkrétní příkazovou řádku a programovou výbavu, což snižuje přenositelnost dokumentů. V našem příkladu se jedná o příkazovou řádku Bourne shell z UNIXu V7 (také `sh`), případně o zpětně kompatibilní nadstavby jako `bash`, `dash` a `ksh`, a o unixovou kalkulačku Bench calculator (`bc`). Většina Linuxových distribucí ale používá příkazovou řádku kompatibilní s `sh` a zahrnuje `bc` v základní programové výbavě; náš příklad na nich tedy můžete bez úpravy vysázet.

Další nevýhodou rozšířené varianty příkazu `\input` je skutečnost, že představuje bezpečnostní riziko a uživatel ji proto musí explicitně povolit parametrem `-shell-escape`. Například pro vysázení našeho příkladu musíme použít příkaz `pdftex -shell-escape` ⟨*jméno dokumentu*⟩.

[5] Rakev (z anglického *coffin*) sestává z TEXového boxu, informací o jeho tvaru a rozměrech a množiny vertikálních a horizontálních přímek. Průsečíky přímek tvoří úchyty rakve, které slouží k vzájemnému pozicování rakví na stránce.

Vývojáře zvyklé na kategorie znaků v konvenčních formátech jako plain TEX, LATEX a ConTEXt mohou u explu překvapit podtržítka a dvojtečky s kategorií písmene a bílé znaky (␣, ↳) s kategorií ignorovaného znaku. Studie však ukazují, že oddělování slov v identifikátorech pomocí_podtržítek je čitelnější než oddělování slov pomocíKapitálek [14], mezery jsou častým zdrojem chyb při programování v TEXu a dvojtečky v explu slouží pro oddělení názvu příkazu (`\int_eval`) od jeho typové signatury (`:n`).[6] Kategorie znaků v explu jsou tedy účelné.

3. Značkovací jazyky pro spisovatele

V této sekci se podíváme na některé existující značkovací jazyky a možnosti jejich využití v TEXu.

> *Je překvapivě obtížné přesvědčit o tom uživatele, ale nedostatky LATEXu pro koncentraci, psaní a myšlení si v ničem nezadají s nedostatky Wordu. Je to z toho prostého důvodu, že LATEX dává spisovateli do rukou příliš velkou moc: Vždy existuje další makrobalík, který můžeme zavést v preambuli, stejně jako vždy existuje další rozbalovací nabídka ve Wordu.* — Thompson [15, v překladu autora]

Formát LATEX 2ε poskytuje jednoduchý značkovací jazyk pro přípravu dokumentů, který je možné rozšiřovat pomocí makrobalíků:[7]

```
\documentclass{book}
\usepackage[czech]{babel}
\title{Ukázkový dokument v~\LaTeX u}
\author{Vít Novotný}
\date{14. května 2022}
\begin{document}
\maketitle
\chapter{Kapitola}
Ahoj, \LaTeX u!
\end{document}
```

[6] Typové signatury umožňují definovat příkazy s jiným typem argumentu, než s jakým příkazy voláme. Můžeme např. zadefinovat příkaz `\pozdrav:n`, který bude přijímat jeden argument s TEXovými tokeny:

```
\cs_new:Nn \pozdrav:n { Ahoj,~#1! }
```

Při volání příkazu pro nás ale může být snazší použít jako argument např. jméno proměnné:

```
\cs_generate_variant:Nn \pozdrav:n { v }
\tl_new:N   \g_jmeno_tl
\tl_set:Nn  \g_jmeno_tl { světe }
\pozdrav:v { g_jmeno_tl }  % Expanduje na \pozdrav:n{světe} a poté na Ahoj, světe!
```

Díky typovým signaturám může expl3 automaticky přetypovat argumenty, což zvyšuje komfort.

[7] Většina TEXových formátů poskytuje vlastní značkovací jazyky. Formát LATEX 2ε je však zdaleka nejznámější a také nejdůslednější v odstínění spisovatele od programování a stylování.

Pokud nejsme spokojeni s automatickým výstupem vysokoúrovňových značek jako `\chapter`, můžeme místo nich použít stylovací příkazy LaTeXu jako `\textbf` a primitivní příkazy TeXového stroje jako `\vskip`. Toto může být vhodné pro řešení konkrétních typografických nedostatků, které nelze řešit systémově. Nadměrné užívání nízkoúrovňových příkazů ale narušuje dělbu práce a vede k nejednotnému vzhledu koncového dokumentu.

Mimo svět TeXu jsou oblíbené značkovací jazyky založené na metajazyku XML. Můžeme si navrhnout buďto svůj vlastní XML jazyk [16] nebo využít existující jazyk jako DocBook, TEI, nebo XHTML (vizte níže):

```
<?xml version="1.0" encoding="utf-8"?>
<html xml:lang="cs" xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title>Ukázkový dokument v XHTML</title>
        <meta name="author" content="Vít Novotný" />
    </head>
    <body>
        <h1>Kapitola</h1>
        <p>Ahoj, XHTML!</p>
    </body>
</html>
```

Pro zpracování tohoto dokumentu TeXem požádáme o pomoc vývojáře. Ten buďto využije vestavěnou podporu XML v pokročilých TeXových formátech jako ConTeXt [17; 18] nebo připraví program v jazyce XSLT:

```
<?xml version="1.0" encoding="utf-8"?>
<stylesheet xmlns="http://www.w3.org/1999/XSL/Transform"
            xmlns:xhtml="http://www.w3.org/1999/xhtml">
            version="1.0">
    <output method="text" />

    <!-- Následující část zpracovává prvek <head> -->
    <template match="xhtml:head">
        \documentclass{book}
        \usepackage[czech]{babel}
        <!-- Zpracuj zanořené prvky <title> a <meta> -->
        <apply-templates select="*"/>
    </template>

    <template match="xhtml:title">
        \title{<value-of select="text()" />}
    </template>
```

```
    <template match="xhtml:meta">
        <choose>
            <when test="@name = 'author'">
                \author{<value-of select="@content" />}
            </when>
        </choose>
    </template>

    <!-- Následující část zpracovává prvek <body> -->
    <template match="/xhtml:html/xhtml:body">
        \begin{document}
        \maketitle
        <!-- Zpracuj zanořené prvky <h1> a <p> -->
        <apply-templates select="*"/>
        \end{document}
    </template>

    <template match="xhtml:h1">
        \chapter{<value-of select="text()" />}
    </template>

    <template match="xhtml:p">
        <value-of select="text()" /> \par
    </template>
</stylesheet>
```

Příkazem `xsltproc` ⟨*jméno XSLT programu*⟩ ⟨*jméno XML dokumentu*⟩ ze softwarové knihovny `libxslt` převedeme náš XML dokument na LaTeXový dokument:

```
\documentclass{book}
\usepackage[czech]{babel}
\title{Ukázkový dokument v~XHTML}
\author{Vít Novotný}
\begin{document}
\maketitle
\chapter{Kapitola}
Ahoj, XHTML! \par
\end{document}
```

XML jazyky mají dobrou podporu v softwarových knihovnách, což usnadňuje další zpracování dokumentů. Na rozdíl od LaTeXu nemůže spisovatel v XML jazycích snadno řešit konkrétní typografické nedostatky, což ztěžuje přípravu akcidenčních a nízkonákladových dokumentů. Vzhledem k vysokému poměru značek vůči textu je pro spisovatele výhodné použít specializovaný textový editor pro snadný zápis.

> *V markdownu je spisovatel vždy konfrontován pouze s jednou otázkou, a je to ta správná otázka: Jak by měla znít následující věta?*
> — Thompson [15, v překladu autora]

Odlehčené značkovací jazyky jako YAMLoň[8] [19] a markdown [20] minimalizují poměr značek vůči textu pro snadný zápis a zvýšenou čitelnost:

```
---
title:  Ukázkový dokument v YAMLoni a markdownu
author: Vít Novotný
date:   2022-05-14
lang:   cs
---

# Kapitola
Ahoj, YAMLoni a *markdowne*!
```

Pro zpracování tohoto dokumentu TEXem můžeme použít např. konverzní nástroj Pandoc [21], makrobalík *markdown* [22] nebo oba zároveň [23; 24, sekce 2.3].

Markdown je jednoduchý jazyk, kterému chybí značky pro složitější a méně časté prvky jako tabulky, poznámky a citace. Pandoc i *markdown* proto nabízí rozšiřující značky[9] a umožňují uživatelům vytvářet značky vlastní [21, sekce Filters; 22, sekce 2.1.2] nebo připojovat k prvkům doplňující *atributy*:[10]

```
Ahoj, rozšiřující značky[^1] a [atributy]{.vysazej-mne-tucne}.

[^1]: Ahoj, já jsem rozšiřující značka pro poznámky.
```

Jazyk markdown původně vznikl jako preprocesor jazyka HTML a spisovatel v něm proto může využívat také HTML značky:[11]

```
Ahoj také <abbr title="hatmatilko">HTML</abbr>.
```

Pro sazbu matematiky, další rozšíření repertoáru značek a řešení konkrétních typografických nedostatků může spisovatel v markdownu použít i primitivní příkazy TEXového stroje a příkazy TEXových formátů:[12]

```
Ahoj i vám, `\LaTeX u`{=latex} a $\text{matematiko}$.
```

Nadužívání rozšiřujících značek a atributů snižuje čitelnost vstupních dokumentů, ztěžuje jejich další zpracování, narušuje dělbu práce a vede k nejednotnému vzhledu koncových dokumentů. Užívejte je proto s mírou.

[8] YAMLoň podle vzoru jabloň

[9] V Pandocu a *markdownu* je třeba zápis tabulek, poznámek a citací povolit volbami `pipe_tables`, `footnotes` a `citations`.

[10] V Pandocu a *markdownu* je třeba zápis atributů povolit volbami `header_attributes`, `fenced_code_attributes`, `inline_code_attributes` a `link_attributes`.

[11] V *markdownu* je třeba zápis HTML značek povolit volbou `html`.

[12] V Pandocu je třeba zápis matematiky a TEXových značek povolit volbami `tex_math_dollars` a `raw_attribute`. V *markdownu* je třeba povolit volbu `hybrid`.

Různé značkovací jazyky mají různé výhody a může být vhodné je kombinovat. Markdown a YAMLoň mohou sloužit jako vstupní jazyky pro spisovatele. Pomocí Pandocu můžeme vstupní dokumenty převést do XML mezijazyka pro archivaci a další zpracování. Z XML mezijazyka získáme dokument v LaTeXu nebo jiném TeXovém formátu, který může sloužit jako koncový jazyk pro sazeče.

4. Stylové jazyky pro grafiky

V TeXu nelze využívat vysokoúrovňové deklarativní stylové jazyky jako CSS a grafici jsou proto odkázáni na programování v TeXu. V této sekci stručně referuji o tom, jaké možnosti usnadnění nabízí grafikům formáty LaTeX 2_ε a LaTeX3.

Formát LaTeX 2_ε poskytuje vysokoúrovňové příkazy pro vytváření pojmenovaných stylů stránek (`\newpagestyle`), změny obsahu záhlaví a zápatí (`\markboth`) a nastavení rodiny, řezu a velikost písma nezávisle na sobě a nezávisle na konkrétním písmu (`\textbf`) [25]. LaTeX 2_ε dále poskytuje délkové registry pro nastavování rozměrů sazebního zrcadla (`\textheight`) a vzhledu některých LaTeXových prvků jako seznamy (`\itemsep`). Rozšiřující makrobalíky LaTeXu jako *`enumitem`*, *`geometry`* a *`fancyhdr`* umožňují grafikovi měnit další aspekty vzhledu koncového dokumentu bez programování.

Součástí formátu LaTeX3 je makrobalík *`xtemplate`* [26; 27], který sazeči, grafikovi a vývojáři pomáhá společně připravovat stylopisy. Nejprve sazeč definuje *typy* prvků dokumentu, např. sekce:

```
\usepackage{xtemplate}
\ExplSyntaxOn
\DeclareObjectType { sekce } { 1 }  % Sekce mají 1 argument: název
```

Grafik pro každý typ vytvoří *šablonu*. Šablona zadává vysokoúrovňové grafické parametry, kterými se od sebe liší různé úrovně sekcí:

```
\DeclareTemplateInterface { sekce } { moje-šablona } { 1 }
  {
    mezera-nad : skip,
    mezera-pod : skip,
    písmo : tokenlist,
    zarovnání : choice { doleva, doprostřed, doprava } = doleva,
  }
```

Vývojář implementuje parametry šablony pomocí vysokoúrovňového jazyka expl3, stylovacích příkazů LaTeXu a primitivních příkazů TeXového stroje:

```
\skip_new:N \l_mezera_nad_skip
\skip_new:N \l_mezera_pod_skip
\tl_new:N \l_pismo_tl
\tl_new:N \l_zarovnani_tl
```

```
\DeclareTemplateCode { sekce } { moje-šablona } { 1 }
  { % Hodnoty parametrů ukládáme do lokálních proměnných
    mezera-nad = \l_mezera_nad_skip ,
    mezera-pod = \l_mezera_pod_skip ,
    písmo = \l_pismo_tl ,
    zarovnání = {
      doleva = \cs_set_eq:NN \l_zarovnani_tl \raggedright ,
      doprava = \cs_set_eq:NN \l_zarovnani_tl \raggedleft ,
      doprostřed = \cs_set_eq:NN \l_zarovnani_tl \centering ,
    },
  }
  { % Při spuštění šablony vysázíme nadpis sekce
    \AssignTemplateKeys
    \par \skip_vertical:N \l_mezera_nad_skip
    \group_begin:
      \l_zarovnani_tl
      \l_pismo_tl
      #1
      \par \skip_vertical:N \l_mezera_pod_skip
    \group_end:
  }
```

Grafik navrhne *instance* šablony s konkrétními hodnotami parametrů pro různé úrovně sekcí jako kapitoly:

```
\DeclareInstance { sekce } { kapitola } { moje-šablona }
  {
    mezera-nad = 10pt,
    mezera-pod = 12pt,
    písmo = { \Large \bfseries } ,
  }
\ExplSyntaxOff
```

Když pak spisovatel v dokumentu zadá nadpis sekce, spustí se příslušná instance:

```
\UseInstance{sekce}{kapitola}{Název mojí kapitoly}
```

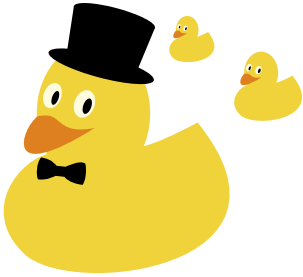
Při použití makrobalíku *xtemplate* může být grafik součástí procesu přípravy dokumentu a průběžně upravovat instance, aniž by musel programovat. V budoucnu má být součástí formátu LaTeX3 systém LDB [28; 29], který grafikům oproti makrobalíku *xtemplate* umožní postihnout vzájemné interakce mezi prvky, jako např. seznam bezprostředně po nadpisu (v notaci LDB `!head<list`), zanořené seznamy (`<list*<list`) a druhý popisek obrázku (`<float*<caption>*<caption`).

5. Doménově specifické jazyky pro experty

Kromě programovacích, značkovacích a stylových jazyků existuje mnoho dalších vysokoúrovňových jazyků pro doménové experty, jako jsou knihovníci, ilustrátoři a hudebníci. V této sekci uvádím přehledový výčet několika takových jazyků.

Makrobalík *TikZ* [30] poskytuje jazyk pro přípravu ilustrací:

```
\usemodule[tikzducks]
\usecolors[xwi]
\starttext
\starttikzpicture
\duck[tophat, bowtie]
\duck[xshift=60pt, yshift=40pt, scale=0.3]
\duck[xshift=44pt, yshift=55pt, scale=0.2]
\stoptikzpicture
\stoptext
```



Jazyk Ti*k*Zu je nezávislý na použitém TEXovém formátu (zde ConTeXt MkIV) a lze ho rozšiřovat pomocí makrobalíků (zde ***tikzducks*** [31]).

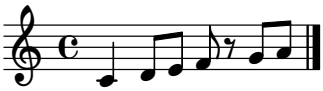
Makrobalík *BibLATEX* [32] poskytuje jazyk pro přípravu bibliografie:

```
@online{lehnman2022biblatex,
    title = {The Bib\LaTeX{} Package},
    subtitle = {Programmable Bibliographies and Citations},
    author = {Philip Kime and Moritz Wemheuer and Philipp Lehman},
    date = {2022-07-12},
    url = {https://ctan.org/pkg/biblatex},
    urldate = {2022-10-06},
}
```

Jazyk BibLATEXu je přesně definovaný [32, sekce 2] a bibliografické záznamy lze validovat a převést do XML jazyka BibLATEXML pro další zpracování [33, sekce 3.4]. BibLATEX lze rozšiřovat pomocí makrobalíků s citačními styly [34]. Chybí podpora stylopisů ve standardním deklarativním stylovém jazyce CSL.

Makrobalík *LyLuaTEX* [35] poskytuje jazyk pro notový zápis:

```
\score {
    \relative c' {
        \time 4/4
        \clef treble
        c4 d8 e f r g a \bar "|."
    }
}
```



LyLuaTEX potřebuje TEXový formát LuaLATEX a instalovaný program LilyPond.

6. Závěr

TEX je strojový kód světa digitální sazby, který od spisovatelů a grafických návrhářů vyžaduje netriviální programátorské dovednosti a programátorům poskytuje minimum vysokoúrovňových abstrakcí. V článku jsme si představili značkovací, programovací a stylové jazyky pro TEX, které umožňují dělbu práce mezi vývojáře, spisovatele a grafické návrháře a které usnadňují proces přípravy elektronických dokumentů.

Článek jsem připravil ve vysokoúrovňovém značkovacím jazyku markdown pro spisovatele a v doménově specifickém jazyce BibLATEX pro knihovníky. Zdrojový kód článku je dostupný online [36].

7. Výhled do budoucnosti

Veškeré jazyky, které jsme si představili v článku, jsou uměle navržené tak, aby byly syntakticky jednoznačné a snadno strojově čitelné. Tím se zcela liší od přirozeného jazyka, který je víceznačný a jeho strojové zpracování bylo svatým grálem informatiky od jejího vzniku. Umělé jazyky jsou překážkou pro doménové experty, kteří ovládají svůj mateřský jazyk, ale nejsou vystudovaní informatici.

V posledních pěti letech došlo na poli umělé inteligence k významným posunům [37; 38], které zvýšily strojovou čitelnost přirozeného jazyka a umožňují automaticky generovat kód programu na základě pokynů zadaných v přirozeném jazyce [39]. Budoucím vysokoúrovňovým značkovacím,[13] programovacím i stylovým jazykem proto může být přirozený jazyk.[14]

Odkazy

1. NOVOTNÝ, Vít. *Vysokoúrovňové jazyky pro TEX*. 2022. Dostupné také z: `https://www.cstug.cz/informace/zpravy/2022-04-01-valna-hromada-2022/`.
2. KNUTH, Donald E. *The TEXbook*. Sv. A. Reading, MA: Addison-Wesley, 1984. Computers & Typesetting. V současnosti jsou dostupné 35. výtisk (měkká vazba, 2017) a 23. výtisk (pevná vazba, 2021).

[13] Dnešní jazykové modely dokážou v písemném projevu opravit překlepy [40] a doplnit chybějící interpunkci [41]. O značkovacích jazycích můžeme uvažovat jako o rozšířené množině interpunkčních znamének, která nám umožňují text dále členit. Na první pohled by se tedy zdálo, že doplňování chybějících značek je typově stejný problém jako doplňování chybějící interpunkce a k jeho řešení je možné použít stejné techniky. Tuto hypotézu by bylo vhodné ověřit na kolekcích označkovaných textů, jako jsou webové stránky označkované v jazyce HTML.

[14] Posuny v umělé inteligenci se týkají i dalších modalit, jako je např. obraz [42]. To grafikovi umožní doplnit textový popis o nákresy, neboť obrázek vydá za tisíc slov. Tuto primární dokumentaci můžeme kdykoliv strojově přeložit na stylopis v jazyce nižší úrovně, jako je CSS.

3. KNUTH, Donald E. *TEX: The Program*. Sv. B. Reading, MA: Addison-Wesley, 1986. Computers & Typesetting. Od pátého výtisku (1994) xvi + 600 stran. V současnosti je dostupný 11. výtisk (pevná vazba, 2021).
4. THE NTS TEAM; BREITENLOHNER, Peter. *ε-TEX: An extended version of TEX, from the NTS project* [online]. CTAN, 1998-02 [vid. 2022-09-26]. Dostupné z: `https://ctan.org/pkg/etex`. Verze 2.
5. THÀNH, Hàn Thế et al. *pdfTEX: A TEX extension for direct creation of PDF* [online]. CTAN, 2022-03-01 [vid. 2022-09-26]. Dostupné z: `https://ctan.org/pkg/pdftex`. Rev. 875.
6. LUATEX DEVELOPMENT TEAM. *LuaTEX Reference Manual* [online]. CTAN, 2022-02-28 [vid. 2022-09-26]. Dostupné z: `https://ctan.org/pkg/luatex`. Verze 1.15.
7. KNUTH, Donald E. *Plain: The Plain TEX format* [online]. CTAN, 2021 [vid. 2022-09-26]. Dostupné z: `https://ctan.org/pkg/plain`. Verze 3.141592653.
8. LAMPORT, Leslie. *LATEX: A Document Preparation System*. 2. vyd. Addison-Wesley, 1994. ISBN 978-0201529838.
9. HAGEN, Hans. *ConTEXt: the manual* [online]. PRAGMA ADE, 2001 [vid. 2022-09-26]. Dostupné z: `http://pragma-ade.com/general/manuals/cont-eni.pdf`.
10. LUAMETATEX DEVELOPMENT TEAM. *LuaMetaTEX Reference Manual* [online]. PRAGMA ADE, 2022-08-03 [vid. 2022-10-04]. Dostupné z: `http://pragma-ade.nl/general/manuals/luametatex.pdf`. Verze 2.09.59.
11. THE LATEX PROJECT TEAM. *The LATEX3 kernel: style guide for code authors* [online]. CTAN, 2022-09-28 [vid. 2022-10-04]. Dostupné z: `https://ctan.org/pkg/l3kernel`.
12. THE LATEX PROJECT TEAM. *The expl3 package and LATEX3 programming* [online]. CTAN, 2022-09-28 [vid. 2022-10-04]. Dostupné z: `https://ctan.org/pkg/l3kernel`.
13. THE LATEX PROJECT TEAM. *The LATEX3 interfaces* [online]. CTAN, 2022-09-28 [vid. 2022-10-04]. Dostupné z: `https://ctan.org/pkg/l3kernel`.
14. SHARIF, Bonita; MALETIC, Jonathan I. An Eye Tracking Study on camelCase and under_score Identifier Styles. In: *18th International Conference on Program Comprehension*. 2010, s. 196–205. Dostupné z DOI: `10.1109/ICPC.2010.41`.
15. THOMPSON, Michael. *pandoc-discuss*. Re: Error in "cabal install pandoc" [online]. Google Groups [vid. 2022-10-04]. Dostupné z: `https://groups.google.com/g/pandoc-discuss/c/tKB4E7y6H2E/m/OiieKAuWsl4J`.
16. WAGNER, Zdeněk. *Kombinace XML a TEXu při sazbě divadelní hry*. 2017. Dostupné také z: `https://www.cstug.cz/informace/zpravy/2017-11-15-valna-hromada-2017/`.
17. *XML* [online]. ConTEXt Garden, 2022-07-01 [vid. 2022-10-04]. Dostupné z: `https://wiki.contextgarden.net/XML`.
18. MAIER, Denis. *Typesetting XML with ConTEXt*. 2019. Dostupné také z: `https://youtu.be/TEZJ9uZmoJY`.

19. BEN-KIKI, Oren; EVANS, Clark; NET, Ingy döt. *YAML Ain't Markup Language* [online]. 2021-10-01. [vid. 2022-10-05]. Dostupné z: `https://yaml.org/spec/1.2.2/`. Verze 1.2, Revize 1.2.2.
20. GRUBER, John. *Markdown* [online]. Daring Fireball, 2004 [vid. 2022-10-05]. Dostupné z: `https://daringfireball.net/projects/markdown/`.
21. MACFARLANE, John. *Pandoc: a universal document converter* [online]. 2022. [vid. 2022-10-05]. Dostupné z: `https://pandoc.org/`.
22. NOVOTNÝ, Vít. *A Markdown Interpreter for TEX* [online]. CTAN, 2022-10-03 [vid. 2022-10-05]. Dostupné z: `https://ctan.org/pkg/markdown`. Verze 2.17.1.
23. REHÁK, Dominik. Priama sadzba dokumentov rôznych formátov v TeXu pomocou nástroja Pandoc. *Zpravodaj CSTUGu*. 2021, roč. 31, č. 1–4, s. 83–92. Dostupné z DOI: `10.5300/2021-1-4/83`.
24. NOVOTNÝ, Vít et al. Markdown 2.15.0: What's New? *TUGboat*. 2022, roč. 43, č. 1, s. 10–15. Dostupné z DOI: `10.47397/tb/43-1/tb133novotny-markdown`.
25. THE LATEX PROJECT TEAM. *LATEX 2ε font selection* [online]. CTAN, 2021-12 [vid. 2022-10-05]. Dostupné z: `https://ctan.org/pkg/fntguide`.
26. THE LATEX PROJECT TEAM. *The xtemplate package: Prototype document functions* [online]. CTAN, 2022-06-22 [vid. 2022-10-05]. Dostupné z: `https://ctan.org/pkg/xtemplate`.
27. NIEDERBERGER, Clemens. The xtemplate package: An example. *TUGboat*. 2012, roč. 33, č. 3, s. 272–275. Dostupné také z: `https://tug.org/TUGboat/tb33-3/tb105niederberger.pdf`.
28. MITTELBACH, Frank. *LATEX3 architecture and current work in progress*. 2011. Dostupné také z: `https://youtu.be/-lr6KEPGLDs`.
29. MITTELBACH, Frank. *Using LATEX3's xtemplate* [online]. Stack Exchange, 2013-06-06 [vid. 2021-12-06]. Dostupné z: `https://tex.stackexchange.com/a/118015/70941`.
30. TANTAU, Till. *The TikZ and PGF Packages: Manual for version 3.1.9a* [online]. CTAN, 2021 [vid. 2022-10-06]. Dostupné z: `https://ctan.org/pkg/pgf`.
31. CARTER, Sam. *The TikZducks package: using ducks in TikZ* [online]. CTAN, 2020 [vid. 2022-10-06]. Dostupné z: `https://ctan.org/pkg/tikzducks`. Verze 1.5.
32. KIME, Philip; WEMHEUER, Moritz; LEHMAN, Philipp. *The BibLATEX Package: Programmable Bibliographies and Citations* [online]. CTAN, 2022-07-12 [vid. 2022-10-06]. Dostupné z: `https://ctan.org/pkg/biblatex`. Verze 3.18b.
33. NOVOTNÝ, Vít. Príprava Zpravodaje CSTUG. *Zpravodaj CSTUGu*. 2018, roč. 28, č. 1–4, s. 1–10. Dostupné z DOI: `10.5300/2018-1-4/1`.
34. LUPTÁK, Dávid. Sadzba bibliografie podľa normy ISO 690 v systéme LATEX. *Zpravodaj CSTUGu*. 2016, roč. 26, č. 1–4, s. 106–120. Dostupné z DOI: `10.5300/2016-1-4/106`.
35. PERON, Fr. Jacques; LISKA, Urs; SPRINGUEL, Br. Samuel. *lyLuaTEX: Programmable Bibliographies and Citations* [online]. CTAN, 2019-05-27 [vid. 2022-10-06]. Dostupné z: `https://ctan.org/pkg/lyluatex`. Verze 1.0f.
36. NOVOTNÝ, Vít. *Vysokoúrovňové jazyky pro TEX* [online]. GitHub [vid. 2022-10-16]. Dostupné z: `http://github.com/witiko/high-level-languages-for-tex`.

37. DEVLIN, Jacob et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: *NAACL 2019*. 2018. Dostupné také z: https://arxiv.org/abs/1810.04805v2.
38. BROWN, Tom et al. Language Models are Few-Shot Learners. In: LAROCHELLE, H. et al. (ed.). *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2020, sv. 33, s. 1877–1901. Dostupné také z: https://arxiv.org/abs/2005.14165v4.
39. PAPERS WITH CODE CONTRIBUTORS. *Code Generation* [online]. Ed. STOJNIC, Robert et al. Papers with Code [vid. 2022-09-29]. Dostupné z: https://paperswithcode.com/task/code-generation.
40. ZHOU, Yingbo; PORWAL, Utkarsh; KONOW, Roberto. Spelling Correction as a Foreign Language. In: DEGENHARDT, Jon et al. (ed.). *eCOM 2019: The SIGIR 2019 Workshop on eCommerce* [online]. Paris, France, 2019 [vid. 2022-09-29]. CEUR Workshop Proceedings, č. 2410. ISSN 1613-0073. Dostupné z: http://ceur-ws.org/Vol-2410/paper28.pdf.
41. NAGY, Attila; BIAL, Bence; ÁCS, Judit. *Automatic punctuation restoration with BERT models* [online]. Cornell University, 2021 [vid. 2022-09-29]. Dostupné z: https://arxiv.org/abs/2101.07343v1.
42. LU, Jiasen et al. ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks. In: WALLACH, H. et al. (ed.). *NeurIPS 2019*. Curran Associates, Inc., 2019, sv. 32. Dostupné také z: https://arxiv.org/abs/1908.02265v1.

Summary: High-Level Languages for TEX

TEX is the assembly language of digital typesetting, which requires advanced programming skills from authors and designers, and which provides few high-level abstractions to programmers. In this article, I introduce selected markup, programming, and style-sheet languages for TEX, which enable the division of labor between authors, programmers, and designers, and which simplify the process of electronic document preparation. The article transcribes my invited talk at the general assembly of CSTUG on May 14, 2022 [1].

Keywords: high-level languages, programming languages, markup languages, style-sheet languages, ε-TEX, pdfTEX, LuaTEX, LuaMetaTEX, LATEX 2ε, LATEX3, expl3, XML, DocBook, TEI, XHTML, XSLT, CSS, CSL, ConTEXt, HTML, markdown, YAML, Pandoc, Ti*k*Z, BibLATEX, BibLATEXML, LyLuaTEX

Vít Novotný, witiko@mail.muni.cz

Automatically Removing Widows and Orphans with lua-widow-control

Max Chernoff

The lua-widow-control package, for plain LuaTeX/LuaLaTeX/ConTeXt/OpTeX, removes widows and orphans without any user intervention. Using the power of LuaTeX, it does so without stretching any vertical glue or shortening any pages or columns. Instead, lua-widow-control automatically lengthens a paragraph on a page or column where a widow or orphan would otherwise occur.

To use the lua-widow-control package, all that most LaTeX users need do is place `\usepackage{lua-widow-control}` in their preamble. No further changes are required.

Keywords: LuaTeX, widows, orphans

1. Motivation

TeX provides top-notch typesetting: even 40 years after its first release, no other program produces higher quality mathematical typesetting, and its paragraph-breaking algorithm is still state-of-the-art. However, its page breaking is not quite as sophisticated as its paragraph breaking and thus suffers from some minor issues.

Unmodified TeX has only two familiar ways of dealing with widows and orphans: it can either shorten a page by one line, or it can stretch vertical whitespace. TeX was designed for mathematical and scientific typesetting, where a typical page has multiple section headings, tables, figures, and equations. For this style of document, TeX's default behaviour works quite well, since the slight stretching of whitespace between the various document elements is nearly imperceptible; however, for prose or other documents composed almost entirely of paragraphs, there is little vertical whitespace to stretch.

Since no ready-made, fully-automated solution to remove widows and orphans from all types of documents was available, I decided to create lua-widow-control.

2. What are widows and orphans?

2.1. Widows

A "widow" occurs when the majority of a paragraph is on one page or column, but the last line is on the following page or column. It not only looks quite odd

First published in *TUGboat* **43**:1 (Chernoff, 2022), pp. 28–39. Reprinted, with additions and corrections, with permission.

Widow	Orphan
A widow is when a paragraph's last line is placed on a different page than where it begins.	An orphan is when the first line of a paragraph occurs on the page before all the other lines.

Figure 1: The difference between widows and orphans. If we imagine that each box is a different page, then this roughly simulates how widows and orphans appear.

for a lone line to be at the start of the page, but it makes a paragraph harder to read since the separation of a paragraph and its last line disconnects the two, causing the reader to lose context for the widowed line.

2.2. Orphans

An "orphan" occurs when the first line of a paragraph is at the end of the page or column preceding the remainder of the paragraph. They are not as distracting for the reader, but they are still not ideal. Visually, widows and orphans are about equally disruptive; however, orphans tend not to decrease the legibility of a text as much as widows, so many authors choose to ignore them.

See Figure 1 for a visual reference.

2.3. Broken hyphens

"Broken" hyphens occur whenever a page break occurs in a hyphenated word. These are not related to widows and orphans; however, breaking a word across two pages is at least as disruptive for the reader as widows and orphans. TEX identifies broken hyphens in the same ways as widows and orphans, so lua-widow-control treats broken hyphens in the same way.

3. History and etymology

The concept of widows and orphans is nearly as old as printing itself. In *Mechanick exercises* (Moxon, 1683), a printers manual from 1683, we have:

> Nor do good *Compoſiters* account it good Workmanſhip to begin a *Page* with a *Break-line*, unleſs it be a very ſhort *Break*, and cannot be gotten in the foregoing *Page*; but if it be a long *Break*, he will let it be the *Direction-line* of the fore-going *Page*, and *Set* his *Direction* at the end of it. (p. 226)

However, the terms "widow" and "orphan" are much newer.

3.1. Widows

The earliest published source that I could find referencing "widows" in typography is *Webster's New International Dictionary* from 1934. However, no one—not even the editors of the dictionary (Brown, 1948a)—seems to know how it got there. Even then, the definition is somewhat different than it is now:

> widow, n. c. *Print.* A short line or single word carried over from the foot of one column or page to the head of a succeeding column or page. (Brown, 1948a)

Contrast this with the modern definition:

> The stub-ends left when paragraphs end on the first line of a page are called widows. They have a past but not a future, and they look foreshortened and forlorn. (Bringhurst, 2004)

which includes a single lone line of any length.

3.2. Orphans

The term "orphan" is even more confusing. Its initial usage seems to have occurred some time after "widow" (Brown, 1948a), and it is given many contradictory definitions. Most sources define an orphan as a first line at the bottom of the page and a widow as the last line at the top (Bringhurst, 2004; Brown, 1948a; Brown, 1948b; Isambert, 2010; Knuth, 2021; Mittelbach, 2021; Oxford English Dictionary, 2021b; Oxford English Dictionary, 2021c); however, some sources define these two terms as *exact opposites* of each other, with a widow as a first line at the bottom of the page and an orphan as the last line! (Ambrose; Harris, 2007; Brown, 1948a; Hunt, 2020; Oxford English Dictionary, 2021b; Saltz, 2019) This usage is plain wrong; nevertheless, it is sufficiently common that you need to be careful when you see the terms "widow" and "orphan".

Similarly to the term "widow", *The Elements of Typographic Style* (Bringhurst, 2004) provides a succinct definition of the term "club", along with a helpful mnemonic:

> Isolated lines created when paragraphs begin on the last line of a page are known as orphans. They have no past, but they do have a future. (Bringhurst, 2004)

3.3. Clubs

The TEXbook never refers to "orphans" as such; rather, it refers to them as "clubs". This term is remarkably rare: I could only find a *single* source published before *The TEXbook*—a compilation article about the definition of "widow"—that mentions a "club line":

> The Dictionary staff informs me that they have no example of the use of the word widow in the typographical sense. [...]

> Mr. Watson of the technical staff says that the Edinburgh printing houses referred to it as a "clubline". (Brown, 1948a, p. 4)

> To my knowledge, a 'widow', or 'widow-line,' is a short line, forming the end of a paragraph, which is carried over from the foot of a page or column to the top of the succeeding one. […]
>
> To my personal knowledge, in typographical parlance in Edinburgh, Scotland, the 'widow' is called a 'club-line.' (Brown, 1948a, p. 23)

Both quotes above are from separate authors, and they each define a "club" like we define "widow", not an "orphan". In addition, they both mention that the term is only used in Scotland. Even the extensive OED—which lists 17 full definitions and 103 subdefinitions for the noun "club"—doesn't recognize the phrase. (Oxford English Dictionary, 2021a)

I spent a few hours searching through Google Books and my university library catalogue, but I could not find a single additional source. However, Don Knuth—the creator of TEX—read the original article (Chernoff, 2022) and sent me this reply:

> I cannot remember where I found the term "club line". Evidently the books that I scoured in 1977 and 1978 had taught me only that an isolated line, caused by breaking between pages in the midst of a paragraph, was called a "widow"; hence TEX78 had only "`\chpar4`" to change the "`widowpenalty`". Sometime between then and TEX82 I must have come across what appeared to be an authoritative source that distinguished between widows at the beginning of a paragraph and orphans or club lines at the end. I may have felt that the term "orphan" was somewhat pejorative, who knows?[1]

So this (somewhat) resolves the question of where the term "club" came from.

4. Pagination in TEX

Let's move on to looking at how TEX treats these widows and orphans.

4.1. Algorithm

It is tricky to understand how lua-widow-control works if you aren't familiar with how TEX breaks pages and columns. For a full description, you should consult Chapter 15 of *The TEXbook* (Knuth, 2021) ("How TEX Makes Lines into Pages");

[1] Note that this definition is somewhat mistaken. Widows are located either at the *end* of a paragraph, or the beginning of a *page or column*. Likewise, orphans/clubs appear at the *beginning* of a paragraph or at the end of a *page or column*.

however, this goes into much more detail than most users require, so here is a *very* simplified summary of TEX's page breaking algorithm:

TEX fills the page with lines and other objects until the next object will no longer fit. Once no more objects will fit, TEX will align the bottom of the last line with the bottom of the page by stretching any available vertical spaces if (in LATEX) `\flushbottom` is set; otherwise, it will break the page and leave the bottom empty.

However, some objects have penalties attached. Penalties encourage or discourage page breaks from occurring at specific places. For example, LATEX sets a negative penalty before section headings to encourage a page break there; conversely, it sets a positive penalty after section headings to discourage breaking.

To reduce widows and orphans, TEX sets weakly-positive penalties between the first and second lines of a paragraph to prevent orphans, and between the penultimate and final lines to prevent widows.

One important note: once TEX begins breaking a page, it never goes back to modify any content on the page. Page breaking is a localized algorithm, without any backtracking.

4.2. Behaviour

Merely describing the algorithm doesn't allow us to intuitively understand how TEX deals with widows and orphans.

Due to the penalties attached to widows and orphans, TEX tries to avoid them. Widows and orphans with small penalties attached—like LATEX's default values of 150—are only lightly coupled to the rest of the paragraph, while widows and orphans with large penalties—values of 10 000 or more—are treated as infinitely bad and are thus unbreakable. Intermediate values behave just as you would expect, discouraging page breaks proportional to their value.

When TEX goes to break a page, it tries to avoid breaking at a location with a high penalty. How it does so depends on a few settings:

4.2.1. `\flushbottom` and `\normalbottom`

With the settings `\normalbottom` (Plain TEX) or `\flushbottom` (LATEX), TEX is willing to stretch any glue on the page by an amount roughly commensurate to the magnitude of the penalty: for small `\clubpenalty` and `\widowpenalty` values, TEX will only slightly stretch the glue on the page before creating a widow or orphan; for very large penalties, TEX will stretch the glue by a near-infinite amount.

This corresponds to the "Stretch" column in Figure 2. It is the default behaviour of Plain TEX, and of the standard LATEX classes when the `twocolumn` option is given.

4.2.2. `\raggedbottom`

When `\raggedbottom` is set, TEX won't stretch any glue. Instead, for sufficiently high `\clubpenalty` and `\widowpenalty` values, TEX will shorten the page or column by one line in order to prevent the widow or orphan from being created.

This corresponds to the "Shorten" column in Figure 2 and is the default behaviour of the LATEX classes when the `twocolumn` option is not given.

5. `\looseness`

Before we can continue further, we need to discuss one more TEX command: `\looseness`. The following is excerpted from Chapter 14 of *The TEXbook* (Knuth, 2021) ("How TEX Breaks Paragraphs into Lines"):

> If you set `\looseness=1`, TEX will try to make the current paragraph one line longer than its optimum length, provided that there is a way to choose such breakpoints without exceeding the tolerance you have specified for the badnesses of individual lines. Similarly, if you set `\looseness=2`, TEX will try to make the paragraph two lines longer; and `\looseness=-1` causes an attempt to make it shorter. [...]
>
> For example, you can set `\looseness=1` if you want to avoid a lonely "club line" or "widow line" on some page that does not have sufficiently flexible glue, or if you want the total number of lines in some two-column document to come out to be an even number.
>
> It's usually best to choose a paragraph that is already pretty "full", i.e., one whose last line doesn't have much white space, since such paragraphs can generally be loosened without much harm. You might also want to insert a tie between the last two words of that paragraph, so that the loosened version will not end with only one "widow word" on the orphans line; this tie will cover your tracks, so that people will find it hard to detect the fact that you have tampered with the spacing. On the other hand, TEX can take almost any sufficiently long paragraph and stretch it a bit, without substantial harm.

The widow and orphan removal strategy suggested in the second paragraph works quite well; however, it requires manual editing each and every time a page or paragraph is rewritten or repositioned.

6. Alternate removal strategies

There have been a few previous attempts to improve upon TEX's previously discussed widow and orphan-handling abilities; however, none of these have been

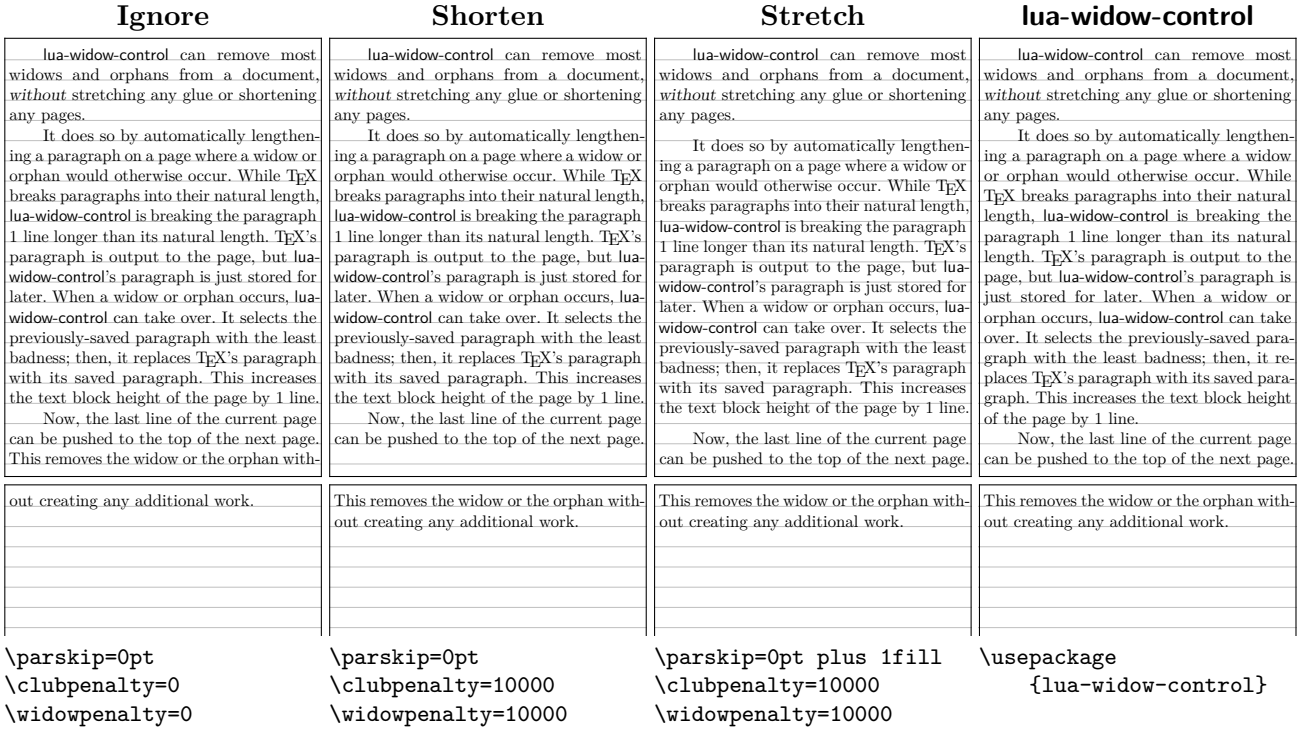


Figure 2: A visual comparison of various automated widow-handling techniques.

able to automatically remove widows and orphans without stretching any vertical glue or shortening any pages.

The articles "Strategies against widows" (Isambert, 2010) and "Managing forlorn paragraph lines (a.k.a. widows and orphans) in LaTeX" (Mittelbach, 2018b) both begin with comprehensive discussions of the methods of preventing widows and orphans. They agree that widows and orphans are bad and ought to be avoided; however, they differ in their solutions. *Strategies* proposes an output routine that reduces the length of facing pages by one line when necessary to remove widows and orphans, while *Managing* proposes that the author manually rewrites or adjusts `\looseness` when needed.

The post *Paragraph callback to help with widows/orphans hand tuning* (jeremie, 2017) contains a file `widow-assist.lua` that automatically detects which paragraphs can be safely shortened or lengthened by one line. The widows-and-orphans package (Mittelbach, 2021) alerts the author to the pages that contain widows or orphans. Combined, these packages make it simple for the author to quickly remove widows and orphans by adjusting the values of `\looseness`; however, it still requires the author to make manual source changes after each revision.

Another article suggests a fully-automated solution to remove widows and orphans (Mittelbach, 2018a). This would seem to offer a complete solution; however, it requires multiple passes, an external tool, and has not yet been publicly released.

lua-widow-control is essentially a combination of `widow-assist.lua` (jeremie, 2017) and widows-and-orphans (Mittelbach, 2021), although its implementation is independent of both: when the `\outputpenalty` value indicates that a widow or orphan has occurred, Lua is used to find a stretchable paragraph. What lua-widow-control mainly adds on top of these packages is automation: it eliminates the requirement for any manual adjustments or changes to your document's source.

7. Visual comparison

Although TeX's page breaking algorithm is reasonably straightforward, it can lead to complex behaviour when widows and orphans are involved. The usual choices, when rewriting is not possible, are to ignore them, stretch some glue, or shorten the page. Figure 2 has a visual comparison of these options, which we'll discuss in the following:

7.1. Ignore

As you can see, the last line of the page is on a separate page from the rest of its paragraph, creating a widow. This is usually highly distracting for the reader, so it is best avoided for the reasons previously discussed.

7.2. Shorten

This page did not leave any widows, but it did shorten the previous page by one line. Sometimes this is acceptable, but usually it looks bad because pages will then have different text-block heights. This can make the pages look quite uneven, especially when typesetting with columns or in a book with facing pages.

7.3. Stretch

This page also has no widows and it has a flush bottom margin. However, the space between each pair of paragraphs had to be stretched.

If this page had many equations, headings, and other elements with natural space between them, the stretched out space would be much less noticeable. TEX was designed for mathematical typesetting, so it makes sense that this is its default behaviour. However, in a page with mostly text, these paragraph gaps look unsightly.

Also, this method is incompatible with grid typesetting, where all vertical glue stretching must be quantised to the height of a line.

7.4. lua-widow-control

lua-widow-control has none of these issues: it eliminates the widows in a document while keeping a flush bottom margin and constant paragraph spacing.

To do so, lua-widow-control lengthened the second paragraph in Figure 2 by one line. If you look closely, you can see that this stretched the interword spaces. This stretching is noticeable when typesetting in a narrow text block, but is mostly imperceptible with larger widths.

lua-widow-control automatically finds the "best" paragraph to stretch, so the increase in interword spaces should almost always be minimal.

8. Installation and standard usage

The lua-widow-control package was first released in October 2021. It is available in the default installations of both MiKTEX and TEX Live, although you will need recent versions of either.

You may also download lua-widow-control manually from either CTAN,[2] the ConTEXt Garden,[3] or GitHub,[4] although it is best if you can install it through your TEX distribution.

[2] ctan.org/pkg/lua-widow-control

[3] modules.contextgarden.net/cgi-bin/module.cgi/action=view/id=127

[4] github.com/gucci-on-fleek/lua-widow-control/releases/latest/

As its name may suggest, lua-widow-control *requires* LuaTEX or LuaMetaTEX regardless of the format used. With that in mind, using lua-widow-control is quite simple:

Plain TEX	`\input lua-widow-control`
OpTEX	`\load[lua-widow-control]`
LATEX	`\usepackage{lua-widow-control}`
ConTEXt	`\usemodule[lua-widow-control]`

And that's usually enough. Most users won't need to do anything else since lua-widow-control comes preconfigured and ready-to-go.

9. Options

Nevertheless, lua-widow-control does have a few options.

lua-widow-control tries very hard to have a "natural" user interface with each format, so how you set an option heavily depends on how you are running lua-widow-control. Also note that not every option is available in every format.

Some general guidelines:

Plain TEX/OpTEX	Specially-named `\lwc`⟨*option*⟩ commands and registers are provided for all options.
LATEX	Options can be set either as package options or at any point in the document with `\lwcsetup`.
ConTEXt	Always use `\setuplwc`.

9.1. Disabling

You may want to disable lua-widow-control for certain portions of your document. You can do so with the following commands:

Plain TEX/OpTEX	`\lwcdisable`
LATEX	`\lwcsetup{disable}`
ConTEXt	`\setuplwc[state=stop]`

This prevents lua-widow-control from stretching any paragraphs that follow. If a page has earlier paragraphs where lua-widow-control was still enabled and a widow or orphan is detected, lua-widow-control will still attempt to remove the widow or orphan.

9.2. Enabling

lua-widow-control is enabled as soon as the package is loaded. If you have previously disabled it, you will need to re-enable it to save new paragraphs.

Plain TEX/OpTEX	`\lwcenable`
LATEX	`\lwcsetup{enable}`
ConTEXt	`\setuplwc[state=start]`

9.3. Automatically disabling

You may want to disable lua-widow-control for certain commands where stretching is undesirable such as section headings. Of course, manually disabling and then enabling lua-widow-control multiple times throughout a document would quickly become tedious, so lua-widow-control provides some options to do this automatically for you.

lua-widow-control automatically patches the default LATEX, ConTEXt, Plain TEX, OpTEX, memoir, KOMA-Script, and titlesec section commands, so you don't need to patch these. Any others, though, you'll need to patch yourself.

Plain TEX/OpTEX	`\lwcdisablecmd{`⟨`\`*macro*⟩`}`
LATEX	`\lwcsetup{disablecmds={`⟨*csnameone*⟩`,`⟨*csnametwo*⟩`}}`
ConTEXt	`\prependtoks\lwc@patch@pre\to\everybefore`⟨*hook*⟩ `\prependtoks\lwc@patch@post\to\everyafter`⟨*hook*⟩

9.4. `\emergencystretch`

lua-widow-control defaults to an `\emergencystretch` value of 3 em for stretched paragraphs, but you can configure this.

lua-widow-control will only use the `\emergencystretch` when it cannot extend a paragraph in any other way, so it is fairly safe to set this to a large value. TEX accumulates badness when `\emergencystretch` is used (Knuth, 1989), so it's pretty rare that a paragraph that requires any `\emergencystretch` will actually be used on the page.

Plain TEX/OpTEX	`\lwcemergencystretch=`⟨*dimension*⟩
LATEX	`\lwcsetup{emergencystretch=`⟨*dimension*⟩`}`
ConTEXt	`\setuplwc[emergencystretch=`⟨*dimension*⟩`]`

9.5. Penalties

You can also manually adjust the penalties that TEX assigns to widows and orphans. Usually, the defaults are fine, but there are a few circumstances where you may want to change them.

Plain TEX/OpTEX	`\widowpenalty=`⟨*integer*⟩
	`\clubpenalty=`⟨*integer*⟩
	`\brokenpenalty=`⟨*integer*⟩
LATEX	`\lwcsetup{ widowpenalty=`⟨*integer*⟩`}`
	`\lwcsetup{orphanpenalty=`⟨*integer*⟩`}`
	`\lwcsetup{brokenpenalty=`⟨*integer*⟩`}`
ConTEXt	`\setuplwc[ widowpenalty=`⟨*integer*⟩`]`
	`\setuplwc[orphanpenalty=`⟨*integer*⟩`]`
	`\setuplwc[brokenpenalty=`⟨*integer*⟩`]`

The value of these penalties determines how much TEX should attempt to stretch glue before passing the widow or orphan to lua-widow-control. If you set the values to 1 (default), TEX will stretch nothing and immediately trigger lua-widow-control; if you set the values to 10 000, TEX will stretch infinitely and lua-widow-control will never be triggered. If you set the value to some intermediate number, TEX will first attempt to stretch some glue to remove the widow or orphan; only if it fails will lua-widow-control come in and lengthen a paragraph. As a special case, if you set the values to 0, both TEX and lua-widow-control will completely ignore the widow or orphan.

lua-widow-control will pick up on the values of `\widowpenalty`, `\clubpenalty`, and `\brokenpenalty` regardless of how you set them, so the use of these dedicated keys is entirely optional.

9.6. `\nobreak` behaviour

When lua-widow-control encounters an orphan, it removes it by moving the orphaned line to the next page. The majority of the time, this is an appropriate solution. However, if the orphan is immediately preceded by a section heading (or `\nobreak`/`\penalty 10000`), lua-widow-control would naïvely separate a section heading from the paragraph that follows. This is almost always undesirable, so lua-widow-control provides some options to configure this.

Plain TEX/OpTEX	`\lwcnobreak{`⟨*value*⟩`}`
LATEX	`\lwcsetup{nobreak=`⟨*value*⟩`}`
ConTEXt	`\setuplwc[nobreak=`⟨*value*⟩`]`

The default value, `keep`, *keep*s the section heading with the orphan by moving both to the next page. The advantage to this option is that it removes the orphan and retains any `\nobreak`s; the disadvantage is that moving the section heading can create a large blank space at the end of the page. The value `split` *split*s up the section heading and the orphan by moving the orphan to the next page while

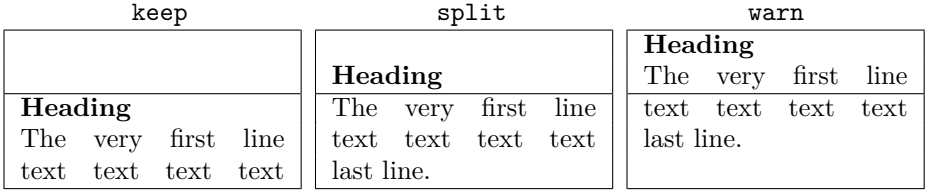


Figure 3: A visual comparison of the `nobreak` option values.

leaving the heading behind. This is usually a bad idea, but exists for the sake of flexibility. The value `warn` causes lua-widow-control to give up on the page and do nothing, leaving an orphaned line. lua-widow-control *warn*s the user so that they can manually remove the orphan.

See Figure 3 for a visual reference.

9.7. Maximum cost

lua-widow-control ranks each paragraph on the page by how much it would "cost" to lengthen that paragraph. By default, lua-widow-control selects the paragraph on the page with the lowest cost; however, you can configure it to only select paragraphs below a selected cost.

If there aren't any paragraphs below the set threshold, then lua-widow-control won't remove the widow or orphan and will instead issue a warning.

Plain TEX/OpTEX	`\lwcmaxcost=`⟨*integer*⟩
LATEX	`\lwcsetup{max-cost=`⟨*integer*⟩`}`
ConTEXt	`\setuplwc[maxcost=`⟨*integer*⟩`]`

Based on my testing, `max-cost` values less than 1 000 cause completely imperceptible changes in interword spacing; values less than 5 000 are only noticeable if you are specifically trying to pick out the expanded paragraph on the page; values less than 15 000 are typically acceptable; and larger values may become distracting. lua-widow-control defaults to an infinite `max-cost`, although the "strict" and "balanced" modes sets the values to 5 000 and 10 000, respectively.

9.8. Draft mode

lua-widow-control has a "draft mode" which shows how lua-widow-control processes pages.

Plain TEX/OpTEX	`\lwcdraft 1`
LATEX	`\lwcsetup{draft}`
ConTEXt	`\setuplwc[draft=start]`

The draft mode has two main features:

First, it colours lines in the document according to their status. Any remaining widows and orphans will be coloured red, any expanded paragraphs will be coloured green, and any lines moved to the next page will be coloured blue.

Second, this draft mode shows the paragraph costs at the end of each paragraph, in the margin.

This draft mode leads to a neat trick: if you don't quite trust lua-widow-control, or you're writing a document whose final version will need to be compilable by both pdfLaTeX and LuaLaTeX, you can load the package with:

```
\usepackage[draft, disable]{lua-widow-control}
```

This way, all the widows and orphans will be coloured red and listed in your log file. When you go through the document to try and manually remove the widows and orphans—whether through the `\looseness` trick or by rewriting certain lines—you can easily find the best paragraphs to modify by looking at the paragraph costs in the margins. If you're less cautious, you can compile your document with lua-widow-control enabled as normal and inspect all the green paragraphs to see if they look acceptable to you.

You can also toggle the paragraph colouring and the cost displays individually:

Plain TeX/OpTeX	`\lwcshowcosts 1`
	`\lwcshowcolours 0`
LaTeX	`\lwcsetup{showcosts=true}`
	`\lwcsetup{showcolours=false}`
ConTeXt	`\setuplwc[showcosts=start]`
	`\setuplwc[showcolours=stop]`

10. Presets

As you can see, lua-widow-control provides quite a few options. Luckily, there are a few presets that you can use to set multiple options at once. These presets are a good starting point for most documents, and you can always manually override individual options.

These presets are only available for LaTeX and ConTeXt.

LaTeX	`\lwcsetup{`⟨*preset*⟩`}`
ConTeXt	`\setuplwc[`⟨*preset*⟩`]`

10.1. `default`

If you use lua-widow-control without any options, it defaults to this preset. In default mode, lua-widow-control takes all possible measures to remove widows

and orphans and will not attempt to stretch any vertical glue. This usually removes $>95\%$ of all possible widows and orphans. The catch here is that this mode is quite aggressive, so it often leaves behind some fairly "spacey" paragraphs.

This mode is good if you want to remove (nearly) all widows and orphans from your document, without fine-tuning the results.

10.2. `strict`

lua-widow-control also offers a strict mode. This greatly restricts lua-widow-control's tolerance and makes it so that it will only lengthen paragraphs where the change will be imperceptible.

The caveat with strict mode is that—depending on the document—lua-widow-control will be able to remove less than a third of the widows and orphans. For the widows and orphans that can't be automatically removed, a warning will be printed to your terminal and log file so that a human can manually fix the situation.

This mode is good if you want the best possible typesetting and are willing to do some manual editing.

10.3. `balanced`

Balanced mode sits somewhere between default mode and strict mode. This mode first lets TEX stretch a little glue to remove the widow or orphan; only if that fails will it then trigger lua-widow-control. Even then, the maximum paragraph cost is capped. Here, lua-widow-control can usually remove 90% of a document's potential widows and orphans, and it does so while making a minimal visual impact.

This mode is recommended for most users who care about their document's typography. This mode is not the default since it doesn't remove all widows and orphans: it still requires a little manual intervention.

11. Compatibility

The lua-widow-control implementation is almost entirely in Lua, with only a minimal TEX footprint. It doesn't modify the output routine or `\everypar` and it doesn't insert any whatsits. This means that it should be compatible with nearly any TEX package, class, and format. Most changes that lua-widow-control makes are not observable on the TEX side.

However, on the Lua side, lua-widow-control modifies much of a page's internal structure. This should not affect any TEX code; however, it may surprise Lua code that modifies or depends on the page's low-level structure. This does not affect Plain TEX or LATEX where even most Lua-based packages don't depend on

Table 2: lua-widow-control options set by each mode.

Option	default	balanced	strict
max-cost	∞	10000	5000
emergencystretch	3em	1em	0pt
nobreak	keep	keep	warn
widowpenalty	1	500	1
orphanpenalty	1	500	1
brokenpenalty	1	500	1

the node list structure. ConTEXt *does* depend on this internal node structure; however, I have carefully tested the package to ensure that this causes no issues.

Finally, keep in mind that adding lua-widow-control to a document will almost certainly change its page break locations.

11.1. Formats

lua-widow-control runs on all known LuaTEX-based formats: Plain LuaTEX, LuaLATEX, ConTEXt MkIV, and OpTEX. Unless otherwise documented, all features should work equally well in all formats.

lua-widow-control is also fully-compatible with the LuaMetaTEX-based formats: ConTEXt MkXL/LMTX, LuaMetaLATEX, and LuaMetaPlain (Krüger, 2022). ConTEXt MkXL works equally well as ConTEXt MkIV and LuaLATEX; however, LuaMetaLATEX and LuaMetaPlain support is still quite early. All features should work, although there are still a few minor bugs.

All told, lua-widow-control supports 7 different format/engine combinations.

11.2. Columns

Since TEX and the formats implement column breaking and page breaking through the same internal mechanisms, lua-widow-control removes widows and orphans between columns just as it does with widows and orphans between pages.

lua-widow-control is known to work with the LATEX class option `twocolumn` and the two-column output routine from Chapter 23 of *The TEXbook* (Knuth, 2021).

11.3. Performance

lua-widow-control runs entirely in a single pass, without depending on any `.aux` files or the like. Thus, it shouldn't meaningfully increase compile times. Although

lua-widow-control internally breaks each paragraph twice, modern computers break paragraphs near-instantaneously, so you are not likely to notice any slowdown.

lua-widow-control has been carefully tested to ensure that there are no memory leaks, so lua-widow-control can now easily compile documents $> 10\,000$ pages long.

11.4. ε-TEX penalties

Knuth's original TEX has three basic line penalties: `\interlinepenalty`, which is inserted between all lines; `\clubpenalty`, which is inserted after the first line; and `\widowpenalty`, which is inserted before the last line. The ε-TEX extensions (The $\mathcal{NTS}$ Team, 1998) generalize these commands with a syntax similar to `\parshape`: with `\widowpenalties` you can set the penalty between the last, second last, and nth last lines of a paragraph; `\interlinepenalties` and `\clubpenalties` behave similarly.

The lua-widow-control package makes no explicit attempts to support these new `-penalties` commands. Specifically, if you give a line a penalty that matches either `\widowpenalty` or `\clubpenalty`, lua-widow-control will treat the lines exactly as it would a widow or orphan. So while these commands won't break lua-widow-control, they are likely to lead to some unexpected behaviour.

12. Short last lines

When lengthening a paragraph with `\looseness`, it is common advice to insert ties (`~`) between the last few words of the paragraph to avoid overly-short last lines (Knuth, 2021). lua-widow-control does this automatically, but instead of using ties or `\hbox`es, it uses the `\parfillskip` parameter (Knuth, 2021; Wermuth, 2018; Olšák, 1997). When lengthening a paragraph (and only when lengthening a paragraph—remember, lua-widow-control doesn't interfere with TEX's output unless it detects a widow or orphan), lua-widow-control sets `\parfillskip` to `0.75\hsize plus 0.05\hsize minus 0.75\hsize`. This normally makes the last line of a paragraph be at least 20% of the overall paragraph's width, thus preventing ultra-short lines.

13. How it works

lua-widow-control uses a fairly simple algorithm to eliminate widows and orphans, but there are a few subtleties.

13.1. Setup

lua-widow-control sets the `\clubpenalty`, `\widowpenalty`, and `\brokenpenalty` parameters to sentinel values of 1. This will signal to lua-widow-control when a widow or orphan occurs, yet it is small enough that it won't stretch any glue.

lua-widow-control also enables LuaTEX's microtypographic extensions (Thành, 2001). This isn't strictly necessary; however, it significantly increases the number of paragraphs that can be acceptably "loosened".

That is all that happens on the TEX end. The rest of lua-widow-control is pure Lua.

13.2. Paragraph breaking

First, lua-widow-control hooks into the paragraph breaking process, before any output routines or page breaking.

Before a paragraph is broken by TEX, lua-widow-control grabs the unbroken paragraph. Then lua-widow-control breaks the paragraph one line longer than its natural length and stores it for later. It does this in the background, *without* interfering with how TEX breaks paragraphs into their natural length.

After TEX has broken its paragraph into its natural length, lua-widow-control appears again. Before the broken paragraph is added to the main vertical list, lua-widow-control "tags" the first and last nodes of the paragraph using a LuaTEX attribute. These attributes associate the previously-saved lengthened paragraph with the naturally-typeset paragraph on the page.

13.3. Page breaking

lua-widow-control intercepts `\box255` (the `\vbox` output by TEX) immediately before the output routine runs, after all the paragraphs have been typeset.

First, lua-widow-control looks at the `\outputpenalty` of the page or column. If the page was broken at a widow or orphan, the `\outputpenalty` will be equal to either the `\widowpenalty` or the `\clubpenalty`. If the `\outputpenalty` does not indicate a widow or orphan, lua-widow-control will stop and return `\box255` unmodified to the output routine, and TEX continues as normal.

Otherwise, we assume that we have a widow or orphan on the page, meaning that we should lengthen the page by 1 line. We iterate through the list of saved paragraphs to find the lengthened paragraph with the least cost. After we've selected a good paragraph, we traverse through the page to find the original version of this paragraph—the one that unmodified TEX originally typeset. Having found the original paragraph, we splice in the lengthened paragraph in place of the original.

Since the page is now 1 line longer than it was before, we pull the last line off the page to bring it back to its original length, and place that line onto the top of TEX's "recent contributions" list. When the next page begins, this line will be inserted before all other paragraphs, right at the top. Now, we can return the new, widow-free page (updated `\box255`) to the output routine, which proceeds as normal.

13.4. Footnotes

Earlier versions of lua-widow-control completely ignored inserts. This meant that if a moved line had associated footnotes, lua-widow-control would move the "footnote mark" but not the associated "footnote text". lua-widow-control now handles footnotes correctly through the mechanism detailed in the next section.

13.4.1. Inserts

Before we go into the details of how lua-widow-control handles footnotes, we need to look at what footnotes actually are to TEX. Every `\footnote` command ultimately expands to something like `\insert`⟨*class*⟩`{`⟨*content*⟩`}`, where ⟨*class*⟩ is an insertion class number, defined as `\footins` in this case (in Plain TEX and LATEX). Inserts can be found in horizontal mode (footnotes) or in vertical mode (`\topins` in Plain TEX and floats in LATEX), but they cannot be inside boxes. Each of these insert types is assigned a different class number, but the mechanism is otherwise identical. lua-widow-control treats all inserts identically, although it safely ignores vertical mode inserts since they are only ever found between paragraphs.

But what does `\insert` do exactly? When TEX sees an `\insert` primitive in horizontal mode (when typesetting a paragraph), it does two things: first, it processes the insert's content and saves it invisibly just below the current line. Second, it effectively adds the insert content's height to the height of the material on the current page. Also, for the first insert on a page, the glue in `\skip`⟨*class*⟩ is added to the current height. All this is done to ensure that there is sufficient room for the insert on the page whenever the line is output onto the page.

If there is absolutely no way to make the insert fit on the page—say, if you placed an entire paragraph in a footnote on the last line of a page—then TEX will begrudgingly "split" the insert, placing the first part on the current page and "holding over" the second part until the next page.

There are some other TEXnicalities involving `\count`⟨*class*⟩ and `\dimen`⟨*class*⟩, but they mostly don't affect lua-widow-control. See Chapter 15 in *The TEXbook* or another reference for all the details.

After TEX has chosen the breakpoints for a paragraph, it adds the chosen lines one by one to the current page. Whenever the accumulated page height is "close enough" to the target page height (normally `\vsize`) the `\output` token list (often called the "output routine") is expanded.

But before `\output` is called, TEX goes through the page contents and moves the contents of any saved inserts into `\vbox`es corresponding to the inserts' classes, namely `\box`⟨*class*⟩, so `\output` can work with them.

And that's pretty much it on the engine side. Actually placing the inserts on the page is reserved for the output routine, which is defined by the format. This

too is a complicated process, although thankfully not one that lua-widow-control needs to worry about.

13.4.2. LuaMetaTEX

The LuaMetaTEX engine treats inserts slightly differently than traditional TEX engines. The first major difference is that insertions have dedicated registers; so instead of `\box`⟨*class*⟩, LuaMetaTEX has `\insertbox`⟨*class*⟩; instead of `\count`⟨*class*⟩, LuaMetaTEX has `\insertmultiplier`⟨*class*⟩; etc. The second major difference is that LuaMetaTEX will pick up inserts that are inside of boxes, meaning that placing footnotes in things like tables or frames should mostly just work as expected.

There are also a few new parameters and other minor changes, but the overall mechanism is still quite similar to traditional TEX.

13.4.3. Paragraph breaking

As stated in the original article (Chernoff, 2022), lua-widow-control intercepts TEX's output immediately before the output routine. However, this is *after* all the inserts on the page have been processed and boxed. This is a bit of a problem because if we move a line to the next page, we need to move the associated insert; however, the insert is already gone.

To solve this problem, immediately after TEX has naturally broken a paragraph, lua-widow-control copies and stores all its inserts. Then, lua-widow-control tags the first element of each line (usually a glyph) with a LuaTEX attribute that contains the indices for the first and last associated insert. lua-widow-control also tags each line inside the insert's content with its corresponding index so that it can be found later.

13.4.4. Page breaking

Here, we follow the same algorithm as in the original article (Chernoff, 2022). However, when we move the last line of the page to the next page, we first need to inspect the line to see if any of its contents have been marked with an insert index. If so, we need to move the corresponding insert to the next page. To do so, we unpack the attributes value to get all the inserts associated with this line.

Using the stored insert indices and class, we can iterate through `\box`⟨*class*⟩ and delete any lines that match one of the current line's indices. We also need to iterate through the internal TEX box `hold_head`—the box that holds any inserts split onto the next page—and delete any matching lines. We can safely delete any of these lines since they are still stored in the original `\insert` nodes that we copied earlier.

Now, we can retrieve all of our previously-stored inserts and add them to the next page, immediately after the moved line. Then, when TEX builds that page, it will find these inserts and move their contents to the appropriate boxes

14. Choosing the "best" paragraph

As we discussed previously, lua-widow-control lengthens the paragraph with the lowest cost. However, assigning a cost to each paragraph is not quite as simple as it sounds. Before we look at how lua-widow-control assigns costs, let's look at how TEX scores paragraphs when breaking them naturally.

14.1. How TEX scores paragraphs

All glue in TEX has a certain natural size: the size that it would be in an ideal scenario. However, most glue also has stretch and shrink components so that the glue can change in size to adapt to its surroundings. For each line, TEX individually sums the total stretch/shrink for the line and the stretch/shrink that was actually used. We define the stretch/shrink ratio r as the quotient of the stretch/shrink used and the stretch/shrink available. Then the badness b of a line is approximately defined as

$$b = 100r^3.$$

This is the badness referenced in the commonly-seen `Underfull \hbox (badness 1234)` warnings that TEX produces.

TEX calculates the badness for each line individually; however, we also need to assess the paragraph as a whole. To do so, TEX defines the demerits for a whole paragraph d as approximately[5] the sum of the squared badnesses for each line. The natural paragraph that TEX breaks is the one that minimizes d.

One important thing to realize is that demerits grow incredibly fast: demerits are proportional to the *sixth* power of glue stretch. This means that you can expect to see extremely large demerit values, even for a relatively "good" paragraph.

14.2. Possible cost functions

Now, let's return to how lua-widow-control assigns costs to each paragraph. This is surprisingly more complicated than it sounds, so we'll go through a few possible cost functions first.

Here, we use c for the cost of a paragraph, d for the total demerits, and l for the number of lines (`\prevgraf`).

[5]We ignore any additional demerits or penalties that TEX may add.

14.2.1. The original implementation
The original implementation of lua-widow-control used the simple cost function

$$c = d.$$

This cost function works reasonably well, but has one major issue: it doesn't take into account the number of lines in the paragraph. The demerits for a paragraph is the sum of the demerits for each line. This means this cost function will prefer using shorter paragraphs since they tend to have fewer demerits. However, long paragraphs tend to have much more available glue stretch, so this strategy can lead to suboptimal solutions.

14.2.2. Scaling by the number of lines
Once I realized this issue, I tried correcting it by dividing by the number of lines in the paragraph to get the average demerits instead of the total demerits:

$$c = \frac{d}{l}$$

This works better than the previous function, but still has an issue. If we have a fairly bad ten-line paragraph with total demerits $10d$ and an almost-equally bad two-line paragraph with total demerits $2d + 1$, then by this cost function, the ten-line paragraph will have a lower cost and will be chosen. This means that our page now has ten bad lines instead of two bad lines, which is not ideal.

14.2.3. Current implementation
Our first cost function, $c = dl^0$, doesn't consider the number of lines at all, while our second cost function, $c = dl^{-1}$, considers the number of lines too much. Splitting the difference between the two functions, we get the current implementation:

$$c = \frac{d}{\sqrt{l}}$$

This solves the issue with the previous function, but it adds a new issue: given a short paragraph with a large number of demerits per line and a long paragraph with fairly few average demerits per line, this function will often choose the shorter line. Although this sounds bad, in practice it gives much better results since very bad short paragraphs are *much* less noticeable than slightly bad long paragraphs.

Of course, this new function may still not be quite perfect. lua-widow-control uses the `lwc.paragraph_cost(demerits, lines)` Lua function to calculate a paragraph's cost; if you want, you can redefine this function to anything that you want.

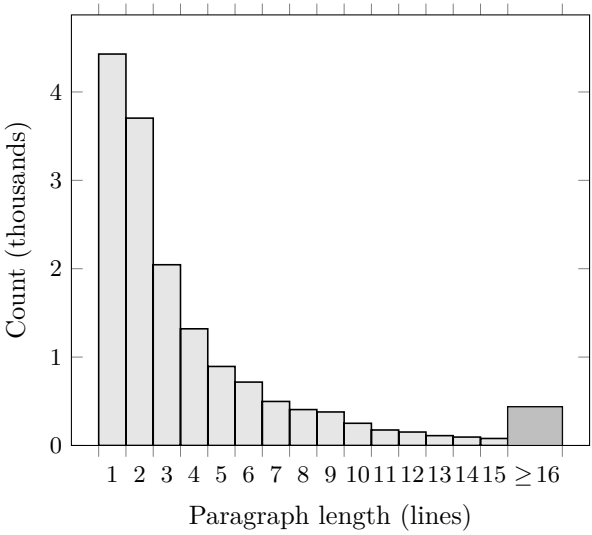


Figure 4: Histogram of natural paragraph lengths in the sample text.

15. Quantitative analysis

Let's look at some statistics for lua-widow-control. For testing, I downloaded the top ten books on *Project Gutenberg*,[6] converted them to LaTeX using pandoc, concatenated them into a single article file, and compiled twice. This gives us a PDF with 1 381 pages, 15 692 paragraphs, 61 865 lines, and 399 widows and orphans (if they aren't removed).

This is a fairly challenging test: almost every third page has a widow or orphan, over half of the paragraphs have two lines or fewer, and the text block is set to the fairly wide article defaults. An average document is much less challenging for lua-widow-control, so we can consider this to be a worst-case scenario.

15.1. Widows and orphans removed

When we run LaTeX with its default settings on the file, 179 (47%) of the widows and orphans are removed. When we add lua-widow-control with default settings, we remove 392 (98%). Switching to strict mode, we can only remove 52 (13%) of the widows and orphans. In balanced mode, we remove 348 (87%). See Figure 5 for a visual comparison.

[6] *Frankenstein*, *Pride and Prejudice*, *Alice's Adventures in Wonderland*, *The Great Gatsby*, *The Adventures of Sherlock Holmes*, *Simple Sabotage Field Manual*, *A Tale of Two Cities*, *The Picture of Dorian Gray*, *Moby Dick*, and *A Doll's House*.

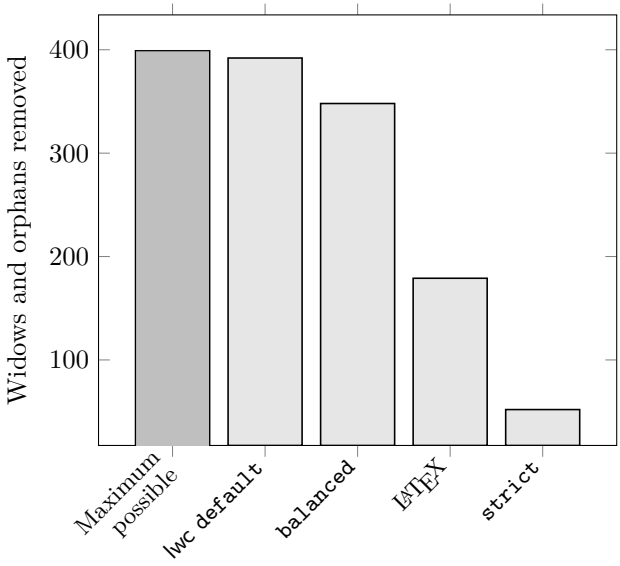


Figure 5: The number of widows and orphans removed by each method.

15.2. Paragraph costs

The last section showed us that lua-widow-control is quite effective at removing widows and orphans, so now let's look at the paragraphs that lua-widow-control expands. As TEX processes a document, lua-widow-control is recording the costs for the naturally-broken and expanded versions of each paragraph in the document. Costs don't mean that much on their own, but a lower cost is always better.

As you can see in Figure 6, the lengthened paragraphs tend to have *much* higher costs than the naturally-broken paragraphs. This is not surprising, since (as we've seen) a paragraph's demerits scale with the sixth power of glue stretch, so even a small amount of glue stretch can cause a huge increase in demerits.

The empty space on the left of the "long" line is from the paragraphs that lua-widow-control was unable to lengthen at any cost. LuaTEX assigns these paragraphs zero demerits, so they disappear on a logarithmic plot.

15.3. Lengthening vs. shortening paragraphs

Figure 7 shows the number of paragraphs that lua-widow-control could potentially stretch or shrink. The one-line paragraphs are broken out separately since this test sample has an anomalous number of them. Otherwise, we can see that lua-widow-control is capable of stretching the majority of paragraphs.

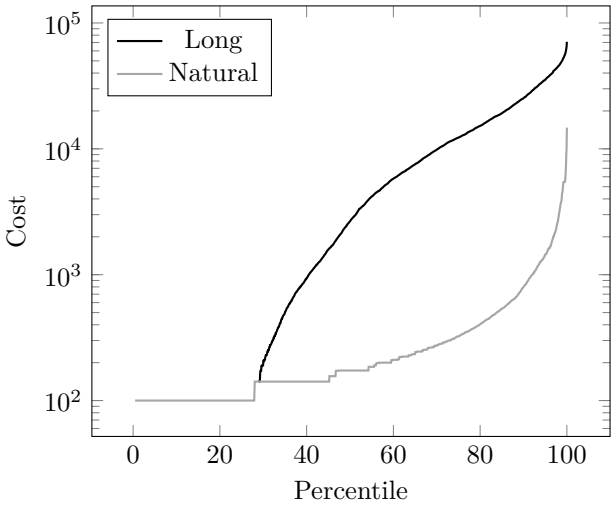


Figure 6: Paragraph costs by percentile rank for naturally-broken and one-line lengthened paragraphs.

We can also see that of non-single-line paragraphs, only about 8% of paragraphs can only be shrunk (the last segment of Figure 7), and this is in a document where 13% of paragraphs have at least eight lines. Most documents rarely have such long paragraphs, and it is these long paragraphs that are the easiest to shrink.

Because of this, lua-widow-control doesn't even attempt to shrink paragraphs; it only stretches them.

16. Known issues

lua-widow-control is quite stable these days. At this point, all *known* bugs have been resolved; some bugs certainly still remain, but I'd feel quite confident using lua-widow-control in your everyday documents. There are, however, some fundamental limitations due to how lua-widow-control operates:

- When a three-line paragraph is at the end of a page forming a widow, lua-widow-control will remove the widow; however, it will leave an orphan. This issue is inherent to any process that removes widows through paragraph expansion and is thus unavoidable. Orphans are considered to be better than widows (Bringhurst, 2004), so this is still an improvement.
- Sometimes a widow or orphan cannot be eliminated because no paragraph has enough stretch. Sometimes this can be remediated by increasing lua-widow-control's `\emergencystretch`; however, some pages just don't have any suitable paragraph.

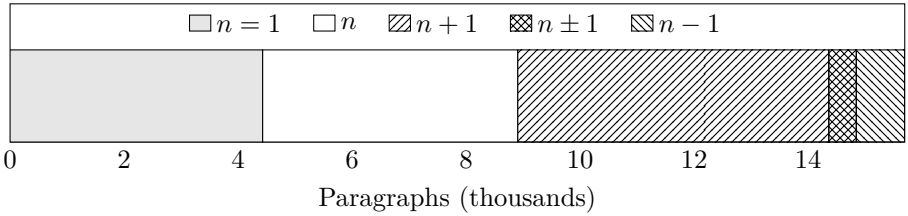


Figure 7: The number of paragraphs in the test sample that (respectively) have exactly one line, cannot be stretched or shrunk, can be only stretched by one line, can be either stretched or shrunk, and can be only shrunk.

Long paragraphs with short words tend to be stretchier than short paragraphs with long words since these long paragraphs have more interword glue. Narrow columns also stretch more easily than wide columns since you need to expand a paragraph by less to make a new line.

- lua-widow-control only attempts to expand paragraphs on a page with a widow or orphan. A global system like in "A general framework for globally optimized pagination" (Mittelbach, 2018a) would solve this; however, this is both NP-complete (Plass, 1981) and impossible to solve in a single pass. Very rarely would such a system remove widows or orphans that lua-widow-control cannot.
- lua-widow-control won't properly move footnotes if there are multiple different "classes" of inserts on the same line. To the best of my knowledge, this shouldn't happen in any real-world documents. If this happens to be an issue for you, please let me know; this problem is relatively easy to fix, although it will add considerable complexity for what I think isn't a real issue.

17. Conclusion

All this probably makes lua-widow-control look quite complicated, and this is true to some extent. However, this complexity is hidden from the end user: as stated at the outset, most users merely need to place `\usepackage{lua-widow-control}` in their LaTeX document preamble, and lua-widow-control will remove all the troublesome widows and orphans, without needing any manual intervention.

Should you have any issues, questions, or suggestions for lua-widow-control, please visit the project's GitHub page: `https://github.com/gucci-on-fleek-lua-widow-control`. Any feedback is greatly appreciated!

References

AMBROSE, G.; HARRIS, P., 2007. *The Layout Book*. Bloomsbury Academic. Advanced Level Series. ISBN 9782940373536.

BRINGHURST, R., 2004. *The Elements of Typographic Style*. 3rd. Hartley & Marks.

BROWN, Karl, 1948a. The Typographical Widow: Who is she? What is she? *Bulletin of the New York Public Library*. Vol. 52, no. 1, pp. 3–25. Available from: `https://hdl.handle.net/2027/uc1.b3310084`.

BROWN, Karl, 1948b. The Typographical Widow: Encore: Encore. *Bulletin of the New York Public Library*. Vol. 52, no. 9, pp. 458–466. Available from: `https://hdl.handle.net/2027/uc1.b3310084`.

CHERNOFF, Max, 2022. Automatically removing widows and orphans with `lua-widow-control`. *TUGboat*. Vol. 43, no. 1, pp. 28–39. Available from DOI: `10.47397/tb/43-1/tb133chernoff-widows`.

HUNT, R., 2020. *Advanced Typography: From Knowledge to Mastery*. Bloomsbury Publishing. ISBN 9781350055926.

ISAMBERT, Paul, 2010. Strategies against widows. *TUGboat*. Vol. 31, no. 1, pp. 12–17. ISSN 0896-3207. Available from: `https://tug.org/TUGboat/tb31-1/tb97isambert.pdf`.

JEREMIE, 2017. *Paragraph callback to help with widows/orphans hand tuning* [online]. 2017-07. [visited on 2022-11-08]. Available from: `https://tex.stackexchange.com/q/372062`.

KNUTH, Donald E., 1989. The new versions of TEX and METAFONT. *TUGboat*. Vol. 10, no. 3, pp. 325–328. ISSN 0896-3207. Available from: `https://tug.org/TUGboat/tb10-3/tb25knut.pdf`.

KNUTH, Donald E., 2021. *The TEXbook*. Addison–Wesley.

KRÜGER, Marcel, 2022. *luametalatex* [online]. 2022-10. [visited on 2022-11-08]. Available from: `https://github.com/zauguin/luametalatex`.

MITTELBACH, Frank, 2018a. A general framework for globally optimized pagination. *Computational Intelligence*. Vol. 35, no. 2, pp. 242–284. Available from: `https://doi.org/10.1111/coin.12165`.

MITTELBACH, Frank, 2018b. Managing forlorn paragraph lines (a.k.a. widows and orphans) in LATEX. *TUGboat*. Vol. 39, no. 3, pp. 246–251. ISSN 0896-3207. Available from: `https://tug.org/TUGboat/tb39-3/tb123mitt-widows.pdf`.

MITTELBACH, Frank, 2021. *The widows-and-orphans package* [online]. 2021-03. [visited on 2022-11-08]. Available from: `https://ctan.org/pkg/widows-and-orphans`.

MOXON, Joseph, 1683. *Mechanick exercises: The doctrine of handy-works applied to the art of printing*. Vol. 2. London. Available from: `https://archive.org/details/mechanickexercis00moxo_0`.

OLŠÁK, Petr, 1997. *TEXbook naruby. [TEXbook inside out]*. Brno, Czech Republic: Konvoj. ISBN 80-85615-64-9. Available from: `https://petr.olsak.net/ftp/olsak/tbn/tbn.pdf`.

OXFORD ENGLISH DICTIONARY, 2021a. *club, n.* [online]. Oxford University Press, 2021-09 [visited on 2022-11-08]. Available from: `https://www.oed.com/view/Entry/34788`.

OXFORD ENGLISH DICTIONARY, 2021b. *line at end of paragraph* [online]. Oxford University Press, 2021-12 [visited on 2022-11-08]. Available from: https://www.oed.com/view/th/class/195380.

OXFORD ENGLISH DICTIONARY, 2021c. *widow, n.* [online]. Oxford University Press, 2021-12 [visited on 2022-11-08]. Available from: https://www.oed.com/view/Entry/228912.

PLASS, Michael Frederick, 1981. *Optimal pagination techniques for automatic typesetting systems*. Available from: https://tug.org/docs/plass/plass-thesis.pdf. PhD thesis. Stanford University.

SALTZ, I., 2019. *Typography Essentials Revised and Updated: 100 Design Principles for Working with Type*. Rockport Publishers. ISBN 9781631596483.

THÀNH, Hàn Thế, 2001. *Micro-typographic extensions to the TEX typesetting system*. Brno. Available from: http://www.pragma-ade.nl/pdftex/thesis.pdf. PhD thesis. The Faculty of Informatics, Masaryk University.

THE $\mathcal{NTS}$ TEAM, 1998. *The ε-TEX manual* [online]. [visited on 2022-11-08]. Available from: https://ctan.org/pkg/etex.

WERMUTH, Udo, 2018. Experiments with \parfillskip. *TUGboat*. Vol. 39, no. 3, pp. 276–303. ISSN 0896-3207. Available from: https://tug.org/TUGboat/tb39-3/tb123wermuth-parfillskip.pdf.

Automatické odstraňování vdov a sirotků pomocí balíčku lua-widow-control

Balíček lua-widow-control pro LuaTEX/LuaLATEX/ConTEXt/OpTEX odstraňuje vdovy a sirotky bez dalšího zásahu uživatele. Využívá přitom sílu LuaTEXu a přitom nenatahuje žádné vertikální mezery a ani nezkracuje stránky nebo sloupce. Namísto toho balíček automaticky prodlužuje některý z odstavců na té stránce nebo sloupci, kde by se vdova nebo sirotek vyskytli.

Pro použití balíčku postačí většině uživatelů LATEXu uvést v preambuli dokumentu \usepackage{lua-widow-control}. Žádné další změny v dokumentu nejsou zapotřebí.

Klíčová slova: LuaTEX, vdova, sirotek

Max Chernoff, mseven at telus dot net

Mělo by to fungovat XII

Peter Wilson

Článek ukazuje několik způsobů, jak lze v LaTeXu vytvořit rámeček kolem textu, přičemž uvnitř rámečku je možné provést stránkový zlom. Jsou popsána makra z balíčku framed a také je ukázáno, jak je možné na základě tohoto balíčku vytvořit vlastní rámečky.

Klíčová slova: Rámečky, balíček framed, LaTeX

> Fame is no plant that grows on mortal soil,
> Nor in the glistering foil
> Set off to the world, nor in broad rumour lies:
> But lives and spreads aloft by those pure eyes
> And perfect witness of all-judging Jove;
> As he pronounces lastly on each deed,
> Of so much fame in heaven expect thy meed.
>
> *Lycidas, Elegy on a Friend drowned in the Irish Channel, 1637*
> John Milton

Cílem tohoto seriálu je ukázat čtenáři krátké kousky kódu, které mohou vyřešit některé z jeho problémů. Doufám, že situaci ještě více nezkomplikuji v důsledku mých chyb. Opravy, poznámky a návrhy na změny budou vždy vítány.

> I wuz framed!
>
> *Tradiční obhajoba zločince*

1. Rámečky

Často na `comp.text.tex` padají dotazy, jak lze kolem textu umístit rámeček nebo jak podbarvit nějaký blok textu. Tyto problémy částečně řeší makro `\fbox`, které vloží svůj obsah do rámečku, nebo makro `\colorbox` z balíčku color nebo xcolor, které svůj obsah podbarví. Tato makra však neumožňují uvnitř rámečku provést stránkový zlom.[1] Někdy však potřebujeme zarámovat text, který bude stránkový zlom obsahovat. V tom případě lze použít balíček framed [2] Donalda Arseneaua. Tento balíček definuje čtyři prostředí, která si postupně ukážeme.

Z anglického originálu *Glisterings* [1] přeložil Jan Šustek.

[1] Tato makra pracují v horizontálním módu, kde svůj argument nejdříve uzavřou do `\hbox`u. Proto v nich není možné provést řádkový ani stránkový zlom. (pozn. překl.)

doi: 10.5300/2022-1-4/77

Prostředí `framed` z balíčku framed vloží svůj obsah do rámečku. Na rozdíl od jiných podobných maker a prostředí, toto prostředí umožňuje uvnitř rámečku provést stránkový zlom.[2]

Dalším prostředím je `shaded`. Toto prostředí podbarví svůj obsah, přičemž opět v něm umožňuje provést stránkový zlom. Před použitím prostředí je nutné nadefinovat barvu `shadecolor`. V této ukázce jsem použil barvu

```
\definecolor{shadecolor}{gray}{0.8}
```

Makro `\definecolor` je definováno v balíčku color nebo xcolor.

Prostředí `snugshade` je podobné jako prostředí `shaded`. Také podbarví svůj obsah a je možné v něm provést stránkový zlom. V této ukázce jsem použil barvu

```
\definecolor{shadecolor}{gray}{0.9}
```

Rozdíl je ten, že v prostředí `snugshade` podbarvení přesahuje okraj textu jen o málo.[3]

Prostředí `leftbar` vloží nalevo od svého obsahu svislou čáru. Jako všechny tři předchozí prostředí, i toto umožňuje uvnitř svého obsahu provést stránkový zlom.

Balíček framed může být užitečný i v případě, kdy víme, že ke stránkovému zlomu nemůže dojít. Týká se to například prostředí `figure`, kde můžeme chtít obrázek zarámovat, abychom jej opticky více odlišili od okolního textu. Obrázek 1 jsme vytvořili pomocí následujícího kódu.

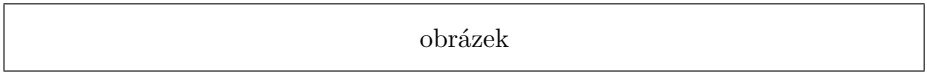
```
\begin{figure}
  \begin{framed}
    \centering
    zarámovaný obrázek
  \end{framed}
  \caption{Ukázka zarámovaného obrázku}
  \label{obr1}
\end{figure}
```

Podobně můžeme zarámovat obrázek 2 včetně popisku.

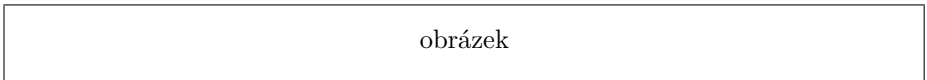
```
\begin{figure}
  \begin{framed}
```

[2] Jak jistě čtenáři pochopili, toto prostředí pracuje ve vertikálním módu. (pozn. překl.)

[3] Konkrétně lze tento přesah nastavit v registru `\fboxsep` podobně jako na řádku 30. (pozn. překl.)



Obrázek 1: Ukázka zarámovaného obrázku



Obrázek 2: Ukázka zarámovaného obrázku včetně popisku

```
    \centering
    obrázek
    \caption{Ukázka zarámovaného obrázku včetně popisku}
    \label{obr2}
  \end{framed}
\end{figure}
```

Jestliže by úplný rámeček kolem obrázku byl příliš výrazný, můžeme použít namísto rámečku pouze vodorovné linky. K tomu nepotřebujeme žádný balíček. Aby byl příklad názornější, použil jsem v obrázku 3 velmi tlusté linky.

```
\begin{figure}
  \centering
  \rule{\linewidth}{2pt}
  obrázek
  \vspace{0.5\baselineskip}
  \hrule
  \caption{Ukázka obrázku s~vodorovnými linkami}
  \label{obr3}
  \rule{\linewidth}{2pt}
\end{figure}
```

Za určitých okolností může být lepší variantou obrázek podbarvit. K tomu použijeme prostředí `snugshade` z balíčku framed.

```
\begin{figure}
  \setlength{\fboxsep}{0mm}
  \begin{snugshade}
    \vspace{0.5\baselineskip}
    \centering
    obrázek
    \caption{Ukázka podbarveného obrázku včetně popisku}
    \label{obr4}
```

obrázek

Obrázek 3: Ukázka obrázku s vodorovnými linkami

obrázek

Obrázek 4: Ukázka podbarveného obrázku včetně popisku

```
    \vspace{0.5\baselineskip}
  \end{snugshade}
\end{figure}
```

Ne všechny objekty je možné zapouzdřit do některého prostředí z balíčku framed. Konkrétně při použití plovoucích objektů, poznámek pod čarou, poznámek v okraji nebo značek `\mark` uvnitř některého z těchto prostředí se tyto objekty ztratí. Dále balíček framed také nespolupracuje s balíčkem multicol nebo s jinými makry používanými pro sloupcovou sazbu. V některé budoucí verzi balíčku možná budou tyto problémy vyřešeny.

Pomocí balíčku framed si můžeme vytvořit vlastní prostředí, v němž může nastat stránkový zlom. K tomu využijeme prostředí[4] `MakeFramed` a nadefinujeme makro `\FrameCommand`, které příslušný „rámeček“ vykreslí. Prostředí `MakeFramed` má jeden argument, v němž je možné nastavit šířku sazby `\hsize` a definovat, jakým způsobem se za vysázeným rámečkem případná nová nastavení vrátí zpět. Balíček pro tyto účely definuje makro `\FrameRestore`, které je jednodušší variantou vnitřního LaTeXového makra `\@parboxrestore`. V délkovém registru `\width` je uložena šířka celého rámečku. Vše si ukážeme později. Ale pojďme od začátku.

Rámeček prostředí `framed` je definován pomocí makra `\fbox` následovně.

```
\providecommand{\FrameCommand}{%
  \setlength{\fboxrule}{\FrameRule}%
  \setlength{\fboxsep}{\FrameSep}%
  \fbox}
```

Délkové registry `\FrameRule` a `\FrameSep` jsou zavedeny v balíčku a určují tloušťku čáry a mezeru na okraji rámečku. Samotné prostředí `framed` pak využívá makro `\MakeFramed`.

[4] Namísto použití maker `\begin{MakeFramed}` a `\end{MakeFramed}` můžeme přímo použít makra `\MakeFramed` a `\endMakeFramed`, což ostatně sám autor dále v textu dělá. (pozn. překl.)

```
\newenvironment{framed}{%
    \MakeFramed {\advance\hsize-\width
      \FrameRestore}}%
  {\endMakeFramed}
```

Zbývající tři prostředí jsou definována podobně. Jako rámovací makro používají `\colorbox` nebo `\vrule`.

```
\newenvironment{shaded}{%
  \def\FrameCommand{\fboxsep=\FrameSep
    \colorbox{shadecolor}}%
  \MakeFramed {\FrameRestore}}%
  {\endMakeFramed}
```

```
\newenvironment{snugshade}{%
    \def\FrameCommand{%
      \colorbox{shadecolor}}%
    \MakeFramed {\FrameRestore\@setminipage}}%
  {\par\unskip\endMakeFramed}
```

```
\newenvironment{leftbar}{%
    \def\FrameCommand{\vrule width 3pt
      \hspace{10pt}}%
    \MakeFramed {\advance\hsize-\width \FrameRestore}}%
  {\endMakeFramed}
```

Povšimněte si, že uvnitř prostředí `framed` a `leftbar` je sazba zúžená. Naopak u prostředí `shaded` a `snugshade` se šířka sazby nemění.

Tyger, tyger, burning bright
In the forests of the night,
What immortal hand or eye
Could frame thy fearful symmetry?

Songs of Experience
William Blake

2. Definice vlastních rámečků

V některých případech je celkem snadné nadefinovat si své vlastní rámovací prostředí založené na balíčku framed, avšak většinou to chce trochu experimentování.

Prostředi `narrowframe` zúží sazbu textu na danou šířku, text zarámuje a rámeček vycentruje.

```
\newenvironment{narrowframe}[1][0.8\hsize]{%
    \MakeFramed{\setlength{\hsize}{#1}
      \FrameRestore}}%
  {\endMakeFramed}
```

Prostředí můžeme použít buď přímo

```
\begin{narrowframe} ... \end{narrowframe}
```

nebo s jedním nepovinným parametrem udávajícím šířku sazby, například

```
\begin{narrowframe}[10cm] ... \end{narrowframe}
```

V této ukázce je použito implicitní nastavení, kdy je šířka sazby uvnitř rámečku rovna 80 % šířky sazby v okolí rámečku.

Můžeme vyzkoušet také hladší rámečky. Prostředí `roundedframe` vytvoří zaoblený rámeček. K tomu potřebuje balíček fancybox[5] a jeho makro `\ovalbox`. V definici prostředí samozřejmě můžeme parametry nastavit dle libosti, případně použít jiná makra z tohoto balíčku k vytvoření jiných rámečků.

```
\newenvironment{roundedframe}{%
    \def\FrameCommand{%
      \cornersize*{20pt}%
      \setlength{\fboxsep}{5pt}%
      \ovalbox}%
    \MakeFramed{\advance\hsize-\width
      \FrameRestore}}
  {\endMakeFramed}
```

Čas od času se na `comp.text.tex` vyskytne požadavek na vytvoření graficky odděleného číslovaného prostředí. Robert Nyqvist [3] v jedné z odpovědí poskytl následující kód založený na balíčku framed.

[5]Jak se ukázalo při sazbě tohoto článku, není balíček fancybox kompatibilní s balíčkem fancyvrb. Oba balíčky definují prostředí `Verbatim`, každý však jinak. (pozn. red.)

```
\makeatletter
\definecolor{rulecolor}{gray}{0.65}
\newcounter{ruledexample}
\newlength{\releftgap}
\setlength{\releftgap}{4pt}
\newlength{\rerightgap}
\setlength{\rerightgap}{1em}
\newlength{\rerule}
\setlength{\rerule}{1.25pt}
\newlength{\Eheight}
\newenvironment{ruledexample}[1][Příklad]{%
    \settoheight{\Eheight}{\textbf{#1}}%
    \addtolength{\Eheight}{-\rerule}%
    \def\FrameCommand{\hspace{-\releftgap}%
      {\color{rulecolor}%
        \vrule width \rerule}%
      \hspace{\releftgap}\hspace{-\rerule}}%
    \MakeFramed{\advance\hsize-\width}%
    \refstepcounter{ruledexample}%
    \makebox[0pt][l]{%
      \hspace{-\parindent}%
      \hspace{\rerightgap}%
      {\color{rulecolor}\rule{1.5em}{\rerule}}%
      \quad
      \raisebox{-0.5\Eheight}[0pt]{%
        \textbf{#1\ \theruledexample}}%
      }\\[.5\baselineskip]%
    \noindent\ignorespaces}%
  {\@afterheading\\
    \makebox[0pt][l]{%
      \hspace{-\releftgap}%
      {{\color{rulecolor}
        \rule{\columnwidth}{\rerule}%
        \rule{\releftgap}{\rerule}}%
      }}
    \endMakeFramed}
\makeatother
```

Příklad 1

Tento příklad ukazuje použití prostředí `ruledexample`. Prostředí má svůj titulek, číslo a můžeme se na něj odkazovat přes `\label`. Pokud je to třeba, provede se uvnitř prostředí stránkový zlom. Zdrojový kód je založený na makrech Roberta

Nyqvista z roku 2003, přičemž jsem k nim přidal následující parametry, jimiž je možno upravit vzhled prostředí.

- `rulecolor` – barva linek
- `\rerule` – tloušťka linek
- `\releftgap` – vzdálenost, o jakou je svislá čára posunuta do okraje
- `\rerightgap` – odsazení linky v titulku
- nepovinný parametr – titulek prostředí, implicitně je „Příklad“

Jak můžete vidět na tomto příkladu, je možné uvnitř prostředí `ruledexample` používat výčtová prostředí jako `itemize`.

Někteří uživatelé si všimli, že se prostředí `framed` uvnitř jiného prostředí, jako například uvnitř `quote` nebo uvnitř `adjustwidth` z balíčku changepage [4], chová jinak, než by očekávali. Níže situaci popíšu blíže.

Zahájíme prostředí `quotation`. V něm můžeme použít prostředí `framed`, stejně jako libovolné další prostředí založené na prostředí `list`.

Nyní jsme uvnitř prostředí `framed`. Všimněte si, že rámeček je napříč celou původní šířkou textu. To není vždy žádoucí.

S Donaldem Arseneauem jsme diskutovali, jak problém vyřešit a udělat rámeček užší. Oba jsme vytvořili nějaký zdrojový kód a jako obvykle ten Donaldův byl lepší.

Nyní jsme uvnitř prostředí `qframe` vytvořeném Donaldem Arseneauem. Definici tohoto prostředí jsem poté vložil do mé třídy memoir [5]. Do ní jsem vložil také stínovanou variantu tohoto prostředí.

Jestliže použijete prostředí `adjustwidth` z balíčku changepage[6] [4] nebo z třídy memoir, pak uvnitř něho prostředí `qframe` funguje lépe než prostředí `framed`.

Zde prostředí `quotation` ukončíme.

Definice prostředí `qframe` je následující.

```
\makeatletter
\newenvironment{qframe}{%
    \def\FrameCommand##1{%
      \setlength{\fboxrule}{\FrameRule}%
      \setlength{\fboxsep}{\FrameSep}%
      \hskip\@totalleftmargin\fbox{##1}%
```

[6] Balíček changepage je nástupce balíčku chngpage, který se již nepoužívá.

```
120       \hskip-\linewidth
121       \hskip-\@totalleftmargin
122       \hskip\columnwidth}%
123     \MakeFramed{\advance\hsize-\width
124       \advance\hsize \FrameSep
125       \@totalleftmargin\z@
126       \linewidth=\hsize}}%
127   {\endMakeFramed}
128 \makeatother
```

Dalším užitečným druhem rámečku je rámeček s popiskem, který se na následující stránce zopakuje. Kód prostředí je výrazně komplikovanější než dříve a musel jsem celkem hodně experimentovat, aby vše fungovalo. Nejdříve si ukážeme příklad.

Rámeček s popiskem

Prostředí `framewithtitle` vytvoří rámeček a na jeho začátek vloží popisek. Jestliže dojde ke stránkovému zlomu, bude za ním následovat rámeček s pokračovacím popiskem. Definice prostředí následuje níže.

Prostředí můžeme použít se dvěma parametry udávajícími pokračovací popisek a úvodní popisek

```
129 \begin{framewithtitle}[pokračovací]{úvodní}
130 ...
131 \end{framewithtitle}
```

První parametr je nepovinný a když jej vynecháme, bude pokračovací popisek stejný jako úvodní, ale následovaný textem „(pokr.)“

```
132 \begin{framewithtitle}{popisek}
133 ...
134 \end{framewithtitle}
```

Umístění popisku je definováno v makru `\frametitle`, které podle definice na řádku 138 vysází popisek doleva. Pokud bychom chtěli popisek vycentrovat, předefinujeme makro následovně.

```
135 \renewcommand*{\frametitle}[1]{%
136   \centerline{\strut#1}}
```

I uvnitř tohoto rámečku je možné sázet zdrojový kód.

Zdrojový kód prostředí `framewithtitle` je založený na námětech Donalda Arseneaua, které měl ke starší verzi balíčku framed. Velká část kódu se zabývá popiskem, aby na začátku rámečku byl úvodní popisek a po stránkovém zlomu následoval pokračovací popisek. Některé části kódu byly uloženy do maker, aby

bylo možné je použít i v dalších situacích. Nejsložitějším problémem je, že makra neustále přenastavují popisek, aby určila místo případného stránkového zlomu, a až pak příslušný popisek vysázejí. Toto je ošetřeno testem `\ifcontframe`, na jehož základě se nastaví konkrétní popisek. Celé toto testování se provádí uvnitř makra `\Fr@meSetup`. Makro `\FrameTitle` pak popisek vysází.

```
\makeatletter
\newcommand*{\frametitle}[1]{\strut#1}
\newif\ifcontframe
\newcommand*{\Fr@meSetup}[2]{%
  \fboxrule=\FrameRule \fboxsep=\FrameSep
  \global\contframefalse
  \def\Fr@meFirst{\textbf{#2}}%
  \def\Fr@meCont{\textbf{#1}}%
  \def\FrameCommand##1{%
    \Title@Fr@me{\Fr@meCurrent}{##1}%
    \global\let\Fr@meCurrent\Fr@meNext
    \ifcontframe
      \global\let\Fr@meNext\Fr@meCont
    \fi
    \global\contframetrue}%
  \global\let\Fr@meCurrent\Fr@meFirst
  \global\let\Fr@meNext\Fr@meFirst}
\newcommand*{\FrameTitle}[1]{%
  \nobreak \vskip -0.7\FrameSep
  \rlap{\frametitle{#1}}%
  \nobreak\nointerlineskip
  \vskip 0.7\FrameSep}
\newenvironment{framewithtitle}[2][\Fr@meFirst\ (pokr.)]{%
    \def\Title@Fr@me##1##2{%
      \fbox{\vbox{\FrameTitle{##1}%
       \hbox{##2}}}}%
    \Fr@meSetup{#1}{#2}%
    \MakeFramed{%
      \advance\hsize-\width
      \FrameRestore}}%
  {\global\contframefalse
    \endMakeFramed}
\makeatother
```

Jako alternativu můžeme použít prostředí `titledframe`, v němž je popisek vysázen nad rámečkem. Parametry tohoto prostředí a horizontální pozice popisku se nastavují stejně jako u prostředí `framewithtitle`.

Popisek nad rámečkem

Prostředí `titledframe` vytvoří rámeček a nad něj vloží popisek. Jestliže dojde ke stránkovému zlomu, bude za ním následovat rámeček s pokračovacím popiskem. Definice prostředí následuje níže.

```
\makeatletter
\newenvironment{titledframe}[2][\Fr@meFirst\ (pokr.)]{%
    \def\Title@Fr@me##1##2{%
      \vbox{\FrameTitle{##1}%
        \noindent\fbox{##2}}}
    \Fr@meSetup{#1}{#2}%
    \MakeFramed{%
      \advance\hsize-\width
      \advance\hsize -2\FrameRule
      \advance\hsize -2\FrameSep
      \FrameRestore}}%
  {\global\contframefalse
    \endMakeFramed}
\makeatother
```

Musím uznat, že jsem detaily zdrojových kódů příliš nepopisoval. Jedním z důvodů je, že by se tímto článek velmi prodloužil. Ale hlavním důvodem je, že příliš nerozumím, jak balíček framed provádí všechna ta svá kouzla. Proto jsem také hodně experimentoval (mnoho pokusů a ještě více omylů), než všechno začalo pořádně fungovat.

Odkazy

1. WILSON, Peter. Glisterings. *TUGboat* [online]. 2011, roč. 32, č. 1, s. 99–103 [vid. 2011-05]. Dostupné z: `https://tug.org/TUGboat/tb32-1/tb100glister.pdf`.
2. ARSENEAU, Donald. *framed: Framed or shaded regions that can break across pages* [online]. CTAN, 2007 [vid. 2011-05]. Dostupné z: `https://ctan.org/pkg/framed`. Verze 0.95.
3. NYQVIST, Robert. *'example' environment or command.* `comp.text.tex` newsgroup, 2003-01-11.
4. WILSON, Peter. *changepage: Margin adjustment and detection of odd/even pages* [online]. CTAN, 2009-10 [vid. 2011-05]. Dostupné z: `https://ctan.org/pkg/changepage`.
5. WILSON, Peter. *memoir: Typeset fiction, non-fiction and mathematical books* [online]. CTAN, 2008 [vid. 2011-05]. Dostupné z: `https://ctan.org/pkg/memoir`.

Summary: It Might Work XII

This paper shows several ways in LaTeX how to create a frame around a text with a possible page break inside the frame. Some solutions using the framed package are given. Several possibilities to create new types of frames based on that package are also presented.

Keywords: Frames, framed package, LaTeX

Peter Wilson, herries.press@earthlink.net
12 Sovereign Close
Kenilworth, CV8 1SQ, UK

Zpravodaj Československého sdružení uživatelů TEXu
ISSN 1211-6661 (tištěná verze), ISSN 1213-8185 (online verze)

Vydalo:	Československé sdružení uživatelů TEXu vlastním nákladem jako interní publikaci
Obálka:	Antonín Strejc
Ilustrace na obálce:	Petr Olšák
Počet výtisků:	260
Uzávěrka:	31. 10. 2022
Odpovědný redaktor:	Jan Šustek
Redakční rada:	Pavel Haluza, Lukáš Novotný, Vít Novotný, Michal Růžička a Jan Šustek (šéfredaktor)
Vědecká rada:	Ján Buša (předseda), Jiří Demel, Jaromír Kuben (zástupce předsedy), Jiří Rybička a Petr Sojka
Technická redakce:	Vít Novotný
Evidenční číslo MK:	E 7629
Adresa:	CSTUG, Nejedlého 373/1, 638 00 Brno
Email:	cstug@cstug.cz

Zřízené poštovní aliasy sdružení CSTUG:

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

`board@cstug.cz`
korespondence členům výboru

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

`gacstug@cstug.cz`
grantová agentura CSTUGu

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

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

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

`bookorders@cstug.cz`
objednávky tištěné TEXové literatury na dobírku

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

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

CONTENTS

Petr Sojka: Introductory Word 1

Petr Olšák: `cropmarks.tex` – Macros for Creating Crop Marks 4

Tereza Vrabcová: Digital Archival of the CSTUG Bulletin 11

Denis Roegel: Kissing Circles: A French Romance in METAPOST 18

Vít Novotný: High-Level Languages for TEX 35

Max Chernoff: Automatically Removing Widows and Orphans with lua-widow-control 49

Peter Wilson: It Might Work XII 77