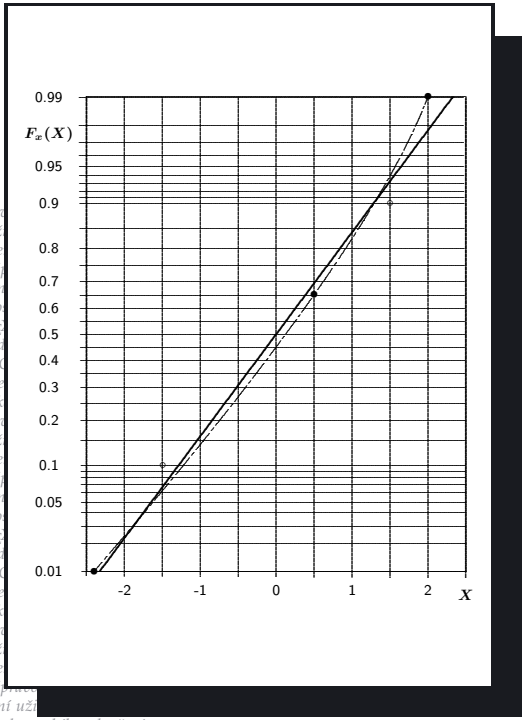
CSTUG



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 C
ení uživatelů TEXu Zpravodaj Československého sdružení uživate
skolovenského 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 C
ení uživatelů TEXu Zpravodaj Československého sdružení uživate
skolovenského 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

ení uživatelů TEXu Zpravodaj Československého sdružení uživatelů TEXu Zpravodaj Če
skoslovenského sdružení uživatelů TEXu Zpravodaj Československého sdružení uživatelů

Československého sdružení uživatelů TEXu

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

2

2004

ISSN 1211-6661 Tištěná verze

ISSN 1213-8185 Online verze

Ročník 14

OBSAH

Zpravodaj Československého sdružení uživatelů TEXu je vydáván v tištěné podobě a distribuován zdarma členům sdružení. Po uplynutí dvanácti měsíců od tištěného vydání je poskytován v elektronické podobě (PDF) ve veřejně přístupném archívu dostupném přes `http://www.cstug.cz/` .

Své příspěvky do Zpravodaje můžete zasílat v elektronické podobě, nejlépe jako jeden archivní soubor (`.zip, .arj, .tar.gz`). Postupujte podle instrukcí, které najdete na stránce `http://bulletin.cstug.cz/` . Pokud nemáte přístup na Internet, můžete zaslat příspěvek na disketě na adresu:

Zdeněk Wagner
Vinohradská 114
130 00 Praha 3

Disketu formátujte nejlépe pro DOS, formáty Macintosh 1.44 MB a EXT2 jsou též přijatelné. Nezapomeňte přiložit všechny soubory, které dokument načítá (s výjimkou standardních součástí CSTEXu), zejména v případě, kdy vás nelze kontaktovat e-mailem.

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

Úvodníček

Petr Olšák

Na nadcházejícím podzimním valném shromáždění proběhnou volby do nového výboru našeho sdružení. Toto shromáždění se uskuteční **v sobotu 27. listopadu 2004 v Praze na Matematicko-fyzikální fakultě**, Sokolovská 83. V rámci tohoto setkání TEXistů vystoupí pan Zdeněk Wagner s přednáškou na téma použití TEXu v praxi. Čas a program upřesníme v pozvánce, kterou pošleme e-mailem případně klasickou poštou všem našim členům.

Chtěl bych při této příležitosti požádat všechny členy našeho sdružení, aby zvážili možnost kandidovat do výboru. Zájemci mohou psát na adresu současného výboru `cstugb-l@muni.cz`. Podrobnosti o pravidlech a průběhu voleb pošleme rovněž všem členům během týdne.

Byl bych velmi potěšen, kdyby výbor našeho sdružení prodělal výraznou omlazovací kúru a došlo k podstatné změně zastoupení. Ne snad proto, že by se mi se současným výborem explicitně špatně spolupracovalo, ale změna ve výboru a zájem členů o dění ve sdružení je důkazem životaschopnosti této organizace. Bohužel, už několikáté volební období se ukazuje, že to s tou životaschopností není zas tak valné a ve výboru máme pořád stejné tváře. Navíc prací pro sdružení už hodně „unavené“. Ani já jako současný předseda určitě nechci zaclánět novému výboru v jeho nových projektech a záměrech a s radostí přenechám tuto funkci někomu jinému, kdo třeba i radikálně změní fungování naší organizace. Pokud ovšem bude k takové změně v novém výboru vůle.

Dovolím si nyní zde v úvodníku ventilovat v souvislosti s volbami své poněkud kontroverznější soukromé názory. Velmi by mě mrzelo, kdyby nový výbor sestával ze stejných lidí, a ti by stejně jako před třemi lety naléhali na to, abych se stal zase předsedou. Pokud by kandidátka na nový výbor vypadala tak, že mezi členy sdružení skutečně není zájem o změnu, vážně uvažuji do výboru nekandidovat. Kdo si vzpomene na situaci před třemi lety, ten ví, že jsem ve svém příspěvku kandidáta předložil útlumový program, protože se mi zdálo, že mezi členy sdružení není vůle aktivně se podílet na činnosti sdružení a současný stav je provozován velmi neefektivně. Já sám taky asi nic nového nedokážu. Výbor ale tehdy rozhodl jinak a útlumový program mi zakázal. Stal jsem se tak předsedou, který měl zakázáno prosazovat svůj volební program. Zvolil jsem si tedy krédo předsedy *nepřekážet*: má-li někdo zájem cokoli v naší organizaci dělat, ať to dělá. Dodnes jsem se ovšem mnoha aktivit od členů sdružení nedočkal. Přesto může někdo doufat, že zákaz útlumového programu bylo od výboru prozíravé rozhodnutí: skrytá a zatím dřímající aktivita našich členů se konečně po třech letech projeví například intenzivním zájmem o kandidování do nového výboru

s představou, jak naši organizaci udělat nově a lépe. Prohlašuji, že pokud se tento zájem členů o práci ve výboru neprojeví, pak rozhodně nechci prodlužovat agónii našeho sdružení a být na další tři roky předsedou – udržovatelem současného zakonzervovaného a z mého pohledu velmi žalostého stavu.

Dost bylo řečí o volbách, přejdu na jiné téma. V tomto časopise najdete sice jen dva články, zato jeden z nich je značně rozsáhlý a rozhodně si zaslouží pozornost. Článek Miroslava Baldy o výpočtech a diagramech v TEXu mě samotného překvapil. Autor zde popisuje vlastní netriviální makra na tvorbu grafů a diagramů přímo pomocí TEXových příkazů. K dispozici jsou navíc velmi silná makra na numerické výpočty v TEXu. Zdá se až neuvěřitelné, co se dá na úrovni maker udělat. Práce je volně k dispozici i jako „elektronická příloha“ časopisu. Věřím, že i článek Vítka Zýky o rozšířeních v pdfTEXu čtenáře zaujme. Mnoho z nás jistě slyšelo, co vše se dá s pdfTEXem dělat, někteří možná i přečetli Thànhovu disertaci. Mít ale stručné shrnutí v českém jazyce od zkušeného autora, který rozšíření pdfTEXu aktivně používá, je určitě užitečné.

O prázdninách mě oslovil pan David Jež s tím, že má přes léto více času a že je ochoten něco udělat pro CSTEX. Požádal jsem ho, ať doplní znaky paragrafu a Eura do CSfontů jednak v METAFONTových a jednak v Type1 variantách. Dále se nabíl udělat novou implemetaci počeštění základních PostScriptových fontů (Avantgarde, Times atd.) postavenou na volně dostupných fontech od URW, které jsou dnes v každé distribuci Ghostscriptu. Na rozdíl od stávající implementace z balíčku `cspsfonts` se v tomto případě pracuje s tím, že všechny znaky s háčky a čárkami potřebné pro český a slovenský jazyk v URW fontech skutečně existují, takže není nutno se uchylovat k virtuálním fontům. Má to tu výhodu, že výsledné PDF obsahuje český text, který se dá „nabrat do myši“ případně v něm vyhledávat. To při použití virtuálních fontů s plovoucími akcenty z balíčku `cspsfonts` nebylo možné.

David Jež se obou úkolů zhostil velmi pečlivě a výsledky mi předvedl na konci prázdnin. Nechal jsem ho, ať práci zveřejní na konferenci `cstex` (to se už stalo) a po případných připomínkách uživatelů a jejich zapracování zařadím nové CSfonty i podporu základních PostScriptových fontů do oficiálního CSTEXu (to se teprve stane). Přiznám se, že jsem si připadal jak Alenka v říši divů. Najednou se na Síti objeví někdo (ani nevím jak vypadá), kdo se nabídne pro CSTEX něco udělat, zeptá se, do čeho by se dalo nejlíp píchnout, udělá netriviální práci, která mu jistě zabrala řádově desítky hodin, výsledek funguje a je zveřejněn. Skutečně jsem si připadal jako v pohádce.

Věřím, že pokud existují takoví lidé, jako autoři článků v tomto čísle nebo jako pan David Jež, je reálná šance, že CSTUG vykročí do dalšího roku s novým kvalitním výborem a s novými vyhlídkami na budoucnost.

17. 10. 2004

Používáme pdfTEX IV: mikrotypografické rozšíření

Vít Zýka

Tento článek se věnuje dvěma mikrotypografickým rozšířením odstavcového zlomu TEXu, které Hàn Thế Thành implementoval do pdfTEXu jako svou doktorskou práci [5]. Prvním rozšířením je prostrkání okrajů. *Jde o zobecnění visící interpunkce na libovolný znak a na obě hrany bloku textu. Cílem je korigovat optické výchylky okrajů textu dané různou světlostí kresby jednotlivých znaků. Druhým rozšířením je hz-*algoritmus. *Jde o možnost horizontálního zúžení nebo rozšíření písma o několik procent; tím se umožní vyšší variabilita algoritmu řádkového zlomu a vyšší pravděpodobnost akceptovatelného vzhledu odstavce. Změna písma musí být taková, aby byla okem nepostřehnutelná. Oba algoritmy navrh věhlasný německý typograf Hermann Zapf [6]. Zaměříme se zde pouze na technickou část použití obou algoritmů; typografickou analýzu a volbu vhodných hodnot parametrů může čtenář najít v publikacích [5, 4, 3, 2].*

Předcházející odstavec je vysázen standardním LaTeXovým nastavením parametrů odstavcového zlomu, především `\tolerance`=200, `\emergencystretch`=0pt. Použijeme-li *hz*-algoritmus s fonty v rozmezí ±3 % a algoritmus prostrkání levého okraje pro znaky *T* a pravého okraje pro tečku, čárku a rozdělovací znaménko, dostaneme náledující výsledek:

Tento článek se věnuje dvěma mikrotypografickým rozšířením odstavcového zlomu TEXu, které Hàn Thế Thành implementoval do pdfTEXu jako svou doktorskou práci [5]. Prvním rozšířením je prostrkání okrajů. *Jde o zobecnění visící interpunkce na libovolný znak a na obě hrany bloku textu. Cílem je korigovat optické výchylky okrajů textu dané různou světlostí kresby jednotlivých znaků. Druhým rozšířením je hz-*algoritmus. *Jde o možnost horizontálního zúžení nebo rozšíření písma o několik procent; tím se umožní vyšší variabilita algoritmu řádkového zlomu a vyšší pravděpodobnost akceptovatelného vzhledu odstavce. Změna písma musí být taková, aby byla okem nepostřehnutelná. Oba algoritmy navrh věhlasný německý typograf Hermann Zapf [6]. Zaměříme se zde pouze na technickou část použití obou algoritmů; typografickou analýzu a volbu vhodných hodnot parametrů může čtenář najít v publikacích [5, 4, 3, 2].*

Prostrkání okrajů (margin kerning)

Začneme algoritmem, jehož použití je jednodušší. Pro zapnutí prostrkání okrajů[1] stačí přiřadit kladnou hodnotu parametru `\pdfprotrudechars` a nastavit každému znaku, který má být prostrčen, velikost přesahu. Parametr má tři polohy:

$\left\{\begin{array}{ll} \leq 0 & \text{Algoritmus je neaktivní, realizuje se standardní zlom TEXu (implicitní hodnota).} \\ 1 & \text{Řádky jsou nalámány standardním algoritmem a pak je aplikován algoritmus prostrkání konce řádků na každý řádek.} \\ \geq 2 & \text{Přesah každého znaku vstupuje do celkového optimalizačního algoritmu zlomu odstavce, takže může dojít i k jinému řádkovému zlomu.} \end{array}\right.$

Registr pracuje jako odstavcové parametry, tj. význam má pouze jeho hodnota při ukončení odstavce. Z toho plyne, že algoritmus prostrkání okrajů nemůžeme aplikovat jen na část odstavce, ale vždy na celý najednou.

Prostrkáním je myšlena míra, o kolik daný znak přesahuje z levého, respektive pravého, okraje odstavce. Levý přesah se nastavuje pomocí

`\lpcode`⟨`font`⟩⟨`8-bit number`⟩⟨`equals`⟩⟨`hodnota`⟩

a pravý pomocí

`\rpcode`⟨`font`⟩⟨`8-bit number`⟩⟨`equals`⟩⟨`hodnota`⟩

kde ⟨`8-bit number`⟩ je ASCII hodnota znaku a ⟨`hodnota`⟩ přesahu je vztažena k velikosti písma, tj. ⟨`hodnota`⟩=1000 značí přesah o 1 em, ⟨`hodnota`⟩=200 určí přesah o ⅕ em. Přesah může nabývat i záporných hodnot.

Příklad: (hodnoty použité při druhém vysázení abstraktu)

```
\def\setupprot{\pdfprotrudechars=2
  \lpcode\font‘\T=80
  \rpcode\font‘\.=70 \rpcode\font‘\,=70
  \rpcode\font\hyphenchar\font=100 }
\itshape \setupprot Tento článek se věnuje ...
```

Hz-algoritmus (font expansion)

Typografické pravidlo říká, že odstavec působí esteticky a nenamáhá při čtení, pokud je v něm tiskařká barva rozprostřena rovnoměrně, takže při pohledu z dálky tvoří jednotnou šedou plochu. Tuto jednotnost mohou narušovat příliš stažené

[1] V příspěvku na SLT 2002 v Seči [7, 8] jsem tento algoritmus nazval *visící znaky* jako zobecnění termínu visící interpunkce. Pojmenování *prostrkání okrajů* však lépe vystihuje důvody zavedení tohoto algoritmu.

nebo příliš roztažené mezislovní mezery, zvláště při sazbě do úzkého bloku (do 40 znaků na řádce).

Použití *hz*-algoritmu[2] je trošku komplikovanější a vyžaduje kromě nastavení na úrovni makrojazyka pdfTEXu i přípravu expandovaných fontů[3].

Úroveň makrojazyka

Algoritmus se zapíná registrem `\pdfadjustspacing`. I on má tři polohy:

- ≤ 0 — Algoritmus je neaktivní, realizuje se standardní zlom TEXu (implicitní hodnota).
- 1 — Řádky jsou nalámány standardním algoritmem a pak při potřebě řádek stáhnout nebo natáhnout pruží vedle mezislovních mezer i jednotlivé znaky (na rozdíl od mezer však jen v diskrétních krocích).
- ≥ 2 — Možnost expanze každého znaku vstupuje do celkového optimalizačního algoritmu zlomu odstavce a je tak ovlivněn i řádkový zlom TEXu.

Registr je odstavcovým parametrem a jeho hodnota je brána v potaz jen při provedení povelu `\par`.

Druhým primitivem pro nastavení *hz*-algoritmu je

`\pdffontexpand`⟨`font`⟩⟨`stretch`⟩⟨`shrink`⟩⟨`step`⟩⟨`scalefactor`⟩

který fontu ⟨`font`⟩ přidělí maximální hodnoty roztažení ⟨`stretch`⟩ a stažení ⟨`shrink`⟩. Dále specifikuje diskrétní krok ⟨`step`⟩, s jakým má expanzi generovat. Tyto tři parametry se zadávají v promilích velikosti písma, tj. v tisícinách jednotky em.

Zatímco parametry ⟨`stretch`⟩, ⟨`shrink`⟩ a ⟨`step`⟩ určují, jaké metriky může pdfTEX při zlomu použít (tím se ovlivní, kolik horizontálního místa budou znaky zabírat), poslední parametr ⟨`scalefactor`⟩ na výpočet zlomu nemá vliv. Jde o koeficient, kterým se mění jen šířka kresby znaku. Hodnota 0 znamená, že znaky fontu budou vykresleny v původní neexpandované šířce a budou prostrkány tak, aby zabíraly zvolenou expandovanou šířku. Hodnota 1000 nastaví šířku kresby na expandovanou velikost. Hodnoty mimo interval 0 až 1000 nemají význam. Pro většinu případů je vhodné použít hodnotu 1000. Je-li výstup do DVI (`\pdfoutput=1`), není možné použít jinou hodnotu než ⟨`spacefactor`⟩=1000.

[2] V příspěvku na SLT 2002 v Seči jsem tento algoritmus nazval *horizontální zvětšování/zmenšování znaků*. Vycházel jsem z překladu termínu *font expansion*. Později jsem objevil termín *hz*-algoritmus. Protože je je tento název odvozen podle jeho tvůrce a je používán i mimo komunitu pdfTEXu [1], rozhodl jsem se tento název respektovat.

[3] Slovo *expandovaný* budeme dále používat jak ve významu horizontálního rozšíření, tak i stažení (expanze se zápornou hodnotou).

Pokud chceme nějaký znak horizontálně expandovat méně než bude vybraná expanze celého fontu, použijeme primitivu

`\efcode⟨font⟩⟨8-bit number⟩⟨equals⟩⟨hodnota⟩`

analogicky jako `\lpcode`. Znaku s ASCII kódem ⟨`8-it number`⟩ lze přiřadit relativní hodnotu v rozmezí 0 (znak expandovat nebude) až 1000 (bude expandovat jako celý font, implicitní hodnota).

Příklad:

```
\pdfadjustspacing=2
\pdffontexpand\tenrm 30 10 10 1000
```

Zapli jsme *hz*-algoritmus s globální optimalizací pro font `\tenrm`, s možností rozšíření fontu o 3 %, stažení o 1 % s krokem 1 %. Kromě základní metriky budeme tedy ještě potřebovat metriku `-10`, `+10`, `+20` a `+30`. Kresba se bude expandovat stejně jako vypočtená šířka, tj. nedojde *k mikroprostrkání*.

```
\pdffontexpand\font 15 0 5 0
```

Zde jsme nepovolili aktuálně nastavené písmo stahovat, ale pro roztažení jsme zvolili jemnější krok 0,5 %. Kresbu písma nechceme expandovat (znaky budou zprava prostrkány).

```
\pdffontexpand\font 15 0 5 500
\efcode\font‘\W=800
```

Kromě toho, že znaky budou expandovány jen na polovinu potřebného expandované místa (zbytek bude doplněn prostrkáním), jsme ještě u znaku *W* omezili expanzi o 20 %.

Příprava rozšířených metrik a bitmapových fontů

Má-li pdfTEX použít expandovaný font, potřebuje znát jeho metriku. Syntaxe názvu metriky je následující: `fontname-shrink.tfm` nebo `fontname+stretch.tfm`, takže například metrika $\mathcal{CS}$fontu Computer Modern Roman o velikosti 10 pt zúžená o 2 % musí být uložena v souboru s názvem `csr10-20.tfm`. Příprava metrik bude závislá na typu fontu. Pro některé z nich, zde uvádím návod:

METAFONT**ové fonty Computer Modern**

Knuth vytvořil Computer Modern pomocí parametrických souborů. Parametr `u#` určuje základní šířku znaků. Pro expandovaný font o -2 % stačí do parametrického souboru `cmr10-20.mf`, který vznikl přejmenováním `cmr10.mf`, připsat za řádek

```
u#:=20/36pt#;       % unit width
```

řádek

```
u#:=u#-20/1000u#;
```

Spustíme-li METAFONT na takto vytvořený soubor

```
mf \mode:=ljfour; mag:=1; input cmr10-20.mf
```

dostaneme jak metriku fontu `cmr10-20.tfm`, tak bitmapovou kresbu[4] `cmr10-20.600gf`. Komprimovanou bitmapu `cmr10-20.pk` získáme příkazem

```
gftopk cmr10-20.600gf
```

Soubory `.tfm` a `.pk` pak uložíme do náležitých adresářů (pro WEB2C instalaci `$TEX/texmf-local/fonts/ftm/` a `$TEX/texmf-local/fonts/pk/`) a přebudujeme databázi souborů (`mktexlsr`).

Pro české $\mathcal{CS}$fonty si zkopírujeme mírně upravený soubor `cscode.mf`, např. do adresáře `$TEX/texmf-local/fonts/source/public/cs/`. Pak už stačí jen připravit soubor s obsahem

```
input cscode
use_driver;
```

a názvem podle požadovaného fontu, např. `csr10-20.mf`. Dále postupujeme analogickým voláním

```
mf \mode:=ljfour; mag:=1; input csr10-20.mf
gftopk csr10-20.600gf
```

Pozor! Předpokladem je, že jsme předem připravili parametrický soubor originálního fontu `cmr10-20.mf`.

Type 1, bez překódování (např. anglické bez potřeby počeštění nebo Štormovy)

Pro PostScriptové Type 1 fonty nepotřebujeme generovat expandované kresby znaků (`.pfb`, `.pfa`), protože takový font si vytvoří pdfTEX automaticky pomocí transformační matice. Je pravda, že takovým způsobem se expanduje i šířka tahů a protože se tak mění světlost písma, není to teoreticky žádoucí postup. Protože však Type 1 fonty nemají parametrizaci oddělující šířku znaku a jeho tahů, nezbývá nám se s tímto nedostatkem smířit. Praxe ukazuje, že při malých hodnotách expanze fontu je změna tahů neznatelná[5].

Vysvětlili jsme si, že kresbami se zabývat nemusíme. Zbývá vytvořit metriky. Napsal jsem pro tento účel skript v Perlu `exppl.pl`, který využívá TEXových metrik (`.tfm`) neexpandovaného fontu. Expandovanou metriku připraví pronásobím horizontálních rozměrů `CHARWD`, `KRN`, `SPACE`, `STRETCH`, `SHRINK`, `QUAD`, `EXTRASPACE` uvnitř původní metriky. Nejdříve však musí metriku převést z komprimovaného tvaru do textové formy (Property List) pomocí programu `tftopl`. Po přepočítání rozměrů provede opačnou konverzi pomocí `pltotf` a soubor přejmenuje do požadovaného tvaru.

[4] Zde je použit `mode` pro laserovou tiskárnu LaserJet 4, 600 dpi, viz soubor `modes.mf`.

[5] Tento problém umožňují správně řešit Multiple Master fonty. Vzhledem k tomu, že Adobe ukončila jejich podporu a existuje velmi málo rodin písem v tomto formátu, návod na jejich použití zde neuvádím, ačkoliv pdfTEX jejich expandované použití umožňuje.

Příklad: Štormův font Preissig Antikva Roman rozšířený o 0,5 %, který se skládá ze dvou metrik:

```
fn=spar
exp=+5
tftopl ${fn}8z.tfm ${fn}8z.pl
tftopl ${fn}6s.tfm ${fn}6s.pl
exppl.pl ${fn}8z $exp
exppl.pl ${fn}6s $exp
pltotf ${fe}8z$exp.pl ${fe}8z$exp.tfm
pltotf ${fe}6s$exp.pl ${fe}6s$exp.tfm
```

Opět nezapomeňme vegenerované metriky umístit do vhodných adresářů a přegenerovat databázi. Expanzi fontu pak musíme zapnout pro obě metriky `8z` a `6r` třeba takto:

```
\input spreiss
\setfonts[PreissigAntikva/]

\newcount\N
\def\resetefcode#1{\N=0
  \loop\efcode#1\N=1000\advance\N by 1 \ifnum\N<256 \repeat}

\def\setupfont#1{\pdfadjustspacing=2
  \def\fotenc{6s}\setfonts[/]
  \resetefcode\font \pdffontexpand\font #1 #1 5 1000
  \def\fotenc{8z}\setfonts[/]
  \resetefcode\font \pdffontexpand\font #1 #1 5 1000 }

\setupfont{20} Text...
```

Používáme-li místo `L2` kódování `T1`, musíme metriku `8z` zaměnit za `8t`.

Virtuální fonty a překódování

Pokud potřebujeme generovat expandované metriky přímo z PostScriptových metrik (`.afm`), lze použít makrobalík `fontinst`. Skripty vytvořil Hàn Thế Thành. Předpokládají metriku v kódování `8a`, tj. musí se jmenovat *name*`8a.afm`. Pokud se font jmenuje jinak, je nutné vytvořit link s vhodným názvem.

Příklad: (Antykwa Torunska, italika, původní metrika `anttri.afm`)

```
ln -s anttri.afm anttri8a.afm
mktexlsr
mktextfm anttri8z+0
mktexlsr
pdftex.map
pdfcsplain file
```

Automatizované generování fontů

Co všechno k sazbě textu variabilními fonty potřebujeme?

- verzi pdfTEXu alespoň 0,14h z ledna 2001,
- rozšířený skript `mktextfm` a jeho další potomky `mktextfm.ext`, `mktfm8z` a `mktfmexstorm`,
- skript v Perlu `exppl.pl` pro Type 1 fonty,
- upravený soubor `cscode.mf` pro generování variabilních českých Computer Modern fontů a
- balík maker `fontinst` (je standardní součástí `tetexu`, `TEX Live` ap.) pro možnosti generování z `.afm` zdrojů.

S laskavým svolením původního autora většiny skriptů Hàn Thế Thành jsou k dispozici na adrese `ftp://cmp.felk.cvut.cz/pub/cmp/users/zyka/fe`. Poskytovány jsou bez jakýchkoliv záruk a garancí.

Reference

[1] Robert Bringhurst. *The Elements of Typographic Style.* Hartley & Marks, Point Roberts, WA, USA, version 2.4 edition, 2001.

[2] Hans Hagen. Does pdfTEX make things better?, 1999.

[3] Mirka Misáková. Písmo s variantní šířkou: nová naděje pro naše úzké sloupce. *Zpravodaj Československého sdružení uživatelů TEXu*, 8(2):65–81, 1998.

[4] Hàn Thế Thành. Bez názvu. `ftp://ftp.cstug.cz/pub/tex/local/cstug/thanh/hz/description.pdf`.

[5] Hàn Thế Thành. *Micro-typographic extensions to the TEX typesetting system.* PhD thesis, Masarykova univerzita v Brně, fakulta informatiky, 2000. Or: TUGboat 21,4, Dec 2000.

[6] Hermann Zapf. About micro-typography and the *hz*-program. *Electronic publishing*, 6(3):283–288, 1993. `http://cajun.cs.nott.ac.uk/compsci/epo/papers/volume6/issue3/zapf.pdf`.

[7] Vít Zýka. TEX a pdf. In Jan Kasprzak and Petr Sojka, editors, *SLT 2002 – sborník semináře o Linuxu a TEXu*, pages 69–77, Brno, Czech Republic, November 2002. Konvoj, CSTUG, CZLUG.

[8] Vít Zýka. TEX a pdf. *Zpravodaj Československého sdružení uživatelů TEXu*, 12(3–4), 2002. rozšřená verze [7].

Vít Zýka `zyka@cmp.felk.cvut.cz`

Výpočty a diagramy v LaTeXu

Miroslav Balda

Úvod

Tento článek se zabývá kreslením diagramů v LaTeXu. Proč? Vždyť na počítání a kreslení jsou určeny jiné programy, tak proč se na toto téma bavit v souvislosti se sázením dokumentů? Určitě by měli pravdu ti, jimž stačí základní arzenál příkazů TeXu nebo LaTeXu, aby vytvořili hezký dokument. Jisté potíže však vznikají těm z uživatelů, kteří mohou mít zvláštní požadavky na jeho úpravu, zejména pak, když chtějí text proložit obrázky a diagramy. Existují prostředky, jimiž lze obrázky vytvářet off-line, a potom v průběhu výstavby dokumentu je na vhodná místa vkládat. Tento postup je vhodný v případě, že obrázek nebo diagram je výsledkem složitých výpočtů. Vznikají však i situace, při nichž se vytváří dokument s mnoha jednoduchými obrázky, které vytvářeny externím prostředkem zabírají každý jeden soubor. Tyto soubory pak musí mít jedinečná jména a musí být evidovány a organizovány tak, aby byly snadno dostupné. I při jednoduché modifikaci obrázku je potom zapotřebí znovu použít pro úpravu externí prostředek.

S nevýhodami tohoto přístupu jsem se začal potýkat před deseti roky při psaní skript, do nichž jsem chtěl vložit celou řadu jednoduchých obrázků. Velice brzy jsem narazil na omezenost latexovské grafiky a zejména pak na výpočetní slabost sázecího programu. To bylo impulsem k hledání vhodných prostředků pro počítání a kreslení v LaTeXu. Od řady neúspěšných pokusů byl již jen krůček k výstavbě vlastních maker a z nich speciálních knihoven, jimiž bylo možno nejen kreslit, ale i rozvrhovat plochu obrázku, počítat průběhy jednodušších funkcí a vytvářet diagramy přímo z LaTeXu při tvorbě dokumentu. Vzniklá množina maker byla rozdělena do několika knihoven – stylů, jejichž výčet je uveden v tabulce 1. Soubor maker je popsán v publikaci uvedené v literatuře pod položkou [1]. Na tomto základě vznikly ještě další specializované knihovny. Navzdory plné funkčnosti maker nedoznaly tyto knihovny většího rozšíření jak pro velké nároky kladené na uživatele, tak i pro jejich malou publicitu.

`base.sty`	základní makra pro ostatní styly pro definice čítačů, délkových registrů, konverze argumentu na číslo, klávesnicových vstupů, kontrolních výstupů na obrazovku a operace se znaky,
`ariint.sty`	systém celočíselné aritmetiky
`arifix.sty`	aritmetika v pevné řádové čárce s rozsahem reálných čísel ± 214747 s pěti desetinnými místy
`funfix.sty`	funkce v aritmetice pevné řádové čárky: absolutní hodnota, sinus, kosinus, tangens, kotangens, exponenciála, přirozený a dekadický logaritmus, reálná mocnina čísla, odmocnina, difrakční funkce, error-funkce, hustota pravděpodobnosti a pravděpodobnost pro normální rozložení, pseudonáhodná čísla s rovnoměrným a normálním rozložením, korelované náhodné procesy a cykly typu „for“
`rplot.sty`	kreslení os diagramů (lineárních i logaritmických), vynášení funkcí s lineárními i hladkými spojnicemi bodů (s využitím `curves.sty` [2]), sloupcový i koláčový diagram, práce s externími daty a jejich položkami.

Tabulka 1: Rozložení maker do speciálních stylů v [1]

V době vzniku těchto maker byl již k dispozici Buchholzův balík `realcalc.sty` [3] pro aritmetiku v pevné řádové čárce s reálnými čísly o rozsahu ± 2147483647 s devíti desetinnými místy. Ten však ještě vykazoval chyby a neměl k dispozici nástroje pro doplňování palety funkcí, takže jeho použití bylo problematické. Kromě toho výpočetní rychlost tehdejších počítačů byla nízká a rovněž limitovala jeho použití.

Aritmetika v pevné řádové čárce – `fp.sty`

Situace se příliš nezměnila ani po pozdějším zveřejnění nové knihovny stylů pro aritmetiku v pevné řádové čárce – `fp.sty` – fix point package [4]. Ta ve všech směrech podstatně rozšířila možnosti svého vzoru, knihovny `realcalc.sty`.

Balík maker `fp.sty` je určen pro počítání s reálnými čísly a lze ho najít v různých distribucích LaTeXu (TeX Live, MiKTeX, ...). Jeho autor uvádí, že jej lze použít nejen v LaTeX2ε, ale i 2.09 a v TeXu. Popis maker je třeba hledat v podadresáři `\texmf\source\latex\fp\readme.fp`. Sám autor `fp.sty` v něm konstatuje, že dokumentace není dobrá. Uveďme zde proto popis použití `fp.sty` pro LaTeX2ε okomentovaný na základě zkušeností s jeho používáním.

Systém `fp` počítá s reálnými čísly z rozsahu

$$\pm\underbrace{999\cdots999}_{\text{až 18 míst}}\,.\,\underbrace{999\cdots999}_{\text{až 18 míst}}$$

Je tvořen 11 dílčími styly, o jejichž zavádění se uživatel nemusí starat, deklaruje-li v úvodu svého dokumentu

`\usepackage[`⟨*volby*⟩`]{fp}`

Ze souboru `*.log` se dozvíme jejich jména a další informace závislé na nepovinném parametru ⟨*volby*⟩, který může mít tvar:

`nomessages` - nevystupují žádné další informace,
`debug` - vystupují názvy a výsledky operací.

Je-li parametr ⟨*volby*⟩ prázdný, vystupují do souboru `*.log` názvy prováděných operací. I při použití volby `nomessages` lze operativně pro účel ladění jen části dokumentu zařadit do textu povely `\FPmessagetrue` či `\FPdebugtrue` pro zapnutí a `\FPmessagefalse` příp. `\FPdebugfalse` pro vypnutí podávání zpráv o prováděných operacích.

Všechny operace se realizují nad položkami ⟨*arg*⟩, které jsou buď řetězci číslic ve složených závorkách (např. `{-123.45}`) nebo povely (commands, ⟨*cmd*⟩), které tyto řetězce obsahují. Při tom se ⟨*cmd*⟩ definuje pomocí příkazu `\FPset`⟨*cmd*⟩⟨*arg*⟩, který je shodný s příkazem TeXu `\edef`⟨*cmd*⟩`{`⟨*arg*⟩`}`. Alternativně lze v případě potřeby použít i příkaz `\xdef`⟨*cmd*⟩`{`⟨*arg*⟩`}`. Příkazem `\FPset` se definují libovolné řetězce, konstanty, anebo i jiné příkazy.

Matematické konstanty

V balíku `fp` jsou předdefinovány dvě často používané konstanty, které jsou uloženy v povelech

`\FPpi%  = 3.141592653589793238 =` π **Ludolfovo číslo**
`\FPe %  = 2.718281828459045235` **základ přirozených logaritmů**

Další matematické konstanty jsou uvedeny v `fp-contrib.sty` (str. 9).

Binární aritmetické operace

`\FPadd` ⟨*cmd*⟩⟨*arg*$_1$⟩⟨*arg*$_2$⟩%	⟨*cmd*⟩ = ⟨*arg*$_1$⟩ + ⟨*arg*$_2$⟩	**sečítání**
`\FPsub` ⟨*cmd*⟩⟨*arg*$_1$⟩⟨*arg*$_2$⟩%	⟨*cmd*⟩ = ⟨*arg*$_1$⟩ − ⟨*arg*$_2$⟩	**odčítání**
`\FPmul` ⟨*cmd*⟩⟨*arg*$_1$⟩⟨*arg*$_2$⟩%	⟨*cmd*⟩ = ⟨*arg*$_1$⟩ ∗ ⟨*arg*$_2$⟩	**násobení**
`\FPdiv` ⟨*cmd*⟩⟨*arg*$_1$⟩⟨*arg*$_2$⟩%	⟨*cmd*⟩ = ⟨*arg*$_1$⟩ / ⟨*arg*$_2$⟩	**dělení**

Příklad:

```
\FPmul \twoPi{2}\FPpi     →     \twoPi  =  2 π
```

Výsledky aritmetických operací stejně jako výpočtů funkcí se ukládají do povelů ⟨*cmd*⟩, které se daným příkazem vytvoří. Výše uvedené aritmetické operace se označují jako *binární*, protože do nich vstupují dva argumenty – operandy.

Unární aritmetické operace

`\FPabs` ⟨*cmd*⟩⟨*arg*⟩% ⟨*cmd*⟩=|⟨*arg*⟩|	**absolutní hodnota argumentu**
`\FPneg` ⟨*cmd*⟩⟨*arg*⟩% ⟨*cmd*⟩=−⟨*arg*⟩	**negace argumentu**

Operaci `\FPneg` se lze vyhnout, uzavřeme-li argument do složených závorek, jako např. v příkazech

```
\FPmul\mtwoPi{-2}\FPpi  =  −2π,
\FPmul\mtwoPi{2}{-\FPpi} = −2π.
```

Elementární funkce

Nejjednoduššími funkcemi v knihovně maker `fp` jsou

`\FPsgn` ⟨*cmd*⟩⟨*arg*⟩%	**−1; 0; 1; znaménko argumentu**
`\FPmin` ⟨*cmd*⟩⟨*arg*$_1$⟩⟨*arg*$_2$⟩%	**menší z obou argumentů**
`\FPmax` ⟨*cmd*⟩⟨*arg*$_1$⟩⟨*arg*$_2$⟩%	**větší z obou argumentů**
`\FPclip`⟨*cmd*⟩⟨*arg*⟩%	**vypustí z** ⟨arg⟩ **závěrečné nuly za '.'**
`\FPround`⟨*cmd*⟩⟨*arg*$_1$⟩⟨*arg*$_2$⟩%	**zaokrouhlí na** ⟨arg$_2$⟩ **číslic za '.'**
`\FPtrunc`⟨*cmd*⟩⟨*arg*$_1$⟩⟨*arg*$_2$⟩%	**usekne** ⟨arg$_1$⟩ **na** ⟨arg$_2$⟩ **číslic za '.'**

Příklady:

```
\FPsgn \sign{-123.45}%      =  -1
\FPmin \Min{10}\FPe%        =  2.718281828459045235
\FPmax \Max\FPpi\FPe%       =  3.141592653589793238
\FPclip \clip{-123.4500}%   =  -123.45
\FPround \round\FPe{4}%     =  2.7183
\FPtrunc \res\FPe{4}%       =  2.7182
```

K dispozici jsou i makra pro výpočet hodnot trigonometrických, cyklometrických, exponenciálních, mocninných a logaritmických funkcí.

`\FPsin` ⟨*cmd*⟩⟨*arg*⟩	⟨*cmd*⟩ = sin(⟨*arg*⟩)	**sin**
`\FPcos` ⟨*cmd*⟩⟨*arg*⟩	⟨*cmd*⟩ = cos(⟨*arg*⟩)	**cos**
`\FPsincos` ⟨*cmd*$_1$⟩⟨*cmd*$_2$⟩⟨*arg*⟩	⟨*cmd*$_1$⟩= sin(⟨*arg*⟩) ⟨*cmd*$_2$⟩= cos(⟨*arg*⟩)	
`\FPtan` ⟨*cmd*⟩⟨*arg*⟩	⟨*cmd*⟩ = tan(⟨*arg*⟩)	**tg**
`\FPcot` ⟨*cmd*⟩⟨*arg*⟩	⟨*cmd*⟩ = cot(⟨*arg*⟩)	**cotg**
`\FPtancot` ⟨*cmd*$_1$⟩⟨*cmd*$_2$⟩⟨*arg*⟩	⟨*cmd*$_1$⟩= tan(⟨*arg*⟩) ⟨*cmd*$_2$⟩= cot(⟨*arg*⟩)	
`\FParcsin` ⟨*cmd*⟩⟨*arg*⟩	⟨*cmd*⟩ = arcsin(⟨*arg*⟩)	**arcsin**
`\FParccos` ⟨*cmd*⟩⟨*arg*⟩	⟨*cmd*⟩ = arccos(⟨*arg*⟩)	**arccos**
`\FParcsincos` ⟨*cmd*$_1$⟩⟨*cmd*$_2$⟩⟨*arg*⟩	⟨*cmd*$_1$⟩= arcsin(⟨*arg*⟩), ⟨*cmd*$_2$⟩= arccos(⟨*arg*⟩)	
`\FParctan` ⟨*cmd*⟩⟨*arg*⟩	⟨*cmd*⟩ = arctan(⟨*arg*⟩)	**arctg**
`\FParccot` ⟨*cmd*⟩⟨*arg*⟩	⟨*cmd*⟩ = arccot(⟨*arg*⟩)	**arccotg**
`\FParctancot` ⟨*cmd*$_1$⟩⟨*cmd*$_2$⟩⟨*arg*⟩	⟨*cmd*$_1$⟩= arctan(⟨*arg*⟩), ⟨*cmd*$_2$⟩= arccot(⟨*arg*⟩)	
`\FPexp` ⟨*cmd*⟩⟨*arg*⟩	⟨*cmd*⟩ = exp(⟨*arg*⟩)	**exponenciála**
`\FPln` ⟨*cmd*⟩⟨*arg*⟩	⟨*cmd*⟩ = ln(⟨*arg*⟩)	**přirozený log.**
`\FPpow` ⟨*cmd*⟩⟨*arg*$_1$⟩⟨*arg*$_2$⟩	⟨*cmd*⟩ = ⟨*arg*$_1$⟩$^{\langle arg_2 \rangle}$	**mocnina**
`\FProot` ⟨*cmd*⟩⟨*arg*$_1$⟩⟨*arg*$_2$⟩	⟨*cmd*⟩ = ⟨*arg*$_1$⟩$^{1/\langle arg_2 \rangle}$	**odmocnina**
`\FPseed=`⟨*arg*⟩		**násada pseudonáhodných čísel**
`\FPrandom`⟨*cmd*⟩		**pseudonáh. číslo** ⟨ 0, 1 ⟩

Na následujících řádkách je ukázáno použití těchto funkcí:

Příklady:

`\FPdiv\pih\FPpi{2}%`	= 1.570796326794896619	$\pi/2$
`\FPdiv\piq\FPpi{4}%`	= 0.785398163397448309	$\pi/4$
`\FPsin\res\pih%`	= 1.000000000000000000	$\sin(\pi/2)$
`\FPcos\res\piq%`	= 0.707106781186547525	$\cos(\pi/4)$
`\FPsincos\resa\resb\piq%`	= 0.707106781186547524	$\sin(\pi/4)$
	= 0.707106781186547525	$\cos(\pi/4)$
`\FParcsin\res{0.5}%`	= 0.523598775598298861	$\pi/6$
`\FPtan\res\piq%`	= 0.999999999999999998	$\tan(\pi/4)$
`\FPcot\res\piq%`	= 1.000000000000000001	`cot` $(\pi/4)$
`\FParccot\res{1}%`	= 0.785398163397448309	$\pi/4$
`\FPexp\res{-2}%`	= 0.135335283236612692	$1/e^2$
`\FPln \res\FPe%`	= 0.999999999999999989	`ln`(e)
`\FPpow\res{10}{-2}%`	= 0.009999999999999999	$1/100$
`\FProot\res{16}{-4}%`	= 0.500000000000000001	$1/\sqrt[4]{16}$
`\FPseed={\FPpi}%`	= 3.141592653589793238	π
`\FPrandom\res`	= 0.218418296993904885	náhodné číslo

Podmíněné příkazy – relační operátory

Často je zapotřebí větvit chod zpracování podle výsledku početních operací. Testování není zcela jednoduché, jsou-li výsledky řetězci znaků, jako je tomu ve stylu `fp.sty`. Naštěstí je v něm několik maker, kterými lze numerické řetězce testovat a následně větvit další postup zpracování:

`\FPifzero`⟨*arg*⟩`{`⟨*TRUE*⟩`}\else{`⟨*FALSE*⟩`}\fi%`	⟨*arg*⟩ = 0?
`\FPifneg` ⟨*arg*⟩`{`⟨*TRUE*⟩`}\else{`⟨*FALSE*⟩`}\fi%`	⟨*arg*⟩ < 0?
`\FPifpos` ⟨*arg*⟩`{`⟨*TRUE*⟩`}\else{`⟨*FALSE*⟩`}\fi%`	⟨*arg*⟩ > 0?
`\FPifint` ⟨*arg*⟩`{`⟨*TRUE*⟩`}\else{`⟨*FALSE*⟩`}\fi%`	⟨*arg*⟩ je celé?
`\FPiflt` ⟨arg_1⟩⟨arg_2⟩ `{`⟨*TRUE*⟩`}\else{`⟨*FALSE*⟩`}\fi%`	⟨arg_1⟩ < ⟨arg_2⟩?
`\FPifeq` ⟨arg_1⟩⟨arg_2⟩ `{`⟨*TRUE*⟩`}\else{`⟨*FALSE*⟩`}\fi%`	⟨arg_1⟩ = ⟨arg_2⟩?
`\FPifgt` ⟨arg_1⟩⟨arg_2⟩ `{`⟨*TRUE*⟩`}\else{`⟨*FALSE*⟩`}\fi%`	⟨arg_1⟩ > ⟨arg_2⟩?

V těchto příkazech `{`⟨*TRUE*⟩`}` představuje větev programu s příkazy, které se provedou v případě, že je podmínka splněna, kdežto `{`⟨*FALSE*⟩`}` větev s příkazy vykonávanými při jejím nesplnění.
Poznámka: Zde je třeba upozornit na skutečnost, že tyto podmíněné příkazy pracují jen v přímých úsecích programu, *nikoliv však v cyklech!*

Vyhodnocování výrazů

Dosud probírané povely umožňovaly realizaci výpočtů jako kdysi v minulosti v tzv. autokódu, totiž po jednotlivých operacích. Ve stylu `fp.sty` je však zabudován mohutný nástroj, jímž lze vyhodnocovat celé formule v rámci jednoho příkazu. Existují dvě možnosti, jak vyhodnotit složitější matematický výraz bez rozkladu na jednotlivé dosud popsané instrukce. Za tím účelem je `fp.sty` vybaven dvěma příkazy, `\FPeval` a `\FPupn`, které si postupně popíšeme. V originálním materiálu existuje řada způsobů, jak zapisovat jednotlivé položky k vyhodnocení výrazů, ale zde si uvedeme pouze jeden, patrně nejjednodušší. V obou příkazech je třeba dodržovat jisté zásady pro zápis *výrazů* ve složených závorkách:

- `FP`-funkce zapisujeme *bez* úvodních znaků „`\FP`“. Tak např. místo `\FPsin` budeme zapisovat pouze `sin`,
- ostatní (uživatelovy) povely – proměnné – se zapíší *bez* úvodního zpětného lomítka, např. místo `\result` pouze `result`,
- pokud položka má začínat znaménkem minus, *musí* se zapsat do složených závorek stejně jako argumenty v makrech např. `{-#2}`,
- místo názvů funkcí `add, sub, mul, div` lze užít běžné znaky operátorů + − ∗ /. `fp-contrib` umožnil užít znak ^ vedle `pow`.

`\FPeval` ⟨*cmd*⟩`{`⟨*výraz*⟩`}` **Vyhodnocení výrazu**

Útvar ⟨*výraz*⟩ ve složených závorkách může obsahovat kromě výše uvedených jmen povelů (bez zpětných lomítek), jmen funkcí (bez `\FP`) a symbolů operací, ještě závorky a případné oddělovače. Oddělovačem bez účinku jsou mezery. Čárky a dvojtečky slouží pro oddělování parametrů funkcí a příp. i postupných kroků výpočtu, protože je třeba mít na paměti, že vždy je možno realizovat pouze operaci mezi dvěma operandy, která dá jeden výsledek. Již tato vágní pravidla ukazují, že postup nemusí být zcela spolehlivý a ani při průchodu bez hlášení chyb nemusí být výsledek správný. Jako příklad uveďme vyhodnocení výrazu $A = \sqrt[3]{[(\pi + e) * 10]^2}$:

Postupnými operacemi bychom vypočetli výsledek do `\res` takto

```
\FPadd\A\FPpi\FPe \FPmul\A\A{10}%
\FPpow\A\A{2}\FProot\A\A{3}% =      15.086628946044704893
```

Naproti tomu zápis pomocí `\FPeval` se zapíše daleko přehledněji. Problémem je změna pořadí u argumentů ve funkcích `\FProot` a `FPeval`:

```
\FPeval\A{root(3,(pi+e)*10)^2} % =  15.086628946044704420
```

Proč je u `\FPeval` obrácené pořadí proti `FProot`, není zřejmé. Patrně to bude z důvodu řazení položek pro příkaz `\FPupn`, který `\FPeval` volá pro vyhodnocení výrazu.

`\FPupn` ⟨*cmd*⟩`{`⟨*posloupnost položek*⟩`}` **Rozklad výrazu na prvky**

Příkaz dostal jméno z německého **U**mgekehrt **P**olisch **N**otation – reverzní polská notace. Jde o způsob rozkladu obecného výrazu do posloupnosti operandů a operátorů. Zatímco kompilátory vyšších programovacích jazyků rozloží výraz na ⟨*posloupnost položek*⟩ automaticky, v `\FPupn` je tato činnost ponechána na uživateli. Ten musí řadit součásti výrazu, tj. veličiny ⟨*arg*⟩ a operátory operací a funkcí ⟨*ope*⟩ do posloupnosti položek za sebou tak, aby nepotřeboval závorky. Položky jsou od sebe odděleny mezerami a jsou za sebou řazeny tak, aby mohly být postupně vkládány do speciální části paměti, tzv. zásobníku. Název vznikl z podobnosti její funkce s funkcí zásobníků zbraní. Prvky z posloupnosti položek postupně vstupují na tzv. vrchol zásobníku (`TOS` – top of stack) tak, že dřívější položky v zásobníku budou nyní ležet o pozici níže (`TOS` -1, `TOS` $-2, \ldots$). Jakmile se obsadí nový `TOS` aritmetickým nebo funkčním operátorem ⟨*ope*⟩, zkontroluje se stav pod vrcholem zásobníku a pokud je to možné, provede se poža-

dovaná operace, použité položky se ze zásobníku vypustí a výsledek $\langle res \rangle$ se uloží na místo nově uprázdněného vrcholu `TOS`. Tak např. pro *unární operaci* bude situace na vrcholu zásobníku následující:

$$\begin{array}{rl} & \text{starý } \texttt{TOS} \downarrow \\ \text{stav před operací} : & \langle \textit{nezpracované položky } arg \rangle \langle \texttt{arg} \rangle \langle ope \rangle \\ \text{stav po operaci} : & \langle \textit{nezpracované položky } arg \rangle \langle res \rangle \\ & \text{nový } \texttt{TOS} \uparrow \end{array}$$

Podobně dopadne zásobník při *binární operaci*:

$$\begin{array}{rl} & \text{starý } \texttt{TOS} \downarrow \\ \text{stav před operací} : & \langle \textit{nezpracované položky } arg \rangle \langle \texttt{arg}_2 \rangle \langle \texttt{arg}_1 \rangle \langle ope \rangle \\ \text{stav po operaci} : & \langle \textit{nezpracované položky } arg \rangle \langle res \rangle \\ & \text{nový } \texttt{TOS} \uparrow \end{array}$$

Všimněme si řazení položek v zásobníku. Nejvýše (v `TOS`) leží operátor a pod ním $\langle \texttt{arg}_1 \rangle$ v $\texttt{TOS}-1$, $\langle \texttt{arg}_2 \rangle$ v $\texttt{TOS}-2$ a dále až dosud nevyužité argumenty, pokud ještě existují. Bohužel, existuje výjimka proti tomuto pravidlu, a to u funkcí `round` a `trunc`, u kterých se počet desetinných míst uvádí těsně před jejich pozicí v posloupnosti na rozdíl od funkcí `\FPround` a `\FPtrunc`, u nichž stojí jako $\langle \texttt{arg}_2 \rangle$. Jako příklad zpracování uveďme vyhodnocení výrazu řešeného příkazem `\FPeval`.

```
\FPupn \res{3 2 pi e + 10 * pow root 6 round}% = 15.086629
                ------
                ------------
              ------------------
            ----------------------
```

Vodorovné podtrhání položek posloupnosti vyznačuje pořadí, v němž se vyhodnocuje výraz. Podtržená část dá jeden výsledek, který se umístí na místo prvního z operandů jako nový `TOS`. S použitím závorek by zápis výrazu měl poněkud jiný tvar: $(((\pi + e) * 10)\hat{}\, 2)\hat{}\, (1/3)$.

Uživatel má možnost v případě potřeby ošetřovat vrchol zásobníku. Slouží k tomu následující tři funkce:

`copy %` okopírování obsahu `TOS` na místo $\texttt{TOS} + 1$ jako nový `TOS`,
`swap %` prohození položek z `TOS` a $\texttt{TOS} - 1$,
`pop %` vypuštění `TOS`; $\texttt{TOS} - 1$ se stane novým `TOS`.

Funkce `copy` se uplatní zejména při výpočtu kvadrátu násobením. Po vyčerpání celé posloupnosti položek zůstane v zásobníku jediná položka - výsledek. Pokud se zásobník vyčerpá, anebo v něm po skončení zpracování zůstane více položek než jediná, je hlášena chyba při běhu programu.

Uveďme ještě přehled základních operátorů použitelných v příkazech `\FPeval` a `\FPupn`. Balík `fp-contrib` přidává další konstanty a funkce.

```
+,add,-,sub,*,mul,/,div,^,pow,
abs,neg,sgn,min,max,round,trunc,clip,
pi,e,exp,ln,pow,root,
sin,cos,sincos,tan,cot,tancot,arcsin,
arccos,arcsincos,arctan,arccot,arctancot,
pop,swap,copy
```

Řešení algebraických rovnic

Výše uvedenými funkcemi není ještě zcela vyčerpaná paleta možností, které skýtá knihovna `fp`. Obsahuje totiž ještě čtyři makra pro řešení reálných kořenů reálných rovic prvního až čtvrtého stupně. Na výstupu z maker jsou kořeny uloženy v $\langle \mathtt{cmd}_i \rangle$. Autor `fp` však nezaručuje správnost výsledků, pokud kořeny nejsou reálné. Jedná se o tato makra:

`\FPlsolve` $\langle \mathtt{cmd}_1 \rangle \langle \mathtt{arg}_1 \rangle \langle \mathtt{arg}_2 \rangle$
$\langle \mathtt{arg}_1 \rangle\, x + \langle \mathtt{arg}_2 \rangle = 0$ **lineární**

`\FPqsolve` $\langle \mathtt{cmd}_1 \rangle \langle \mathtt{cmd}_2 \rangle \langle \mathtt{arg}_1 \rangle \langle \mathtt{arg}_2 \rangle \langle \mathtt{arg}_3 \rangle$
$\langle \mathtt{arg}_1 \rangle\, x^2 + \langle \mathtt{arg}_2 \rangle\, x + \langle \mathtt{arg}_3 \rangle = 0$ **kvadratická**

`\FPcsolve` $\langle \mathtt{cmd}_1 \rangle \langle \mathtt{cmd}_2 \rangle \langle \mathtt{cmd}_3 \rangle \langle \mathtt{arg}_1 \rangle \langle \mathtt{arg}_2 \rangle \langle \mathtt{arg}_3 \rangle \langle \mathtt{arg}_4 \rangle$
$\langle \mathtt{arg}_1 \rangle\, x^3 + \langle \mathtt{arg}_2 \rangle\, x^2 + \langle \mathtt{arg}_3 \rangle\, x + \langle \mathtt{arg}_4 \rangle = 0$ **kubická**

`\FPqqsolve` $\langle \mathtt{cmd}_1 \rangle \langle \mathtt{cmd}_2 \rangle \langle \mathtt{cmd}_3 \rangle \langle \mathtt{cmd}_4 \rangle \langle \mathtt{arg}_1 \rangle \langle \mathtt{arg}_2 \rangle \langle \mathtt{arg}_3 \rangle \langle \mathtt{arg}_4 \rangle \langle \mathtt{arg}_5 \rangle$
$\langle \mathtt{arg}_1 \rangle\, x^4 + \langle \mathtt{arg}_2 \rangle\, x^3 + \langle \mathtt{arg}_3 \rangle\, x^2 + \langle \mathtt{arg}_4 \rangle\, x + \langle \mathtt{arg}_5 \rangle = 0$ **4. stupně**

Rovnice jsou splněny v bodech řešení $x = \langle \mathtt{cmd}_i \rangle$ (pokud existují).

Rozšiřující styl `fp-contrib.sty`

Autor tohoto příspěvku potřeboval pro svoji činnost pracovat ještě s dalšími funkcemi, které nejsou součástí balíku `fp`. Protože nemohl zasahovat do `fp.sty` a jeho modulů, zařadil nové funkce do nového modulu `fp-contrib.sty`. Pro uživatele libovolné z jeho funkcí stačí, když v „předmluvě“ ke svému dokumentu uvede pouze

```
\usepackage[volby]{fp-contrib},
```

který zastoupí všechny moduly `fp.sty`, protože si je sám zavede. Nepovinný parametr `volby` má stejný význam jako u originálního balíku `fp`

(viz výše). Knihovna `fp-contrib.sty` definuje další dvě *konstanty*:

`\FPeps` nejmenší kladné zobrazitelné číslo, rozlišitelnost `fp.sty`
`\FPloge` dekadický logaritmus čísla e; `M = 0.434294481903251828`

Dále jsou zpracována makra pro výpočet *funkcí* důležitých jak pro kreslení diagramů tak i pro statistiku normálního rozdělení.

`\FPatan` ⟨*cmd*⟩⟨*arg*$_1$⟩⟨*arg*$_2$⟩ `%` ⟨*cmd*⟩= arctg(⟨*arg*$_2$⟩/⟨*arg*$_1$⟩) **arctg**
`\FPlog` ⟨*cmd*⟩⟨*arg*⟩ `%` ⟨*cmd*⟩ = $\log_{10}(x)$ **dekad. logaritmus**
`\FPsinh` ⟨*cmd*⟩⟨*arg*⟩ `%` ⟨*cmd*⟩ = $\sinh(x)$, **hyperbolický sinus**
`\FPcosh` ⟨*cmd*⟩⟨*arg*⟩ `%` ⟨*cmd*⟩ = $\cosh(x)$, **hyperbolický kosinus**
`\FPdiff` ⟨*cmd*⟩⟨*arg*⟩ `%` ⟨*cmd*⟩ = $\sin(x)/x$, **difrakční funkce**
`\FPerf` ⟨*cmd*⟩⟨*arg*⟩ `%` ⟨*cmd*⟩ = $\mathrm{erf}(x)$, **chybová funkce**
`\FPpdn` ⟨*cmd*⟩⟨*arg*⟩ `%` ⟨*cmd*⟩ = $p_n(x)$, **hustota pravděpodob.**
`\FPPnx` ⟨*cmd*⟩⟨*arg*⟩ `%` ⟨*cmd*⟩ = $P_n(X \le x)$, **pravděpodobnost**
`\FPxPn` ⟨*cmd*⟩⟨*arg*⟩ `%` ⟨*cmd*⟩ = $x_P(P_n)$, **arg. pravděpodobnosti**
`\FPppt` ⟨*cmd*⟩⟨*arg*$_1$⟩`%` ⟨*cmd*⟩ = $z_{1-0,05}(\nu)$, **90% kvantil rozdělení** t
`\FPdtor` ⟨*cmd*⟩⟨*arg*⟩ `%` ⟨*cmd*⟩ = $\alpha\ [deg]$, **stupně → radiány**
`\FPrtod` ⟨*cmd*⟩⟨*arg*⟩ `%` ⟨*cmd*⟩ = $\alpha\ [rad]$, **radiány → stupně**
`\FPrandn`⟨*cmd*⟩ `%` ⟨*cmd*⟩ = **pseudonáhod. číslo s norm. rozděl.**

Styl `fp-contrib` obsahuje ještě další makra zvláštního určení:

`\FPpoly` ⟨*cmd*⟩=⟨*arg*⟩(⟨*seznam*⟩) **Hodnota polynomu**
Makrem se počítají hodnoty polynomu pro danou hodnotu argumentu. Parametr ⟨*seznam*⟩ v kulatých závorkách je seznamem koeficientů polynomu od nejvyšší mocniny počínaje. Prvky seznamu jsou reálná čísla oddělená navzájem čárkou.

`\FPnsolve` ⟨*cmd*⟩⟨*arg*$_1$⟩⟨*arg*$_2$⟩⟨*arg*$_3$⟩⟨*arg*$_4$⟩⟨*arg*$_5$⟩ **Kořen funkce**
Toto makro doplňuje dříve uvedená makra pro řešení polynomiálních rovnic do 4. stupně o řešení reálného kořene obecné reálné nelineární funkce $f(x)$ jedné proměnné, jejíž makro dodá uživatel. Přitom musí zajistit, že její deklarace bude mít tvar

`\def\funx #1#2{`⟨*tělo funkce*⟩`}`,

kde `#1` je povel, který na výstupu z makra ponese hodnotu funkce f(`#2`). Odtud plyne, že parametr `#2` zde zastupuje výše uvedený argument x funkce $f(x)$. Kořen se hledá rychlou stabilní metodou, v níž se po nalezení intervalu s opačnými znaménky funkce střídá iterační krok metody regula falsi s bisekcí intervalu.

`\FPnsolve` má následující parametry:

⟨*cmd*⟩	`\root`	povel (proměnná) pro uložení hodnoty kořene,
⟨*arg*$_1$⟩	`x`$_a$	počátek intervalu hledání kořene
⟨*arg*$_2$⟩	`dx`	počáteční krok prohledávání intervalu od x_a
⟨*arg*$_3$⟩	`tolx`	povolená tolerance na rozdíl dvou iterací v x
⟨*arg*$_4$⟩	`tolf`	povolená tolerance na rozdíl dvou iterací v $f(x)$
⟨*arg*$_5$⟩	`maxit`	maximální povolený počet iterací

`\For` ⟨*cmd*⟩=(⟨*seznam cyklů*⟩)`{`⟨*tělo cyklu*⟩`}` **Obecný cyklus**

Jde o příkaz obecného cyklu typu `for`, který je zapotřebí zapsat tak, je uveden výše. To znamená, že nejsou povoleny mezery před a za rovnítkem, a že seznam položek je v kulatých závorkách, zatímco tělo cyklu ve složených závorkách! Příkaz `\For` má následující parametry:

⟨*cmd*⟩	proměnná cyklu,
⟨*seznam cyklů*⟩	seznam položek oddělených čárkami pro proměnnou cyklu, které mohou mít tvar (i v kombinaci) ⟨*arg*⟩, nebo intervalu ⟨*arg*$_1$⟩:⟨*arg*$_2$⟩:⟨*arg*$_3$⟩, v němž krajní argumenty jsou počáteční a konečné hodnoty intervalu lineárně rozděleného krokem ⟨*arg*$_2$⟩,
⟨*tělo cyklu*⟩	je skupina příkazů, které se mají opakovat.

Uživatel může používat proměnnou cyklu v těle cyklu, ale v žádném případě ji nesmí měnit! Uveďme však cyklus, který to nesplňuje

```
\For \I=(1, 2:.5:4, 5){\FPclip\I\I \ \ \I}
```

a vysází do dokumentu řadu čísel 1 2 2.5 3 3.5 4 5.

Zdálo by se, že výše uvedená zásada nedotknutelnosti proměnné cyklu zde byla porušena, protože proměnná `\I` byla použita ve funkci jako výstupní parametr. Protože však funkce `\FPclip` pouze odtrhává nadbytečné nuly za poslední desetinnou číslicí a nemění tím hodnotu proměnné cyklu, nemohlo dojít k chybě.

Již výše bylo upozorněno, že podmíněné příkazy `\FPif`⋯ nemohou být použity v cyklech, protože u nich dochází k chybě. Proto byly sestaveny ještě alternativní podmíněné příkazy, které lze v cyklech použít:

`\ifAltB` ⟨arg$_1$⟩⟨arg$_2$⟩`{`⟨*TRUE*⟩`}{`⟨*FALSE*⟩`}` **Větvení při** ⟨arg$_1$⟩<⟨arg$_2$⟩

Příkaz srovnává řetězce s číselným obsahem. Pokud hodnota v ⟨*arg*$_1$⟩ je menší než v ⟨*arg*$_2$⟩, vykoná příkazy z větve `{`⟨*TRUE*⟩`}` a v opačném případě z větve `{`⟨*FALSE*⟩`}`.

`\ifAeqB` ⟨*arg*$_1$⟩⟨*arg*$_2$⟩`{`⟨*TRUE*⟩`}{`⟨*FALSE*⟩`}` **Větvení při** ⟨arg$_1$⟩=⟨arg$_2$⟩
Příkazem se testuje shoda hodnot v obou argumentech. Pokud jí je dosaženo, vykoná se větev `{`⟨*TRUE*⟩`}` a není-li, pak větev `{`⟨*FALSE*⟩`}`.

`\ifAgtB` ⟨*arg*$_1$⟩⟨*arg*$_2$⟩`{`⟨*TRUE*⟩`}{`⟨*FALSE*⟩`}` **Větvení při** ⟨arg$_1$⟩>⟨arg$_2$⟩
Je-li hodnota v ⟨*arg*$_1$⟩ větší než v ⟨*arg*$_2$⟩, vykoná se větev `{`⟨*TRUE*⟩`}` a není-li, pak větev `{`⟨*FALSE*⟩`}`.

Obrázky a diagramy představují nejnázornější prostředky pro vyjádření výsledků. Proto odborné publikace jsou bez obrázků takřka nemyslitelné. Bohužel, základní výbava TEXu s obrázky vůbec nepočítala a o mnoho lepší situace není ani u LATEXu. Není proto divu, že v průběhu času vznikly prostředky, které umožňují doplnit dokument obrázky. Nebudeme se zde zabývat importovanou grafikou, ale věnujeme pozornost dvěma stylům, které umožňují vytvářet obrázky přímo z prostředí právě zpracovávaného dokumentu.

Nejdříve popišme knihovnu `curves.sty`, která je i významným nástrojem pro druhý balík, `diagram.sty`. Oba styly lze používat v běžném prostředí LATEXu – `picture`.

Jednoduchý styl pro kreslení křivek – `curves.sty`

Instalační soubory `curves.dtx` a `curves.ins` lze nalézt v podadresáři `texmf\source\latex\curves`. Ty po přeložení pomocí LATEXu vytvoří jak `curves.sty`, tak i dokumentaci `curves.dvi`, která je základním referenčním manuálem. Soubor `curves.sty` umístíme do míst, kde ho překladač LATEXu po aktualizaci databáze najde.

První vydání `curves.sty` spatřilo světlo světa v roce 1995. Dnes je k dispozici novější vydání z roku 2000 [2]. Jak již název napovídá, umožňuje kreslit hladké křivky, ale na rozdíl od jiných realizací, jako jsou např. styly, příp. makra `bezier, qbezier, bez123, ...` procházejí jeho křivky – paraboly – všemi zadanými body bez nutnosti definovat pomocné body. Tím ale nekončí možnosti tohoto stylu. Nezávislými měřítky mezi oběma hlavními směry kreslení umožňuje rotovat s objekty o libovolný úhel anebo je i deformovat. V zadaných bodech může vykreslovat symboly (markery) a pod. Protože pracuje v prostředí `picture`, lze používat souběžně i všechny příkazy tohoto prostředí tak, jak jsou definovány v LATEXu. Zde popíšeme jen nejdůležitější příkazy.

Čáry jsou vytvářeny kladením plných kroužků k sobě s takovou roztečí, aby uživatel měl dojem hladké křivky, i když je její obrys mírně zvlněný. Rozteč je řízena povelem `\diskpitchstretch` původně nastaveným na hodnotu `1`. Pro zrychlené vytváření dokumentu lze tuto proměnnou nastavit na libovolné reálné číslo větší jak jedna, např. příkazem

`\Cmd{`⟨*cmd*⟩`}={`⟨*reálné číslo*⟩`}`,

kde příkaz ⟨*cmd*⟩ je `\diskpitchstretch`. Zvolíme-li ho rovné 5, zrychlí se vytváření čar také přibližně 5×, ale čáry zhrubnou, anebo dokonce budou tečkované. Zjemnění čar dostaneme jeho zmenšením pod jednotku za cenu prodloužení času vynášení čar.

Základními příkazy `curves.sty` jsou

`\scaleput` (⟨arg_1⟩,⟨arg_2⟩)`{`⟨*objekt*⟩`} %` **Umisťovací příkaz**

Má shodný tvar s příbuzným příkazem `\put`, ale na rozdíl od něho jsou všechny rozměry v rámci tohoto příkazu ovlivněny nastavením měřítek. To se týká jak argumentů tak i grafického objektu, pokud byly vytvořeny příkazy stylu `curves`. Počáteční nastavení měřítek:

`\xscale = 1`, `\xscaley = 0`, `\yscalex = 0`, `\yscale = 1`.

Nové rozměry objektu i jeho umístění budou podléhat transformaci

$$\begin{aligned} x' &= \texttt{\textbackslash xscale} \times x + \texttt{\textbackslash xscaley} \times y, \\ y' &= \texttt{\textbackslash yscalex} \times x + \texttt{\textbackslash yscale} \times y, \end{aligned}$$

což lze zapsat v maticovém tvaru

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \texttt{\textbackslash xscale , \textbackslash xscaley} \\ \texttt{\textbackslash yscalex, \textbackslash yscale} \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix}.$$

Měřítka se mění pomocí příkazu `\Cmd` (viz str. 44) nebo `\edef` z TEXu.

Příklad:

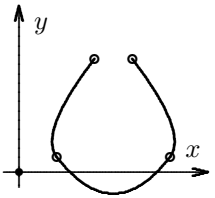
```
\Cmd\xscale={0.5}%
\edef\yscale{-.75}%
\put(0,0){\circle*{1}}
\put(25,8){\circle{1}}
\scaleput(25,8){\circle{1}}
```

Horní kroužek byl vynesen příkazem `\put` a dolní (při stejném zadání) příkazem `\scaleput`. Změna měřítek vůbec neovlivnila umís-

tění kroužku příkazem \put, zatímco poloha kroužku vyneseného příkazem \scaleput byla plně ovlivněna použitými měřítky.

\curve[⟨počet⟩](⟨seznam souřadnic⟩)% **Vykreslování křivky**

Příkazem se vykreslí křivka procházející body, jejichž souřadnice x, y jako dvojice reálných čísel jsou uvedeny v pořadí, v jakém mají být pospojovány. Čísla jsou navzájem oddělena čárkami a vytvářejí tak seznam uzavřený v kulatých závorkách.



Modul prvního nepovinného parametru, ⟨počet⟩, udává, kolik symbolů bude v každém podúseku křivky vyneseno. Tloušťka čáry se řídí standardními příkazy \linethickness{⟨len⟩}, \thinlines a \thicklines. Zde ⟨len⟩ je tloušťka čáry s rozměrem z intervalu 0.5pt až 15pt. Jako příklad použití makra \curve uveďme \put(0,0){\curve(10,15, 5,2, 20,2, 15,15)}, který vykreslí křivku plnou čarou. Počátek (0,0) je vyznačen plným kroužkem. Chceme-li navíc vykreslit i zadané body anebo změnit charakter čáry, je zapotřebí vyvolat dva další příkazy.

\curvesymbol{⟨symbol⟩}% **Definice symbolu**

Tímto příkazem se definuje *grafický* symbol, který má být vykreslen. Všimněme si, že se název symbolu v něm vyskytuje dvakrát. První výskyt, v příkazu \phantom vyhrazuje symbolu místo, druhý pak definuje vlastní symbol. Pokud se opomene příkaz \phantom, budou symboly posunuty zhruba o polovinu své šířky doleva. Druhý příkaz je opět příkaz \curve, ve kterém se udávají souřadnice bodů, v nichž budou uvedené symboly vykresleny. Pro náš příklad to uskutečnil povel \put(0,0)\curve[-1](10,15, 5,2, 20,2, 15,15). Na rozdíl od pouhého kreslení čáry se v něm vyskytuje navíc jako nepovinný argument údaj ⟨počet⟩, jímž se interpretuje zadaný seznam souřadnic bodů. Má-li ⟨počet⟩ jméno \n, pak pro

- \n<0, vykreslí se jen žádané symboly na hranicích |\n| podúseků, na něž se úseky dané sousedními body rozdělí, a čára se nevykreslí;
- \n=0, vykreslí se pouze čára procházející danými body;
- \n>0, úsek mezi každými dvěma zadanými body se rozdělí na \n podúseků. Na jejich hranicích se vynesou jen *tečky*. Jejich velikost je řízena tlouštkou čáry, takže je třeba pamatovat na to, že při

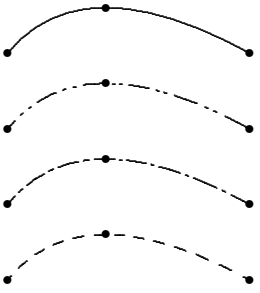
kreslení čáry jiným příkazem `\curve` budou překresleny a nebudou vidět.

`\curvedashes[`⟨*unit len*⟩`]{`⟨*blank len*⟩`,`⟨*struktura čáry*⟩`}%`

Definování struktury vykreslované čáry

Slouží k ovládání vzhledu čáry. Parametry tohoto příkazu jsou: ⟨*unit len*⟩ = jednotka délek, v nichž bude definována struktura čáry. Pokud tento parametr nebude uveden, použije se `\unitlength`; ⟨*blank len*⟩= délka prázdného místa na počátku kreslené čáry; ⟨*struktura čáry*⟩ = seznam délek plných a prázdných úseků vykreslované čáry v jednotkách ⟨*unit len*⟩. Položky v seznamu jsou oddělované čárkami. Jako příklad uveďme serii čtyř stejných křivek vykreslených různým stylem čar:

```
\curvesymbol{\phantom{\circle*{1}}\circle*{1}}%  symbol bodu
\newcommand{\curv}{0,0, 13,6, 32,0}%         definice křivky
\put(0,0){\curve[-1](\curv)}%
\put(0,10){\curve[-1](\curv)}%
\put(0,20){\curve[-1](\curv)}%
\put(0,30){\curve[-1](\curv)}%
\curvesymbol{}
\curvedashes[1pt]{0, 2,4,2}%
\put(0,0){\curve(\curv)}%
\curvedashes[1pt]{0, 4,2,2,2,4}%
\put(0,10){\curve(\curv)}%
\curvedashes[1pt]{0, 4,3,1,3,1,3,4}%
\put(0,20){\curve(\curv)}%
\curvedashes[0pt]{}%
\put(0,30){\curve(\curv)}%
```



Je třeba zde upozornit, že definice struktury čáry platí stále. Je tedy třeba pro následné kreslení čar plnou čarou anebo vynášení bodů příkaz `\curvedashes` zrušit. S ohledem na relativní složitost podmínek, za kterých probíhá kreslení, je vhodné dodržovat následující zásady:

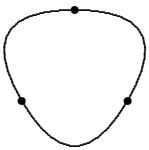
- Nejdříve zadat ⟨*unit len*⟩ = 0pt příkazem `\curvedashes[0pt]{}` (pro jistotu). Pokud je to žádáno, vybrat symbol pro vyznačení bodů a nakonec je vykreslit příkazem `\curve[`⟨*počet*⟩`](`⟨*seznam*⟩`)`.
- Poté nastavit, pokud nemá být čára plná, vhodnou strukturu čáry příkazem `\curvedashes` podle výše uvedeného postupu a *zrušit*

dříve nastavený symbol příkazem `\curvesymbol{}`. Teprve potom vyvolat kreslení čáry.

Dalšími příkazy pro kreslení křivek jsou

`\closecurve[`⟨*počet*⟩`](`⟨*seznam souřadnic*⟩`)%` **Uzavřená křivka**

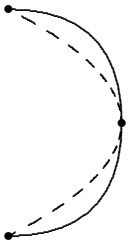
Příkaz kreslí uzavřenou křivku. Je zapotřebí zadat minimálně tři body (6 souřadnic), kterými má křivka procházet. Význam parametrů je stejný jako u příkazu pro kreslení křivek – `\curve`. Obrázek byl nakreslen příkazy



```
\closecurve[-1](0,0, 7,12, 14,0)
\closecurve(0,0, 7,12, 14,0)
```

`\tagcurve[`⟨*počet*⟩`](`⟨*seznam souřadnic*⟩`)%` **Křivka s převisy**

Příkaz slouží k vykreslení křivky bez prvního a posledního úseku. Je určen k vykreslení křivek, u nichž chceme vhodnými krajními úseky zajistit požadované sklony u viditelných částí . Je zapotřebí zadat minimálně čtyři body (8 souřadnic), kterými má křivka procházet. V případě, že zadáme pouze 6 souřadnic, dodrží se požadovaný sklon pouze na konci křivky. Význam parametrů je stejný jako u příkazu `\curve` pro kreslení křivek. Pro vedlejší obrázek bylo zadáno 5 bodů (první a poslední totožné):

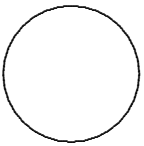


```
\tagcurve(-15,0, 0,-15, 15,0, 0,15, -15,0)
\tagcurve[-1](-15,0, 0,-15, 15,0, 0,15, -15,0)
```

V obrázku je čárkovaně vynesena i čára, která by se vykreslila z viditelných bodů při užití příkazu `\curve`.

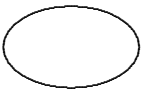
`\bigcircle[`⟨*počet*⟩`]{`⟨*průměr*⟩`}%` **Kružnice**

LaTeXovské omezení na průměr vykreslované kružnice odstraňuje tento příkaz. Průměr se udává v jednotkách `\unitlength`. Parametr ⟨*počet*⟩ je stejný jako u ostatních příkazů pro kreslení křivek. Příklad:



```
\put(0,0){\bigcircle{18}}
```

Pokud bychom změnili měřítko `\yscale`, řekněme na `0.6`, změní se kružnice na elipsu:

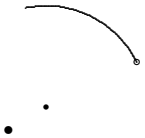


```
\Cmd\yscale={0.6}
\put(0,0){\bigcircle{18}}
```

`\arc[`⟨*počet*⟩`](X1,Y1){`⟨*úhel*⟩`}%` **Kruhový oblouk**

Příkazem se kreslí oblouk, jehož střed křivosti je v bodě daném příkazem `\put` nebo `\scaleput` a počátek oblouku v bodě `(X1,Y1)`. Koncový bod oblouku se ustaví tak, že úhel oblouku dosáhne hodnoty ⟨*úhel*⟩ ve stupních. V následujícím příkladu je plným velkým kroužkem označen počátek, malým střed oblouku a prázdným kroužkem počáteční bod `(X1,Y1)`:

```
\put(0,0){\circle*{1}}%     počátek
\put(5,3){\circle*{.75}}%   střed oblouku
\put(17,9){\circle{.75}}%   (X1,Y1)
\put(5,3){\arc(12,6){75}}%  oblouk
```



`\bezier{`⟨*počet*⟩`(X1,Y1)(X2,Y2)(X3,Y3)%` **Bezierova křivka**

Příkaz nakreslí Bezierovu křivku jdoucí koncovými body `(X1,Y1)` a `(X3,Y3)` s tečnou v bodě `(X2,Y2)` rovnoběžnou se spojnicí krajních bodů. Jde o rozšířenou a zrychlenou verzi stejně pojmenovaného makra z LaTeXu.

Ale pozor! Nesmí se kombinovat se standardním stylem `bezier.sty`!

Zde uvedený popis dává pouze stručnou informaci k užití příkazů z balíku `curves.sty`. Popis dalších příkazů tohoto stylu lze najít v manuálu [2]. Pozorný čtenář si jistě všimne, že se neuváděly žádné volitelné parametry balíku, i když v manuálu jsou uvedeny hned čtyři. To proto, že umožňují používat pro vytváření `*.dvi` souborů příkazy typu `\special`, které však nemusí být rozpoznány různými prohlížeči, takže se nakonec pracně vytvořený grafický útvar nemusí objevit ani na obrazovce, ani třeba v `*.pdf` souboru výsledného dokumentu. Proto v balíku `diagram.sty` není možno žádnou z voleb použít.

Knihovna maker `diagram.sty` dále rozvíjí možnosti kreslení a usnadňuje uživatelovu práci při vytváření dokumentu s přílohami.

Podpora výpočtů a kreslení – `diagram.sty`

Nový styl pro kreslení diagramů dostal název `diagram`. Jeho definice umožňuje, aby si uživatel nemusel pamatovat všechny knihovny maker, které celý systém kreslení bude potřebovat. Jemu bude pro všechny dosud probírané aplikace stačit napsat do úvodní části programu pouze

`\usepackage[`⟨*FP volby*⟩`](diagram)`

Nepovinný argument ⟨*FP volby*⟩ je identický s parametrem ⟨*volby*⟩ u `FP` stylu, jak je popsáno na straně 3. Vyžádáním knihovny `diagram.sty` se zajistí i zavedení balíků `support.sty`, `graphicx.sty`, `curve.sty` a `fp-contrib.sty` a s jeho pomocí i `fp.sty` a `ifthen.sty`. Z těchto důvodů žádný z uvedených stylů již ve svém dokumentu *nežádáme*! Jiné balíky si ovšem uživatel zavádí pomocí příkazů `\usepackage` podle potřeby. Makra jsou uspořádána do logických skupin, které také tak postupně probereme. Řada z nich, pokud se osvědčila, byla převzata ze starého systému maker popsaného v [1].

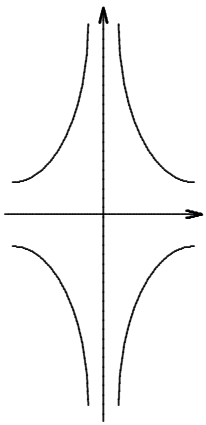
Úpravy měřítek

Práce s měřítky je velmi důležitá. Proto byly původní možnosti měřítek velmi rozšířeny.

`\setscales(`⟨arg_1⟩`,`⟨arg_2⟩`,`⟨arg_3⟩`,`⟨arg_4⟩`)` **Nastavení všech měřítek**
Jde o nejčastější úpravu, která v předložené verzi šetří čas programování i zápisu a odstraňuje časté chyby způsobované nepřehlednými názvy měřítek. Argumenty v *kulatých* závorkách jsou prvky matice měřítek zapsané po řádkách (viz stranu 13). Normální měřítka se nastaví povelem `\setscales(1,0,0,1)`.

`\mirrorscale{`⟨*osa*⟩`}` **Zrcadlení kolem vybrané osy**
Argument makra ⟨*osa*⟩ je jedno z písmen ⟨*h*⟩ nebo ⟨*v*⟩, kterým se udává, zda se budou dále kreslené objekty zrcadlit kolem horizontální nebo vertikální osy. Při zadání jiného písmene se nic neudělá. Pokud chceme zrcadlení podle obou os (středové), vyvolá se toto makro dvakrát, pokaždé s jinou indikací osy, například:

```
\setscales(1,0,0,2.1)%
\XYaxes(-13,13){}(-13,13){}
\scaleput(12,12){\arc(-10,0){90}}% 1.kvadr.
\mirrorscale{h}
\scaleput(12,12){\arc(-10,0){90}}% 4.kvadr.
\mirrorscale{v}
\scaleput(12,12){\arc(-10,0){90}}% 3.kvadr.
\mirrorscale{h}
\scaleput(12,12){\arc(-10,0){90}}% 2.kvadr.
```



`\storescales⟨cmd⟩` **Uložení všech měřítek do povelu**

Hodnoty prvků matice měřítek oddělené čárkami se uloží do zvoleného povelu.

Příklad: `\storescales\scales`

`\restorescales⟨cmd⟩` **Obnovení všech měřítek z povelu**

Matice měřítek se obnoví z hodnot jejích prvků uložených dříve do zvoleného povelu příkazem `\storescales`.

Příklad: `\restorescales\scales`

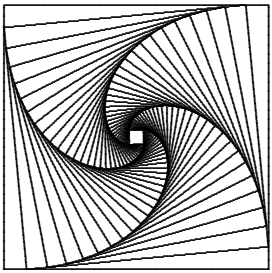
`\scalerot{⟨úhel⟩}` **Nastavení měřítek pro rotaci o daný úhel**

Z daného argumentu $\langle \textit{úhel} \rangle = \alpha$ ve stupních se vypočtou hodnoty funkcí $\sin(\alpha)$ a $\cos(\alpha)$ a z nich se sestaví matice rotace *objektu*

$$\begin{bmatrix} \cos(\alpha)\ , & -\sin(\alpha) \\ \sin(\alpha)\ , & \cos(\alpha) \end{bmatrix},$$

jíž se vynásobí zleva matice aktuálních měřítek. To zajistí správnou rotaci i již nějak oměřítkovaných objektů, které se budou kreslit. Je třeba upozornit na obrácená znaménka u sinů proti znaménkům platným pro rotaci os. V tomto příspěvku se dodržuje standardní měření kladného úhlu natáčení *objektů* proti směru otáčení hodinových ručiček.

Příklad:



```
\FPset\A{35} \FPmul\Ah\A{.5}% a, a/2
\FPset\N{35}%          počet čtverců
\FPset\da{3}%              odskočení
\FPsub\Am\A\da \FPatan\alf\Am\da%
\FPupn\scl{alf cos A * Am /}%
\FPrtod\alf\alf%
\FPset\da\alf%           delta úhlu
\setscales(1,0,0,1)
\unitlength=1mm
\begin{picture}(30,35)(-15,-15)
 \For\C=(1:1:\N)%              Cyklus čtverců
 {\polyline(-\Ah,-\Ah, -\Ah,\Ah, \Ah,\Ah, \Ah,-\Ah, -\Ah,-\Ah)%
  \scalerot{\alf}%             otočí o alfa stupňů
  \unitlength=\scl\unitlength% zmenší stranu čtverce
 }%                            konec cyklu čtverců
\end{picture}
```

Kreslení grafických objektů

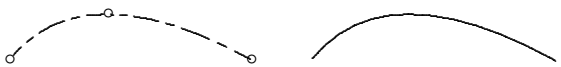
Již při popisu struktury čar se upozorňovalo na relativní složitost zadávání kresby čar. Kromě toho snadno poznáme, že některé příkazy pro vynášení objektů chybějí. Proto se přistoupilo k realizaci následujících maker:

`\curves{`⟨*symbol*⟩`}{vzor}(`⟨*seznam souřadnic*⟩`)` **Kreslení křivky**

Příkaz sdružuje všechny potřebné informace v jednom volání, ale zásady uvedené výše sám dodržuje (viz str. 15). Tím podstatně zjednodušuje složitější zadání pro kreslení křivek.

Příklad:

```
\put(0,0){\curves{\circle{1}}{2,1,1,1,2}(0,0, 13,6, 32,0)}
\put(40,0){\curves{}{}(0,0, 13,6, 32,0)}
```



`\polyline[`⟨*počet*⟩`](`⟨*seznam souřadnic*⟩`)` **Lomené přímky**

Existují případy, kdy je účelné pospojovat dané body přímkami. Vznikne tak po částech přímková čára. Parametr ⟨*počet*⟩ má stejný význam jako u příkazu `\curve` (viz str. 14).

Příklad:

```
\Cmd \xy={-9.51,3.09, 9.51,3.09, -5.88,-8.09,
    0,10, 5.88,-8.09, -9.51,3.09}
\put(0,-5){\polyline(\xy)}
\curvesymbol{\phantom{\circle*{.75}}\circle*{.75}}
\put(0,-5){\polyline[-8](\xy)}
```

`\polylineq[`⟨*počet*⟩`](`⟨*seznam y-souř.*⟩`)`

Lomené přímky s ekvidistantním x

Příkaz se s výhodou použije při rovnoměrně rostoucí souřadnici x. Potom není nutné tuto souřadnici vypisovat v seznamu, ale lze ji generovat jako index a skutečný rozměr zajistit měřítkem.

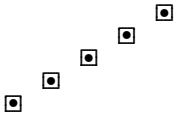
```
\setscales(2,0,0,1)%
\scaleput(0,0){\polylineq(1,3,-4,6,8,0,-5,2,-3,8,-10,-1,1)}
```

`\mscput(⟨x⟩,⟨y⟩)(⟨dx⟩,⟨dy⟩){⟨počet⟩}{⟨objekt⟩}` **Multi scaleput**

Jde o variantu standardního příkazu `\multiput`, avšak s možností měřítkování jako u `\scaleput`.

Příklad:

```
\mscput(0,0)(5,3){5}{\Square(2)}
\mscput(1,1)(5,3){5}{\circle*{1}}
```

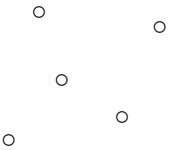


`\mscputxy(⟨seznam⟩){⟨objekt⟩}` **Obecný multi scaleput**

Tento příkaz na rozdíl od `\mscput`, který umisťuje objekty rovnoměrně, umožňuje rozprostírat objekty na ploše libovolně. V seznamu jsou reálné souřadnice oddělené čárkami po dvojicích `x,y`.

Příklad:

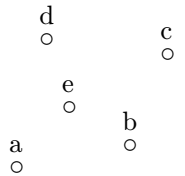
```
\mscputxy(0,0, 15,3, 20,15, 4,17, 7,8)
        {\circle{1.5}}
```



`\mscputtext(⟨seznam souřadnic a textů⟩)` **Obecný multi text**

Pro usnadnění umisťování textů do obrázků byl sestaven tento příkaz, v němž ⟨*seznam souřadnic a textů*⟩ obsahuje položky navzájem oddělené čárkami, vytvářející trojice, v nichž první dvě položky jsou souřadnice levého dolního rohu boxu s textem, který je třetí položkou ve skupině. Jako příklad uveďme předešlý příkaz s popisem:

```
\mscputxy(0,0, 15,3, 20,15, 4,17, 7,8)
        { \circle1.5}
\scaleput(-1,2){\mscputtext(0,0,a, 15,3,b,
        20,15,c, 4,17,d, 7,8,e)}
```

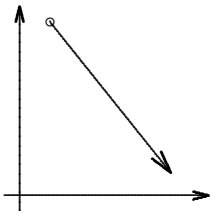


`\arrow(⟨x⟩,⟨y⟩){⟨len⟩}` **Orientovaná úsečka**

Nakreslí úsečku počínající v bodě daném umisťovacím příkazem a končící v bodě jehož relativní souřadnice jsou prvními dvěma argumenty příkazu. Třetí argument ⟨*len*⟩ udává délku koncové šipky v jednotkách `\unitlength`.

Příklad:

```
\put(4,23){\circle{1}}
\scaleput(4,23){\arrow(16,-20){3}}
```

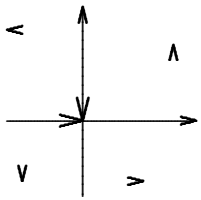


`\harrow{⟨dist⟩}{⟨len⟩}` **Horizontální šipka**
`\varrow{⟨dist⟩}{⟨len⟩}` **Vertikální šipka**

Oba příkazy vykreslují koncovou šipku (špičku) o délce $|\langle len \rangle|$ s vrcholem v bodě vzdáleném od bodu daného umisťovacím příkazem o $\langle dist \rangle$ a orientovanou podle znaménka $\langle len \rangle$ horizontálně nebo vertikálně.

Příklad:

```
\put(0,0){\harrow{0}{3}}
\put(0,0){\varrow{0}{-3}}
\put(12,10){\varrow{0}{2}}
\put(-10,12){\harrow{0}{2}}
\put(-8,-8){\varrow{0}{-2}}
\put(8,-8){\harrow{0}{2}}
```

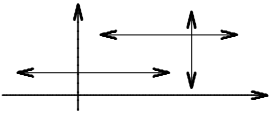


`\hdim{⟨dist⟩}{⟨len⟩}` **Horizontální kóta**
`\vdim{⟨dist⟩}{⟨len⟩}` **Vertikální kóta**

Příkazy vytvářejí horizontální a vertikální kótovací čáry. Argument $\langle dist \rangle$ udává délku kótovací čáry mezi šipkami a $\langle len \rangle$ délku křidélek jejích koncových špiček. Referenčními body jsou levý u horizontální a dolní u vertikální kóty.

Příklad:

```
\put(3,8){\hdim{18}{2}}
\put(-8,3){\hdim{20}{2}}
\put(15,1){\vdim{10}{2}}
```

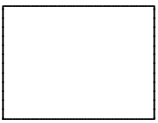


`\Rectan(⟨arg1⟩,⟨arg2⟩)` **Obdélník**

Makro vykreslí obdélník o straně $\langle arg_1 \rangle$ ve směru aktuální osy x a straně $\langle arg_2 \rangle$ ve směru aktuální osy y. Oba argumenty jsou v případně oměřítkovaných jednotkách `\unitlength`.

Příklad:

```
\unitlength=1mm
\begin{picture}(30,15)(0,0)
  \put(0,0){\Rectan(20,15)}
\end{picture}
```



`\Square(⟨arg⟩)` **Čtverec**

Užije se pro nakreslení čtverce o straně $\langle arg \rangle$ jednotek `\unitlength` se všemi tranformacemi podobně jako u obdélníka.

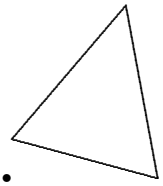
`\Triang(⟨arg1⟩,⟨arg2⟩,⟨arg3⟩,⟨arg4⟩)` **Trojúhelník**

Slouží ke kreslení trojúhelníka s vrcholy v bodech $(0,0)$, $(\langle arg_1 \rangle,\langle arg_2 \rangle)$

a (⟨*arg*$_3$⟩,⟨*arg*$_4$⟩) vzhledem k bodu definovanému libovolným umisťovacím příkazem.

Příklad:

```
\begin{picture}(30,30)(0,0)
  \setscales(1,0,0,1.05)%
  \scalerot{-15}%
  \put(0,0){\circle*{1}}%          počátek
  \put(20,0){\Triang(-20,0,-10,20)}%
\end{picture}
```



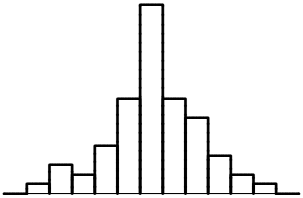
`\barq(`⟨*seznam*⟩`)` **Vykreslení sloupkového diagramu**

Toto makro se použije, je-li přírůstek nezávisle proměnné na horizontální ose rovnoměrný. Jako nejběžnější případ uveďme histogram se stejně širokými třídami na vodorovné ose a četnostmi na ose pořadnic. Argumentem ⟨*seznam*⟩ v kulatých závorkách je seznam obsahující výšky jednotlivých sloupků, např. relativní četnosti. Měřítka zobrazení se řídí pomocí příkazu `\setscales`.

Příklad:

Má se zobrazit histogram o četnostech $n_i =$ 0, 1, 3, 2, 5, 10, 20, 12, 8, 4, 2, 1, 0

```
\unitlength1mm
\begin{picture}(0,0)(-58,-15)
   \put(0,0){\line(1,0){39}}
   \scaleput(0,0)%
    {\setscales(3,0,0,1.5)%
     \linethickness{.8pt}%
     \barq(0,1,3,2,5,10,20,10,8,4,2,1,0)%
    }%
\end{picture}
```



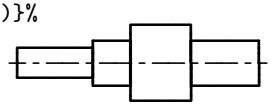
`\barxy(`⟨*seznam*⟩`)` **Sloupkový diagram s nestejnými třídami**

Makro pracuje podobně jako předcházející s tím rozdílem, že ⟨*seznam*⟩ v kulatých závorkách je složen z *dvojic* reálných čísel, souřadnic levých rohů obdélníků – sloupků.

Příklad:

Nakresleme náčrt hřídele:

```
\put(0,0){\curves{}{2,1,1,1,2}(-1,0, 33,0)}%
\thicklines%
\For \yscale=(1,-1)%
 {\barxy(0,2, 10,3, 15,5, 23,3, 32,0)}%
```



Jiným typem diagramu, užívaným zejména pro názorné zobrazování poměrů dílčích jevů k celku, je kruhový diagram označovaný hovorově jako „koláč“. Celek tvoří kruh rozdělený na výseče, každá s plochou úměrnou zastoupení elementárního jevu v celku. Tyto diagramy jsou velmi populární pro svoji názornost, a proto jsou často užívány k zobrazování výsledků průzkumů a pod. K tomuto účelu je k dispozici makro

`\pie{`⟨*dia*⟩`}(`⟨*seznam*⟩`)` **Kruhový diagram**

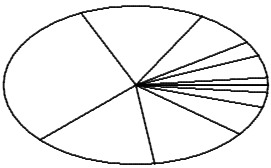
Parametry makra jsou:

⟨*dia*⟩ je reálný průměr kružnice ohraničující diagram jako násobek `\unitlength`

⟨*seznam*⟩ je seznam reálných položek oddělených čárkami, představujících v součtu celek, který obsadí plochu celého kruhu. Poměr hodnoty položky k součtu hodnot všech položek se vyjádří stejným podílem úhlu kruhové výseče k plnému úhlu.

Příklad:

```
\setscales(1,0,0,.6)
\scaleput(0,0)%
  {\pie{35}(1,3,2,5,10,20,10,8,4,2,1)}
```



V příkladu jsou zobrazeny poměry odpovídající stejným datům, jaké byly užity pro ukázku kreslení histogramu:

Kreslení diagramů

Vynášení diagramů patří k velmi častým úlohám v praxi. I když lze diagram zpracovat v jiném programu a následně ho přenést do výsledného dokumentu, jsou případy (kromě již výše zmíněných), kdy je potřebné, aby byl diagram vytvářen současně s dokumentem. Jde zejména o prezentace, při nichž chceme generovat diagram postupně, např. při vyznačování měřených bodů, jejich proložení regresní funkcí, vyznačení pásem spolehlivosti, vložení doprovodných textů a pod.

K základním činnostem při vynášení diagramů patří nakreslení os. Za tím účelem obsahuje `\diagram.sty` řadu příkazů k vykreslování os, sítí a grafických papírů.

Jednoduché osy diagramu

`\Xaxis(`⟨*xbeg*⟩`,`⟨*xfin*⟩`)(`⟨*xtxt*⟩`,`⟨*ytxt*⟩`){`⟨*text*⟩`}` **Vodorovná osa**
Makro slouží pro nakreslení osy `x` o zadané délce a popisu v požadovaném místě. Parametry makra jsou:

`(`⟨*xbeg*⟩`,`⟨*xfin*⟩`)`	reálné souřadnice začátku a konce osy x na kreslící ploše vymezené prostředím `picture`. Tyto souřadnice také určují orientaci osy, protože v bodě definovaném ⟨*xfin*⟩ bude nakreslena horizontální šipka;
`(`⟨*xtxt*⟩`,`⟨*ytxt*⟩`)`	reálné souřadnice referenčního bodu textu – popisu osy x;
⟨*text*⟩	libovolný popis osy s referenčním bodem v levém dolním rohu prvního znaku textu.

Příklad:

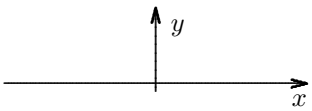
```
\unitlength1mm
\begin{picture}(0,0)(-75,-19)
\put(0,0){\circle*{1}}
\put(0,0){\Xaxis(20,-20)(-26,-5){$hloubka$}}
\end{picture}
```

`\Yaxis(`⟨*ybeg*⟩`,`⟨*yfin*⟩`)(`⟨*xtxt*⟩`,`⟨*ytxt*⟩`){`⟨*text*⟩`}` **Svislá osa**
Příkazem se vykreslí osa y o zadané délce a popisu v požadovaném místě. Parametry makra jsou:

`(`⟨*ybeg*⟩`,`⟨*yfin*⟩`)`	reálné souřadnice začátku a konce osy y na kreslící ploše vymezené prostředím `picture`. Tyto souřadnice také určují orientaci osy, protože v bodě definovaném ⟨*yfin*⟩ bude nakreslena vertikální šipka;
`(`⟨*xtxt*⟩`,`⟨*ytxt*⟩`)`	reálné souřadnice referenčního bodu textu – popisu osy y;
⟨*text*⟩	libovolný popis osy s referenčním bodem v levém dolním rohu prvního znaku textu.

Příklad:

```
\put(0,0){\Xaxis(-20,20)(18,-3){$x$}}
\put(0,0){\Yaxis(-1,15)(2,12){$y$}}
```



Z příkladu je patrno, že dvojicí příkazů `\Xaxis` a `\Yaxis` lze vytvořit libovolnou soustavu os, s libovolným a libovolně umístěným popisem. Mnohdy je tato volnost přepychem, a proto bylo sestaveno makro `\XYaxes`, které za cenu jistých omezení nevyžaduje na uživateli tolik dat:

`\XYaxes(`⟨*xbeg*⟩`,`⟨*xfin*⟩`){`⟨*xlab*⟩`}(`⟨*ybeg*⟩`,`⟨*yfin*⟩`){`⟨*ylab*⟩`}`

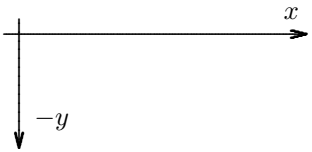
Vynesení osového kříže

Makro vynese a popíše osy x a y. Popisy budou umístěny u konců os a s ohledem na vyhrazené místo by měly být krátké. Parametry makra jsou:

`(`⟨*xbeg*⟩`,`⟨*xfin*⟩`)`	reálné souřadnice začátku a konce osy x na kreslící ploše vymezené prostředím `picture`. Tyto souřadnice také určují orientaci osy, protože v bodě definovaném ⟨*xfin*⟩ bude nakreslena horizontální šipka;
⟨*xlab*⟩	text pro popis osy `x`;
`(`⟨*ybeg*⟩`,`⟨*yfin*⟩`)`	reálné souřadnice začátku a konce osy y na kreslící ploše vymezené prostředím `picture`. Tyto souřadnice také určují orientaci osy, protože v bodě definovaném ⟨*yfin*⟩ bude nakreslena vertikální šipka;
⟨*ylab*⟩	text pro popis osy y.

Příklad:

```
\unitlength1mm
\begin{picture}(0,0)(-58,-15)
  \XYaxes(-2,38){$x$}(2,-15){$-y$}
\end{picture}
```



Stupnice a osnovy

Při kreslení diagramů není nutné jen nakreslit osy, ale i je označit stupnicemi. Zatímco nakreslení lineární stupnice není žádným problémem, protože máme k dispozici příkazy `\multiput`, příp. `\mscput`, je vynášení nelineárních stupnic, případně celé osnovy, úloha podstatně složitější.

Nejobvyklejší nelineární stupnicí je stupnice logaritmická. Pro ni jsou připravena následující dvě makra:

`\logx(`⟨*seznam*⟩`){`⟨*n*⟩`}{`⟨*délka*⟩`}{`⟨*tlouštka*⟩`}`

Logaritmická stupnice na vodorovné ose

Parametry tohoto příkazu jsou:

⟨*seznam*⟩ je seznamem reálných kladných hodnot z jednoho intervalu (dekády), pro které bude vykreslena čárka logaritmického dělení osy `x`;

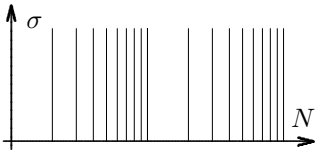
⟨*n*⟩ je celočíselný počet opakování N (např. dekád), které se mají vykreslit;

⟨*délka*⟩ je reálné číslo udávající v násobcích `\unitlength` délku čar dělení osy `x`;

⟨*tlouštka*⟩ je reálná délka (s rozměrem!) udávající tloušťku čar

Příklad:

```
\unitlength1mm
\begin{picture}(0,0)(-75,-10)
\XYaxes(-1,40){$N$}(-1,18){$\sigma$}
\setscales(18,0,0,1)
\scaleput(0,0)%
    {\logx(1,2,3,4,5,6,7,8,9,10){2}{15}{.2pt}}
\end{picture}
```



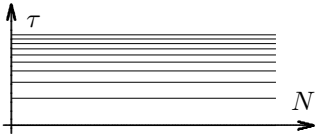
`\logy(`⟨*seznam*⟩`){`⟨*n*⟩`}{`⟨*délka*⟩`}{`⟨*tlouštka*⟩`}`

Logaritmická stupnice na svislé ose

Parametry tohoto makra jsou stejné jako u `\logx` jen s tím rozdílem, že se týkají vertikální osy y.

Příklad:

```
\unitlength1mm
\begin{picture}(0,0)(-60,-7)
 \XYaxes(-1,40){$N$}(-1,15){$\tau$}
 \scaleput(0,0)%
    {\logy(1,2,3,4,5,6,7,8,9,10){1}{35}{.2pt}}
\end{picture}
```

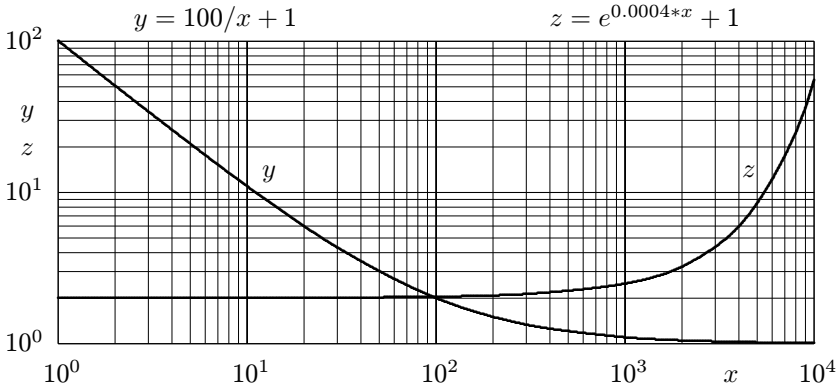


Kombinací obou příkazů lze získat grafický papír „log-log“, případně použitím jednoho z těchto příkazů a lineární stupnice papír „semilog“.

Účelem dále uvedeného příkladu je ukázat nejen použití maker `\logx` a `\logy`, ale i možnosti až dosud vysvětlených maker v sestavě s aritmetikou a funkcemi (makro `\Appitem` je popsáno na straně 51).

Příklad: Vykresli funkce $y = 100/x + 1$ a $z = \mathrm{e}^{0.0004*x} + 1$:

```
\Cmd \xy={}
\Cmd \xz={}
\For \x=(1,3,10,30,100,300,1000,2000,4000,7000,10000)%
{\FPupn \y{x 100 / 1 + log}%
 \FPupn \z{x .0004 * e ^ 1 + log}%
 \FPlog \x\x%
 \Appitem\xy+{\x,\y}
 \Appitem\xz+{\x,\z}
}%
\Cmd \c={1,2,3,4,5,6,7,8,9,10}
\Cmd \dx={25}\Cmd \dy={20}%
\unitlength1mm
\begin{picture}(100,55)(-3,-6)
 \setscales(\dx,0,0,1)
 \scaleput(0,0){\logx(\c){4}{40}{.2pt}}
 \For \x=(0:1:4)%   popis dekád y
   {\FPclip\x\x \put(-2,-5){\scaleput(\x,0){10$^\x$}}}
 \setscales(1,0,0,\dy)
 \scaleput(0,0){\logy(\c){2}{100}{.2pt}}
 \For \x=(0:1:2)%   popis dekád x
   {\FPclip\x\x \put(-7,-1){\scaleput(0,\x){10$^\x$}}}
 \setscales(\dx,0,0,\dy)
 \thinlines
 \multiput(0,0)(\dx,0){5}{\line(0,1){40}}
 \multiput(0,0)(0,\dy){3}{\line(1,0){100}}
 \thicklines
 \scaleput(0,0){\curve(\xy)}
 \scaleput(0,0){\curve(\xz)}
 \setscales(1,0,0,1)
 \mscputtext(88,-5,$x$, -5,30,$y$, -5,25,$z$, 10,42,$y=100/x+1$,
   65,42,$z={\rm e}^{0.0004*x}+1$, 90.5,22,$z$, 27,22,$y$)
\end{picture}
```



Výpočty a vynášení funkcí

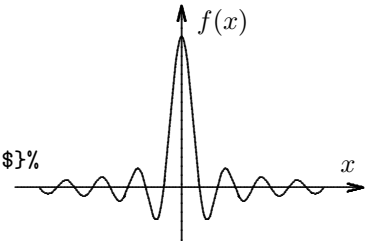
Velmi často je zapotřebí vynášet průběhy funkcí. To se zajišťuje výpočtem jejich hodnot pro serii hodnot argumentů a potom pospojováním takto vzniklých bodů buď hladkou křivkou, nebo čarou po částech lineární. První variantu zajistí příkaz `\curve`, druhou pak `\polyline`. Pro usnadnění práce uživatele byla vypracována makra zajišťující sestavení seznamu dvojic souřadnic funkcí a případně i náhodných procesů:

`\Funx` ⟨*flist*⟩`={`⟨*fname*⟩`}(`⟨*xlist*⟩`)` **Výpočet hodnot funkce $f(x)$**
Makro vypočte hodnoty funkce ⟨*fname*⟩ argumentu nabývajícího hodnot ze seznamu ⟨*xlist*⟩ a uloží je do ⟨*flist*⟩. Parametr ⟨*flist*⟩ má tvar příkazu. Seznam ⟨*xlist*⟩ může být příkazem nebo přímo seznamem reálných hodnot nezávisle proměnné oddělených navzájem čárkami. Uživatel musí sestavit předpis pro výpočet žádané funkce a tu definovat jako ⟨*fname*⟩ bez úvodního zpětného lomítka.

Příklad:

Vynést graf difrakční funkce:

```
\Cmd\fxy={}%
\Funx\fxy={FPdiff}(-25:1:25)%
\unitlength=1mm%
\begin{picture}(100,30)(-40,-5)%
 \XYaxes(-22,24){$x$}(-7,24){$f(x)$}%
 \setscales(.75,0,0,20)%
 \scaleput(0,0){\curve(\fxy)}%
\end{picture}
```

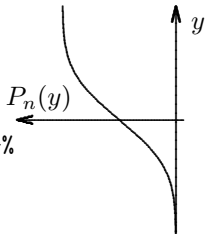


`\Funy` ⟨*flist*⟩`={`⟨*fname*⟩`}(`⟨*ylist*⟩`)` **Výpočet hodnot funkce $f(y)$**
Činnost makra je analogická funkci `\Funx`, jen v seznamu argumentů jsou hodnoty pořadnic. Tato skutečnost je ošetřena i v pořadí dvojic souřadnic v ⟨*flist*⟩, takže vynášení funkce bude bez komplikací.

Příklad:

Vertikální diagram normální distribuční funkce:

```
\Cmd\fxy={}%
\Funy\fxy={FPPnx}(-3:.2:3)%
\unitlength=1mm%
\begin{picture}(0,38)(-85,-19)%
 \XYaxes(1,-21){\hskip-3ex$P_n(y)$}(-15,15){$y$}%
 \setscales(-15,0,0,5)%
 \scaleput(0,0){\curve(\fxy)}%
\end{picture}
```



`\RPGini(`⟨*seed*⟩`,`⟨*alfa*⟩`,`⟨*start*⟩`)`

Iniciace generátoru náhodného procesu

Makrem se zavede „násada“ generátoru pseudonáhodných čísel s normálním rozložením z parametru ⟨*seed*⟩, jímž může být jak reálné číslo, tak i příkaz obsahující reálné číslo. Dále se nastaví koeficient α pro generování procesu z parametru ⟨*alfa*⟩, z něj se vypočte koeficient $\beta = 1 - \alpha$ (viz dále) a nakonec se nastaví počáteční hodnota generátoru ze ⟨*start*⟩.

`\RPGval`⟨*cmd*⟩ **Vzorek náhodného procesu**

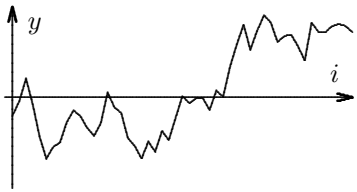
Vyvolání tohoto příkazu s hodnotou p_{i-1} vzorku náhodného procesu z předešlého kroku v povelu ⟨*cmd*⟩ zajistí výpočet dalšího normálního pseudonáhodného čísla, řekněme r_i, a z něj pak hodnotu vzorku p_i pseudonáhodného procesu pomocí vztahu $p_i = \alpha\, p_{i-1} + \beta\, r_i$, kterou vrátí do ⟨*cmd*⟩. Proces má exponenciální autokorelační funkci. Čím je větší α, tím pomaleji tato funkce klesá.

`\RPG`⟨*cmd*⟩⟨*arg*⟩ **Generování náhodného procesu**

Celou realizaci náhodného procesu o ⟨*arg*⟩ vzorcích lze získat voláním tohoto makra. Je třeba jen zadat povel ⟨*cmd*⟩, do kterého se uloží souřadnice bodů vzorků. To vše ovšem až po iniciaci generátoru voláním `\RPGini` s požadovanými parametry.

Příklad:

```
\RPGini(12345,.975,0)%
\Cmd\fxy={}%
\RPG\fxy{50}%
\begin{picture}(0,25)(-55,-15)%
 \XYaxes(-1,45){$i$}(-12,12){$y$}%
 \setscales(.9,0,0,75)%
 \polyline(\fxy)%
\end{picture}
```



Diagramy s obecnými osami

Diagramy samy nemusí být kresleny pouze v jednoduchých osách, ale často se vynášejí do speciálních grafických papírů. Jeden takový složitější případ byl ukázán na straně 28, kde se vynášely funkce do diagramu s oběma osami logaritmickými. Mnohdy se u komerčních programů setkáváme s diagramy, které byly sestrojeny s přednastavenými mezemi os, takže využití plochy diagramu je špatné. To má pak za následek

malou vypovídací schopnost diagramu. Velice často k tomu dochází u diagramů s logaritmickými osami. Aby se tomu mohlo předejít, byla sestavena makra, jimiž lze sestrojit dosti obecné diagramy a to postupně, což ocení zejména přednášející, kteří mohou při prezentacích vytvářet diagramy v souběhu s předávanou informací.

Výhodou dále uvedených maker je jejich obecnost umožňující uživateli nakreslit skoro jakýkoliv diagram. Na druhé straně ovšem tato obecnost přinesla komplikace ve složitosti zadání, i přes maximální snahu usnadnit uživatelovi práci. Z uživatelova pohledu jsou pro nakreslení diagramu důležitá zejména dvě makra - `\Dgrid` a `\Drawfun`. Prvním se nadefinuje kreslící plocha a transformace souřadných os (stupnice). Druhé makro potom slouží ke kreslení bodů a lomených a hladkých čar. Uživatel navíc může ovlivňovat písmo a velikost popisu os a doplňovat diagram texty a výsledky výpočtů.

Aby nedošlo k vzájemným posunům vynášených součástí diagramu v po sobě jdoucích etapách je naprosto nezbytné zapsat celé zadání grafu jako jeden blok s řádky zakončenými znaky procento (%)!

`\Dgrid{`⟨*ul*⟩`}%` **Pracovní plocha diagramu**
`(`⟨*lenx*⟩`,`⟨*leny*⟩`)(`⟨*dx,dy*⟩`)%`
`(`⟨*funx*⟩`,`⟨*textx*⟩`,`⟨*xtx*⟩`,`⟨*ytx*⟩`,`⟨*ylo*⟩`,`⟨*yhi*⟩`)%`
`(`⟨*xgrid*⟩`)(`⟨*xlabs*⟩`)%`
`(`⟨*funy*⟩`,`⟨*texty*⟩`,`⟨*yty*⟩`,`⟨*xty*⟩`,`⟨*xlo*⟩`,`⟨*xhi*⟩`)%`
`(`⟨*ygrid*⟩`)(`⟨*ylabs*⟩`)%`

Struktura popisu pracovní plochy diagramu je tedy následující:

⟨*ul*⟩ velikost `\unitlength`

Popis pracovní plochy:

`(`⟨*lenx*⟩`,`⟨*leny*⟩`)` šířka a výška aktivní plochy diagramu

`(`⟨*dx,dy*⟩`)` odsazení levého dolního rohu aktivní plochy od referenčního bodu

Parametry vodorovné osy (x):

`(`⟨*funx*⟩`,` jméno funkce, která se použije pro transformaci hodnot nezávisle proměnné x (*bez* úvodního zpětného lomítka). Pro lineární stupnici na ose x se předepíše funkce pouhého přiřazení – `FPset`. Lze použít libovolnou z výše popsaných funkcí z balíků `fp`, `fp-contrib`, anebo i uživatelovu funkci o struktuře ⟨*jméno funkce*⟩⟨*cmd*⟩⟨*arg*⟩.

⟨*textx*⟩, označení vodorovné osy

⟨*xtx*⟩, souřadnice x, ve keré bude vyneseno označení ⟨*textx*⟩

⟨*ytx*⟩, vertikální pozice označení v jednotkách `\unitlength`

⟨*ylo*⟩,⟨*yhi*⟩) počátek a konec úseček kolmých na osu x. Pokud ⟨*yhi*⟩=⟨*ylen*⟩, vyplní se x-ová část mřížky v pracovní ploše

(⟨*seznam x*⟩) seznam hodnot argumentu, v nichž budou vyneseny svislice od ⟨*ylo*⟩ do ⟨*yhi*⟩

(⟨*hladiny x*⟩) výběr hodnot z ⟨*seznam x*⟩, které budou popsány pod osu x

Parametry svislé osy (y):

(⟨*funy,*⟩ jméno funkce, která se použije pro transformaci hodnot závisle proměnné y (*bez* úvodního zpětného lomítka)

⟨*texty*⟩, označení svislé osy

⟨*yty*⟩, souřadnice y, ve které bude vyneseno označení ⟨*texty*⟩

⟨*xty*⟩, horizontální pozice označení v jednotkách `\unitlength`

⟨*xlo*⟩,⟨*xhi*⟩) počátek a konec úseček kolmých na osu y. Pokud ⟨*xhi*⟩=⟨*xlen*⟩, vyplní se y-ová část mřížky v pracovní ploše

(⟨*seznam y*⟩) seznam hodnot pořadnic, v nichž budou vyneseny horizontály od ⟨*xlo*⟩ do ⟨*xhi*⟩

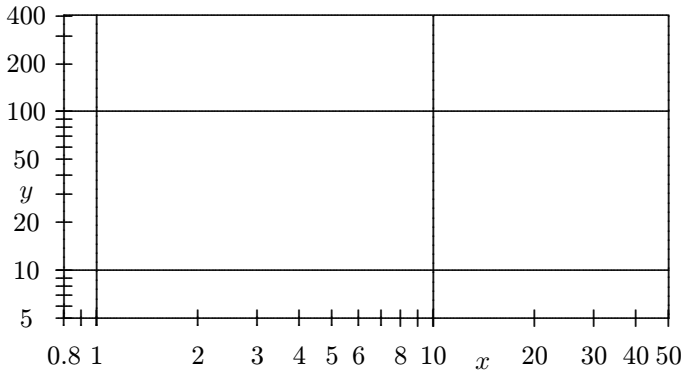
(⟨*hladiny y*⟩) výběr hodnot z ⟨*seznam y*⟩, které budou popsány vedle osy y

Kromě prvního parametru – délkové jednotky diagramu – jsou všechny skupiny parametrů uzavřeny v *kulatých* závorkách. Pro správnou funkci makra je nesmírně důležité dodržení formálních požadavků na parametry.

Vytváření pracovní plochy se realizuje zvlášť pro osu x pomocí makra `\Dxaxis` a zvlášť pro osu y za použití makra `\Dyaxis`. Tato makra využívají vždy příslušnou část parametrů makra `Dgrid`. Pro zvýrazňování vybraných čar mřížky lze ještě použít pomocné makro `\Daxmod`. Jako ukázku uveďme konstrukci plochy s logaritmickými osami se zvýrazněním mezí dekád:

Příklad:

```
\Dgrid%                                          Definice pracovní plochy
  {1mm}%                                                 unit length
  (80,40)(-10,-8)%                                       parametry plochy
  (FPlog,$x$,14,-5,-1,1)%                                parametry osy x
  (0.8,0.9,1,2,3,4,5,6,7,8,9,10,20,30,40,50)%            škála x
  (0.8,1,2,3,4,5,6,8,10,20,30,40,50)%                    popis x
  (FPlog,$y$,35,-5,-1,1)%                                parametry osy y
  (5,6,7,8,9,10,20,30,40,50,60,70,80,90,100,200,300,400)% škála y
  (5,10,20,50,100,200,400)%                              popis y
\thicklines
\Daxmod%                                         Modifikace osnovy
  (-1,\ylen)%                                            (y_od,y_do)
  (1,10)%                                                x
  (-1,\xlen)%                                            (x_od,x_do)
  (10,100)%                                              y
```



V předcházejícím příkladu se použilo makro `\Daxmod`. Vyvoláním tohoto makra zajistíme doplnění původního zadání pracovní plochy.

`\Daxmod(`⟨*ylo*⟩`,`⟨*yhi*⟩`)(`⟨*xlist*⟩`)(`⟨*xlo*⟩`,`⟨*xhi*⟩`)(`⟨*ylist*⟩`)%`

Modifikace pracovní plochy

Půjde obvykle o změnu tloušťky nebo délky některých čar osnovy. Jeho parametry jsou:

(⟨*ylo*⟩,⟨*yhi*⟩) dolní a horní vzdálenost konců úseček od osy x v `\unitlength`, rovnoběžných s osou y

⟨*xlist*⟩ seznam souřadnic x těchto úseček

(⟨*xlo*⟩,⟨*xhi*⟩) levá a pravá vzdálenost konců úseček od osy y v `\unitlength`, rovnoběžných s osou x

⟨*ylist*⟩ seznam souřadnic y těchto úseček

`\Drawfun{`⟨*objekt*⟩`}{`⟨*funx*⟩`}{`⟨*funy*⟩`}(`⟨*body*⟩`){`⟨*symbol*⟩`}{`⟨*vzor*⟩`}`

Kreslení objektu

Tímto makrem lze vykreslovat body, lomené čáry i křivky podle sestavy parametrů. Těmi jsou:

⟨*objekt*⟩ pro ⟨*objekt*⟩=`polyline` se vykreslí lomená (po částech přímková) čára,
pro ⟨*objekt*⟩= `curve` se vykreslí hladká křivka

⟨*funx*⟩ jméno funkce pro transformaci souřadnic osy x

⟨*funy*⟩ jméno funkce pro transformaci souřadnic osy y

(⟨*body*⟩) seznam souřadnic bodů $x_1, y_1,\ x_2, y_2, \ldots$

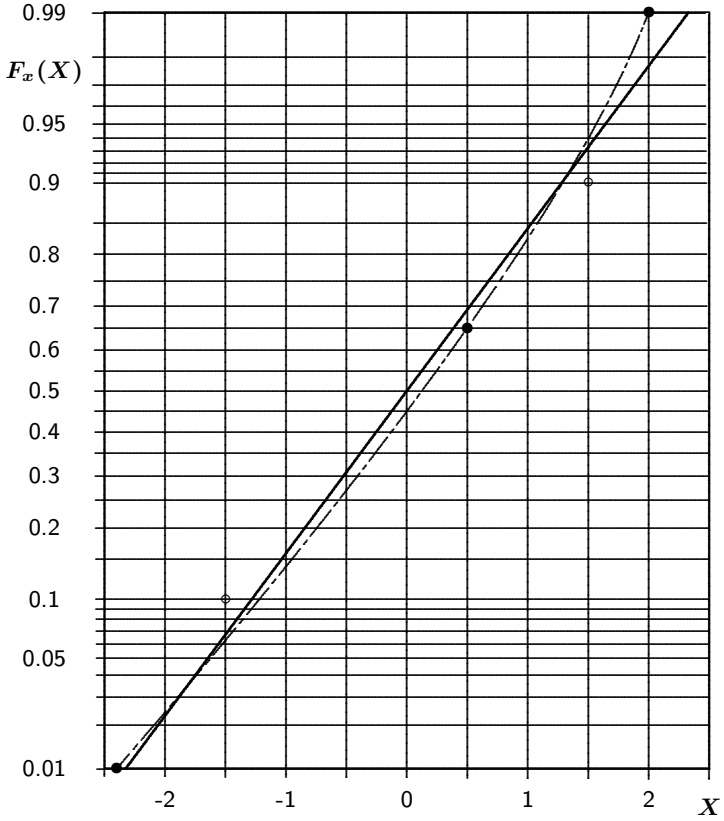
⟨*symbol*⟩ libovolný symbol pro vykreslování bodů

⟨*vzor*⟩ vzor přerušování čáry = 0 pro plnou čáru,
jiný – viz `\curvedashes`, str. 15.

Příklad:

Vykreslit pravděpodobnostní papír pro normální rozložení a vynést silnou čarou distribuční funkci, slabě čerchovaně hladkou křivku spojující 3 body a nakonec dva volné body.

```
\boldmath \small \sffamily%
\Dgrid{1mm}%
(80,100)(-10,-8)%
(FPset,$X$,2.5,-4,-1,100)%                    osa x
(-2.5:.5:2.5)%                                mřížka x
(-2:1:2)%                                     popis x
(FPxPn,$F_x(X)$,.975,-8,-1,80)%               osa y
(.01:0.01:0.09,.1:.05:.9,0.91:.01:.99)%       mřížka y
(0.01,0.05,0.1,0.2:0.1:0.8,0.9,0.95,0.99)%    popis y
\thicklines%
\Drawfun{polyline}{FPset}{FPxPn}(-2.327,0.01,2.327,0.99){}{0}%
\thinlines%
\Drawfun{curve}{FPset}{FPxPn}(-2.4,0.01,0.5,0.65,2,0.99)%
  {$\bullet$}{0,2,1,.5,1,2}%
\Drawfun{curve}{FPset}{FPxPn}(-1.5,.1, 1.5,.9)%
  {\phantom{\circle{1}}\circle{1}}{}%
```



`\Dtext(`⟨*seznam*⟩`)` **Vkládání textů**

Úkolem makra je vložení libovolných popisů do plochy diagramu. Parametr ⟨*seznam*⟩ je množina položek oddělených čárkami, která obsahuje trojice údajů, z nichž první dva udávají neoměřítkované souřadnice v `\unitlength` začátku textu a třetí vlastní text. Textů může být jedním voláním zapsáno více.

Několik poznámek k použití maker `D...`

Obecnost zadání vzhledu diagramů je zaplacena většími nároky na uživatele. Ten *musí* dodržet formální požadavky na strukturu dat bez vý-

jimek, protože jinak dojde k chybovému hlášení, které, jak je obvyklé u LaTeXu, je dosti zmatečné a nedává uživateli dobrou informaci k lokalizaci chyby. Rovněž je nutné udržet argumenty při kreslení malé, aby nedošlo k přeplnění.

Požadavky na vzhled popisu je třeba zadávat *předem*, protože uvnitř seznamů nejsou jakékoliv úpravy povoleny. Symboly vykreslovaných bodů je třeba uvádět podle `\curvesymbol`, pokud jsou grafickými objekty. U matematických symbolů se `\phantom{}` vypouští (viz příklad).

Komplexní příklad – SNcurv

Mějme za úkol zpracovat protokol z únavových zkoušek materiálů. Mělo by v něm být označení zkoušeného materiálu, diagram s měřenými body, regresní čarou a pásmem spolehlivosti a nakonec i tabulka naměřených a zpracovaných dat. Dále žádejme, aby dílčí výsledky bylo možno zobrazovat postupně při prezentacích. Z tohoto důvodu bude nutné celou úlohu rozdělit na etapy a ty podle potřeby přerušovat. Celá úloha je řešena makrem `\SNcurve`, které vyvolává další makra plnící specifické úkoly.

Aby se snížilo nebezpečí interakce výpočtů s přerušovanými grafickými výstupy je účelné, aby všechny výpočty byly provedeny nejdříve. To zajišťuje první makro, které nese název `\SNcalc`. Jím se počítají koeficienty regresní funkce, pásma spolehlivosti a načítají tabulky pro grafické výstupy a závěrečnou tabulku výsledků. Následuje posloupnost volání maker pro výstup pracovní plochy diagramu `\Dgrid` a vykreslení měřených bodů, regresní funkce a pásem spolehlivosti pomocí šestice maker `\Drawfun`. Tato volání jsou na vhodných místech proložena voláním uživatelova makra `\SNuser`, které podle uživatelova přání může při prezentacích zastavovat postup zobrazování výsledků.

Regresní funkce

Regresní funkce se nazývá Wöhlerovou křivkou nebo v anglicky mluvících oblastech také S-N křivkou. Vynáší se do diagramu, v němž jsou z historických důvodů zaměněny osy, takže nezávisle proměnná – amplituda napětí σ – se vynáší na svislou osu a závisle proměnná – počet cyklů do lomu N – na vodorovnou osu. Návod ke zpracování únavových experimentálních dat lze nalézt v dosud platné normě [5]. Bez dalších

podrobností uveďme, že budeme hledat regresní funkci ve tvaru

$$\log N = a + b \log \sigma,$$

v níž N je závisle proměnná, počet harmonických cyklů do lomu vzorku zatěžovacího procesu vyvolávajícího ve zkoušeném vzorku napětí (nezávisle proměnnou) o amplitudě σ z množiny n zkoušek o výsledcích $(\sigma_i, N_i), i = 1, \ldots, n$. Neznámé koeficienty a a b regresní funkce dostaneme řešením maticové rovnice plynoucí z metody nejmenších čtverců

$$\begin{bmatrix} n\,, & S_\sigma \\ S_\sigma, & S_{\sigma\sigma} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} S_N \\ S_{\sigma N} \end{bmatrix},$$

kde použité symboly jsou tvořeny podle schématu $S_x = \sum_{i=1}^{n} \log x_i$, případně $S_{xy} = \sum_{i=1}^{n} \log x_i \, \log y_i$. Odtud plyne, že

$$a = \frac{S_N - b\, S_\sigma}{n},$$

$$b = \frac{n\, S_{\sigma N} - S_\sigma\, S_N}{n\, S_{\sigma\sigma} - (S_\sigma)^2}.$$

Potřebné jsou i výběrové statistiky pozorování – výběrový průměr logaritmu nezávisle proměnné (amplitudy napětí)

$$\mu(\log \sigma) = \frac{S_\sigma}{n}$$

a výběrový rozptyl logaritmu závisle proměnné (počtu cyklů do lomu) se určí z formule

$$s^2(\log N) = \frac{S_{NN} - (a\, S_N + b\, S_{\sigma N})}{\nu},$$

kde $\nu = n - 2$. Jemu odpovídající výběrová směrodatná odchylka je

$$s(\log N) = \sqrt{s^2(\log N)}.$$

Pásmo spolehlivosti

Označme pro zjednodušení zápisu $x = \log N$ a $y = \log \sigma$. Potom oboustranný konfidenční interval pro střední hodnotu $E(x; y)$ v bodě y se vypočte z nerovnosti

$$x(a, b; y) - k(y) \;<\; E(x; y) \;<\; x(a, b; y) + k(y),$$

kde $x(a,\, b;\, y)$ je bod regresní funkce a $k(y)$ polovina šířky pásma, která se vypočte z formule

$$k(y) = t_\alpha(\nu)\, s(\log N) \left[\frac{1}{n} + \frac{n\,(\log\sigma - \mu(\log\sigma))^2}{n\,S_{\sigma\sigma} - S_\sigma^2} \right]^{1/2} .$$

Kritickou hodnotu $t_\alpha(\nu)$ Studentova rozdělení t lze sice nalézt pro libovolný počet stupňů volnosti $\nu = n - 2$ a vybrané hodnoty α v tabulkách (viz např. [6]), ale to je pro automatizované zpracování dat nevhodné. Byly proto nalezeny funkční náhrady pro nejčastější šířky pásem spolehlivosti $P = 90\%$ a $P = 95\%$, jimž odpovídají $\alpha = 1 - (100 - P)/200$, tedy $\alpha = 0,95$ a $\alpha = 0,975$.

Regresní funkce pro $t_\alpha(\nu)$ měly tvar polynomů v mocninách $1/\nu$

$$t_\alpha(\nu) \doteq \sum_{k=0}^{4} \frac{c_k}{\nu^k}$$

s koeficienty z vedlejší tabulky:

k	$c_k;\ P = 90\%$	$c_k;\ P = 95\%$
0	1.64498489	1.96166717
1	1.51730932	2.31430561
2	1.50707503	3.42121967
3	0.58878795	0.25509826
4	1.05584280	4.75370697

Aplikace

Mějme za úkol zpracovat protokol o sérii únavových zkoušek vrubovaných vzorků z materiálu ČSN 411523.1 o mezi únavy 120 MPa.

```
\sffamily%
 \noindent%
 \begin{center}%
  \edef\matdata%        Nadpis protokolu, materiál, mez únavy
    {Wöhlerova křivka, ČSN 41\,1523.1, 120}%
  \edef\diaglen%        Rozměry os diagramu
    {90,60}%                xlen, ylen
  \edef\diagdxy%        Odsazení diagramu od počátku
    {10,8,4}%               dx, dy, prostor na záhlaví
  \edef\expdata%        Experimentální data s klesajícím napětím
    {A,200,133.738,%        vzorek, sigma [MPa], N [tisíce cyklů]
     B,170,464.717,%
     C,140,2216.019,%
     D,130,4947.244,%
     E,120,10000.001}%      neukončená zkouška
  \edef\xgrid%          Seznam souřadnic x mřížky v tisících cyklů
    {100:100:1000,1500,2000:1000:10000}%
```

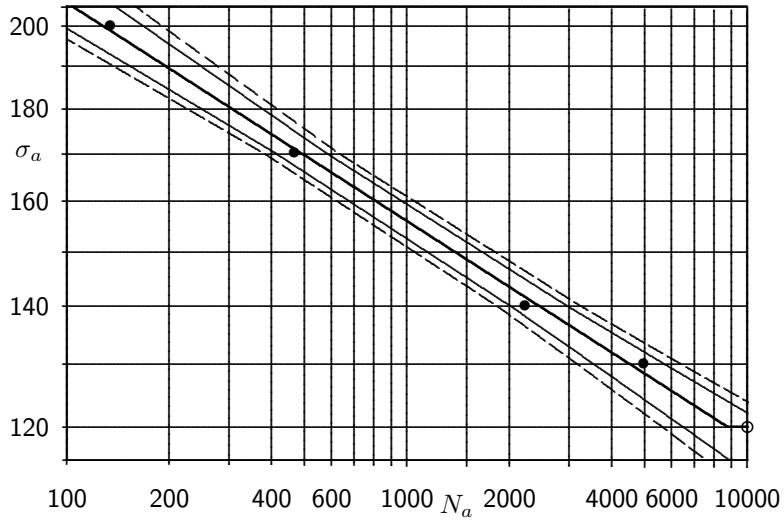
```
  \edef\xlabels%        Popis osy x
    {$N_a$,1400,-5,%
     100,200,400,600,%
     1000,2000,4000,6000,10000}%
  \edef\ygrid%          Seznam souřadnic y mřížky v [MPa]
    {115,120:10:200,205}%
  \edef\ylabels%        Popis osy y
    {$\sigma_a$,170,-5,%
     120,140,160,180,200}%
%
  \SNcurv%              Výpočty a diagram
   (\matdata)(\diaglen)(\diagdxy)%
   (\expdata)%
   (\xgrid)(\xlabels)%
   (\ygrid)(\ylabels)%
  \lines0%

  \SNtable%             Tabulka výsledků
  \lines0%
  \rmfamily%
 \end{center}%
 }%
```

Wöhlerova křivka

Materiál: ČSN 41 1523.1 4. října 2004

$\sigma_c = 120\,\mathrm{MPa}\,, \qquad N_c = 8705309$ cyklů

$\log N_a = 24.1300 - 8.2678\,\log \sigma_a$



Tabulka měřených dat, hodnot regresní funkce a pásem spolehlivosti:

#	měření		regrese	$P = 90\%$		$P = 95\%$	
	σ_a	N_a	N_r	N_ℓ	N_h	N_ℓ	N_h
A	200	133.738	127.522	97.321	167.094	85.627	189.915
B	170	464.717	488.804	411.900	580.066	379.819	629.060
C	140	2216.019	2433.785	2009.499	2947.654	1835.189	3227.628
D	130	4947.244	4491.365	3540.116	5698.221	3162.702	6378.205
E	120	10000.001	8705.309	6461.844	11727.675	5611.128	13505.736

Navenek jsou pro uživatele závažné pouze následující úkoly – sestavit seznamy dat a vyvolat dvě makra – `\SNcurv` a `\SNtable`. Tato makra jsou součástí speciálního stylu `SNcurv.sty`, který je zapotřebí uvést v záhlaví dokumentu příkazem `\usepackage{SNcurv}`.

Z popisu dat je patrna jejich skladba, a proto se jí nebudeme podrobně zabývat. Uveďme jen, že amplitudy napětí se vkládají v megapascalech a počty cyklů do lomu v tisících. Seznamy pro kreslení osnovy a popis os mohou využívat zápisu obvyklého u lineárních seznamů (viz popis makra `\For`).

Makro `\SNcurv` zanalyzuje své parametry a vytvoří z nich jiné pro `\SNcalc` a další makra pro kreslení. `\SNcalc` najde parametry a a b regresní funkce, jejíž hodnoty vypočte jak pro měřené napěťové úrovně, tak i pro dolní a horní mez pracovní plochy. Dále pro stejné úrovně vypočte hodnoty 90 a 95 procentních pásem spolehlivosti vypočteného odhadu. Z vypočtených hodnot se sestavují pracovní tabulky. U všech funkcí potom prověřuje jejich zobrazitelnost uvnitř pracovní plochy a případné přečnívající konce se oříznou makrem `\modtbl`. Za tím účelem se musí vyřešit průsečíky uvedených čar s levou a pravou mezí zobrazovaného intervalu pomocí `\ysolve`. Navíc je zapotřebí zalomit regresní čáru vytaženou silně na hladině meze únavy σ_c. Bod zlomu N_c se vypočte pomocí makra `\xsolve`. Ukončené zkoušky se vyznačují plnými tečkami a neukončené se vynesou na pravém okraji pracovní plochy prázdnými kroužky.

Vlastní diagram se kreslí pomocí maker z `diagram.sty` a to osnova pomocí `\Dgrid` měřené body a funkce pomocí `\Drawfun` a hlavička diagramu pomocí `\Dtext`. Všechna tato makra se volají z makra `\SNcurv`. Tabulka zpracovaného měření vytvořená již v makru `\SNcalc` potom vystoupí vyvoláním `\SNtable`. Hlavička tabulky bude mít v závislosti na jazyku dokumentu český anebo anglický text.

Podpůrná makra – `support.sty`

Funkce tak rozsáhlého balíku je nemyslitelná bez významné podpory celé řady maker řešících dílčí úkoly. Protože tato makra mohou být užitečná pro uživatele i jako samostatná, uveďme zde jejich stručný popis.

Makra pro podporu popisů

Tato část knihovny `support.sty` obsahuje několik maker, která se osvědčila při vkládání textů do popisů diagramů i při zápisu dokumentů včetně tohoto.

`\bs` **Zpětné lomítko**

Povel lze použít jak v textovém tak i v matematickém módu. Ovšem generované znaky se v obou módech navzájem liší.

Příklad:

Zápis `\bs, $\bs$` způsobí výstup znaků \bs, $\bs$

`\ds` **Zkrácení zápisu povelu `\displaystyle`**

Použije se k zpřehlednění zápisů matematických výrazů.

`\{` **Levá složená závorka**

`\}` **Pravá složená závorka**

Tato makra slouží k předefinování existujících maker pro složené závorky. V textovém režimu dají na výstup složené závorky ve stylu psacího stroje, v matematickém režimu pružné složené závorky.

Příklady:

Zápis	dodá na výstup
`\{\}`	{}
`$\{\ds\sum_{k=1}^N a_k\}$`	$\left\{\displaystyle\sum_{k=1}^{N} a_k\right\}$

`\h` **Pružná mezera**

Tento povel je na rozdíl od `\hfill`, ze kterého je vytvořen, nejen kratší, ale funguje i na začátku řádky. Jeho užití oceníme zejména při centrování popisů v hlavičkách tabulek ve sloupcích deklarovaných jinak než pomocí 'c'.

Příklad:

Centrování textu: `|\h TEXT\h|` udělá |\h TEXT\h|

`\lines{`⟨*arg*⟩`}` **Vytvoření prázdných řádek**

Tímto povelem lze odřádkovat tak, že vznikne ⟨*arg*⟩ prázdných řádek. Hodnotou v parametru ⟨*arg*⟩ může být reálné (tedy i necelé nebo záporné) číslo.

Příklady:

`\lines0`	přejde na novou řádku (žádnou nevynechá)
`\lines{10}`	vytvoří 10 prázdných řádek

`\T{`⟨*arg*⟩`}` **Zkrácení zápisu povelu `\texttt`**

Povel pro změnu stylu písma na písmo psacího stroje se běžně zajistí povelem `\verb|`⟨*arg*⟩`|`. Avšak ve všech příkazech a prostředích se nemůže `\verb|`⟨*arg*⟩`|` použít. Tam je nutno v případě potřeby strojopisného textu zajistit výstup pomocí `\texttt{text}`. Je-li takových výstupů více, jako např. v tabulkách nebo makrech, je vhodné zkrátit dobu výstavby dokumentu a i zpřehlednit jejich zápis pomocí povelu `|\aarg|`. Protože se ⟨*arg*⟩ na rozdíl od `\verb` napřed expanduje, musí se všechny aktivní znaky zapisovat s úvodním znakem `\`.

Příklady:

`\T{\bs oval(25,5)}`	vytvoří text	`\oval(25,5)`
`\T{\bs setn\{n\}\{123\}}`	vysází	`\setn{n}{123}%`

`\ifcz{`⟨*TRUE*⟩`}{`⟨*FALSE*⟩`}` **Test jazyka dokumentu**

Makro se použije v makrech, které mohou sloužit k vytvoření dokumentů v libovolném jazyce. Makro otestuje, zda je dokument český, a pokud ano, provede větev `{`⟨*TRUE*⟩`}` a v opačném případě větev `{`⟨*FALSE*⟩`}`.

Příklad:

`\ifcz{měření}{measurement}%` dá do českého dokumentu: `měření`

`\ifm{`⟨*arg*⟩`}` **Výpis argumentu v textovém i matematickém módu**

Povel se s výhodou použije při tvorbě maker pro výstup matematických symbolů, které se často vyskytují jak v textu tak i ve formulích v matematickém prostředí. Při vyvolání se jeho argument v matematickém prostředí interpretuje beze změny, zatímco v textovém režimu se obklopí znaky dolar. V LaTeX2e tuto funkci zajišťuje `\ensuremath`. Jako příklad použijme dále uvedený příkaz pro výpis matic a vektorů.

Příklad:

`\def\mx#1{\ifm{\mbox{\boldmath $#1$}}}%` výpis matice, vektoru

`\mx` ⟨*arg*⟩ **Vektor nebo matice**
`\dmx` ⟨*arg*⟩ **Derivace vektoru nebo matice**
`\ddmx` ⟨*arg*⟩ **Druhá derivace vektoru nebo matice**

Makra slouží pro výstup jednopísmenového argumentu ⟨*arg*⟩ do výsledného dokumentu bez ohledu na to, zda jsou použity v matematickém nebo textovém prostředí. Je vhodné používat pro vektory malá písmena a pro matice velká. Zvolený symbol se vysází v tučné matematické kurzivě, i když nejde o zcela ustálenou zvyklost. Je to však v souladu s doporučením a značením z knihy prof. Rektoryse [7].

Příklady:

```
$\mx c = \mx a^T\,\mx b$%
```
vysází skalární součin $\boldsymbol{c} = \boldsymbol{a}^T \boldsymbol{b}$

```
$\mx M\,\ddmx q(t) + \mx B\,\dmx q(t) + \mx K\, q(t) = \mx f(t)
```
vysází maticovou diferenciální rovnici $\boldsymbol{M}\,\ddot{\boldsymbol{q}}(t) + \boldsymbol{B}\,\dot{\boldsymbol{q}}(t) + \boldsymbol{K}\,\boldsymbol{q}(t) = \boldsymbol{f}(t)$

Obecná makra

Jsou sem zařazena makra často velmi jednoduchá, ale užitečná pro stavbu složitějších maker. Umožňují např., aby bylo možno pracovat s čítači bez ohledu na jejich „původ“. TEX má k dispozici 256 čítačů, které s ním sdílí i LATEX. Vnější vzhled jmen čítačů se u obou liší. Zatímco jména texovských čítačů mají formálně tvar povelů, protože začínají zpětným lomítkem, latexovské čítače navenek tak nevypadají, protože jejich jména jsou složena jen z písmen. To je ale jen zdání, protože při deklaraci čítače LATEXu se vnitřně předřadí před jeho jméno „předpona“ `\c@`. Aby to bylo možné, vytváří se čítač pro LATEX povelem `\newcounter{`⟨*jméno*⟩`}`, kdežto pro TEX povelem `\newcount\`⟨*jméno*⟩, kde pod pojmem ⟨*jméno*⟩ chápeme v obou případech řetězec písmen.

Pro sjednocení obou pojmů užívejme označení ⟨*cnt*⟩ pro oba typy čítačů, příp. ⟨*T-cnt*⟩ pro čítače TEXu a ⟨*L-cnt*⟩ pro čítače LATEXu. Pro práci s nimi se sestavily následující příkazy, které na rozdíl od svých vzorů lze užívat i v makrech:

`\newcnt{`⟨*cnt*⟩`}` **Podmíněná definice obecného čítače**

Makro lze užít pro vytváření ⟨*T-cnt*⟩ i ⟨*L-cnt*⟩. Nejdříve se otestuje, zda požadovaný čítač je již definován. Pokud tomu tak není, vytvoří se. V případě, že ⟨*cnt*⟩ je ⟨*T-cnt*⟩, lze složené závorky vynechat.

Příklad:

`\newcnt\Fun%` vytvoří čítač `\Fun`
`\newcnt{Fun}%` vytvoří čítač `\c@Fun` – v LATEXu pojmenovaný `Fun`

`\newlen{`⟨*len*⟩`}` **Podmíněná definice délkového registru**

V případě, že není délkový registr ⟨*len*⟩ požadovaného jména nalezen, vytvoří se.

Příklad:

```
\newlen \delka
```

`\Cnt{`⟨*cnt*⟩`}=`⟨*arg*⟩ **Podmíněná definice a naplnění čítače**

Makro vyvolá `\newcnt` a potom dosadí do něho hodnotu z ⟨*arg*⟩, která musí být celočíselná.

Příklady:

`\Cnt {xa}=12345%`	vytvoří čítač `xa` a dosadí 12345
`\Cnt \ft={xa}%`	vytvoří čítač `\ft` a dosadí obsah xa
`\Cmd \xb={4321}%`	vytvoří povelu `\xb` (TEX)
`\Cnt \gt=\xb%`	vytvoří čítač `\gt` a dosadí číslo z `\xb`

`\Cmd` ⟨*cmd*⟩`={`⟨*arg*⟩`}` **Definice povelu**

Toto makro je shodné či podobné s definicemi, které mají tvar:

TEX : `\edef`⟨*cmd*⟩`{`⟨*arg*⟩`}%`,

`fp.sty`: `\FPset`⟨*cmd*⟩⟨*arg*⟩`%`.

Tělo definice se v těchto příkazech expanduje a výsledek se dosadí do definovaného příkazu na rozdíl od definice

LATEX : `\newcommand{`⟨*cmd*⟩`}{`⟨*arg*⟩`}%`,

která definuje *podprogram*, který se při každém vyvolání znovu provádí, a proto může dát při každém vyvolání jinou hodnotu. To nastane v případě, že ⟨*arg*⟩ je příkazem, který se od okamžiku definice nebo redefinice změnil.

Příklady:

```
\newcommand{\D}{\C}%
\Cmd \C={1}            \wric\D%   *** \D = 1
\Cmd \C={23456}        \wric\D%   *** \D = 23456
\Cmd \C={\D}           \wric\D%   *** \D = 23456
\Cmd \D={1}            \wric\C%   *** \C = 23456
```

Výsledky příkladů potvrzují rozdílnost obou definic. Povel `\D` nabývá okamžitých hodnot povelu `\C`, kdežto povel `\C` je na pozdější změnu argumentu necitlivý.

`\Arg{`⟨*arg*⟩`}` **Celočíselný obsah obecného argumentu**

Makro funguje jako funkce aplikovaná na zcela obecný argument s celočíselným obsahem. Argumentem může být číslo, čítač nebo povel.

Protože výsledek neukládá, lze ho přímo použít jako argument jiného makra. Užívá se v makrech, do nichž mohou vstupovat argumenty různého typu.

Příklad: `\newcnt\n \n=\Arg{#1}%  naplní \n 1. argumentem makra`

Celočíselná aritmetika

Jak TEX tak i LATEX poskytují uživateli jisté minimální prostředky pro základní aritmetické operace s celými čísly a s čítači. Ovšem původní příkazy pro tyto operace jsou formálně rozvleklé a bez systematické stavby. Jsou to příkazy TEXu

`\advance` ⟨*T-cnt*⟩⟨*iarg*⟩`%`	pro součet, ⟨*T-cnt*⟩:=⟨*T-cnt*⟩+⟨*iarg*⟩
`\advance` ⟨*T-cnt*⟩`-`⟨*iarg*⟩`%`	pro rozdíl, ⟨*T-cnt*⟩:=⟨*T-cnt*⟩-⟨*iarg*⟩
`\multiply` ⟨*T-cnt*⟩⟨*iarg*⟩`%`	pro součin, ⟨*T-cnt*⟩:=⟨*T-cnt*⟩∗⟨*iarg*⟩
`\divide` ⟨*T-cnt*⟩⟨*iarg*⟩`%`	pro podíl, ⟨*T-cnt*⟩:=⟨*T-cnt*⟩/⟨*iarg*⟩,

kde ⟨*T-cnt*⟩ je čítač TEXu a ⟨*iarg*⟩ může být buď ⟨*T-cnt*⟩ nebo celé číslo, anebo povel s celočíselným výrazem. V LATEXu jsou k dispozici pouze nejjednodušší příkazy pro práci s čítači LATEXu ⟨*L-cnt*⟩:

`\setcounter` ⟨*L-cnt*⟩⟨*num*⟩`%`	dosadí celé číslo ⟨*num*⟩ do ⟨*L-cnt*⟩
`\addtocounter` ⟨*L-cnt*⟩⟨*num*⟩`%`	přičte celé číslo ⟨*num*⟩ do ⟨*L-cnt*⟩
`\stepcounter` ⟨*L-cnt*⟩`%`	přičte jedničku do čítače ⟨*L-cnt*⟩

Proto se vypracoval systém celočíselné aritmetiky, kterým se odstranily výše uvedené nedostatky a doplnily se některé další operace. Makra celočíselné aritmetiky jsou uspořádána do třech skupin, a to na makra pro práci s ⟨*L-cnt*⟩ a celými čísly vracející výsledek do ⟨*L-cnt*⟩ a makra obecná.

Binární operace s celými čísly v LATEXu

Pokud víme, jaké budou typy parametrů, lze postavit makra neobsahující dešifrování typů. Mezi ně patří makra, v nichž první z operandů je ⟨*L-cnt*⟩ a druhý z operandů je celé číslo, jméno příkazu nebo ⟨*T-cnt*⟩, vždy s celočíselným obsahem. Na památku toho je koncovým znakem jména makra písmeno `n`. Je třeba upozornit, že cílový čítač ⟨*L-cnt*⟩ musí být již definován. Jsou-li definovány čítač ⟨*T-cnt*⟩ jako `\Tn` nebo jméno povelu ⟨*cmd*⟩ jako `\C`, oba s obsahem 123, potom všechny příkazy z každé trojice dají stejný výsledek v čítači `n`.

`\setn{`⟨*L-cnt*⟩`}{`⟨*arg*⟩`}`	`\setn{n}{123}%` `\setn{n}\Tn \setn{n}\C%`	`n:=123`
`\addn{`⟨*L-cnt*⟩`}{`⟨*arg*⟩`}`	`\addn{n}{123}%` `\addn{n}\Tn \addn{n}\C%`	`n:=n+123`
`\subn{`⟨*L-cnt*⟩`}{`⟨*arg*⟩`}`	`\subn{n}{123}%` `\subn{n}\Tn \subn{n}\C%`	`n:=n-123`
`\muln{`⟨*L-cnt*⟩`}{`⟨*arg*⟩`}`	`\muln{n}{123}%` `\muln{n}\Tn \muln{n}\C%`	`n:=n*123`
`\divn{`⟨*L-cnt*⟩`}{`⟨*arg*⟩`}`	`\divn{n}{123}%` `\divn{n}\Tn \divn{n}\C%`	`n:=n/123`

Binární operace s čítači LaTeXu

Je-li druhým operandem ⟨*L-cnt*⟩, nelze s ním pracovat stejným způsobem jako s celými čísly, ale je zapotřebí z jeho jména dostat obsah. To se v LaTeXu zajišťuje povelem `\value`, např. `\value{n}`. Rovněž je možno použít příkaz TeXu `\the`, v našem případě `\then`. Dále uvedené příkazy, které tento převod zajišťují samy, mají názvy zakončené písmenem `c` (counter). Předpokládejme dále, že jsou již definovány čítače `m` a `n`.

`\setc{`⟨*L-cnt*⟩$_a$`}{`⟨*L-cnt*⟩$_b$`}%`	`\setc{n}{m}`	`n:=m`
`\addc{`⟨*L-cnt*⟩$_a$`}{`⟨*L-cnt*⟩$_b$`}%`	`\addc{n}{m}`	`n:=n+m`
`\subc{`⟨*L-cnt*⟩$_a$`}{`⟨*L-cnt*⟩$_b$`}%`	`\subc{n}{m}`	`n:=n-m`
`\mulc{`⟨*L-cnt*⟩$_a$`}{`⟨*L-cnt*⟩$_b$`}%`	`\mulc{n}{m}`	`n:=n*m`
`\divc{`⟨*L-cnt*⟩$_a$`}{`⟨*L-cnt*⟩$_b$`}%`	`\divc{n}{m}`	`n:=n/m`

Unární operace s čítači LaTeXu

`\incc{`⟨*L-cnt*⟩$_a$`}%`	`\incc{n}`	`n:=n+1`
`\negc{`⟨*L-cnt*⟩$_a$`}%`	`\negc{n}`	`n:=-n`
`\absc{`⟨*L-cnt*⟩$_a$`}%`	`\absc{n}`	`n:=|n|`

Makra pro práci s obecnými čítači

V této skupině jsou makra pracující s obecnými předem definovanými *celočíselnými* veličinami ⟨*arg*⟩, jimiž mohou být celá čísla, povely a čítače ⟨*T-cnt*⟩ a ⟨*L-cnt*⟩. Pokud se výsledek někam ukládá, tak výhradně do obou typů čítačů, např. do `n` i `\n`. Do této skupiny zařadíme i

makra realizující unární operace, jako jsou absolutní hodnota, znaménko a změna znaménka a navíc i víceparametrické operace minima a maxima.

`\Set{`⟨*cnt*⟩`}{`⟨*arg*⟩`}%`	`\Set{\n}{123}`	`\n:=123`
`\Add{`⟨*cnt*⟩`}{`⟨*arg*⟩`}%`	`\Add{n}\Ta`	`n:=n+\Ta`
`\Sub{`⟨*cnt*⟩`}{`⟨*arg*⟩`}%`	`\Sub\n\C`	`\n:=\n-\C`
`\Mul{`⟨*cnt*⟩`}{`⟨*arg*⟩`}%`	`\Mul{n}{123}`	`n:=n*123`
`\Div{`⟨*cnt*⟩`}{`⟨*arg*⟩`}%`	`\Div\n\Ta`	`\n:=\n/\Ta`
`\Abs{`⟨*cnt*⟩`}%`	`\Abs\n`	`\n:=|\n|`
`\Neg{`⟨*cnt*⟩`}%`	`\Neg{n}`	`n:=-n`
`\Sgn{`⟨*cnt*⟩`}%`	`\Sgn\n`	`\n:=sign{\n}`
`\Min{`⟨*cnt*⟩`}(seznam)%`	`\Min`⟨*cnt*⟩`(`n_1`,`n_2`,...,`n_n`)`	`n:=minimum`
`\Max{`⟨*cnt*⟩`}(seznam)%`	`\Max`⟨*cnt*⟩`(`n_1`,`n_2`,...,`n_n`)`	`n:=maximum`

Převod obsahu délkového registru na povel bez „pt“

`\unpt`⟨*len*⟩`%`	`\unpt{\hsize}`	`312.9803`

Klávesnicové vstupy

Jsou případy, kdy je vhodné ovlivnit chod překladu operativně zadaným parametrem. Nemá-li se celý soubor editovat, zbývá jen málo možností, jak to zařídit. Jednou z nich je vložení onoho parametru z klávesnice během překladu zdroje dokumentu. Bohužel LaTeX nepočítá s touto potřebou, takže nemá prostředky, jak to umožnit. Naštěstí TeX má pro operativní vstup z klávesnice příkaz

`\typein[`⟨*cmd*⟩`]{`⟨*libovolný text*⟩`}`

Povinný parametr, ⟨*libovolný text*⟩, má funkci výzvy, která vystoupí na obrazovce. Odpověď na ni vložená klávesnicí a zakončená stiskem `Enter` se potom uloží do povelu ⟨*cmd*⟩. Toho se využilo v knihovně `support.sty`, která obsahuje dvě specializovaná makra pro tento účel. Jedním lze vytvářet a naplňovat čítače a druhým povely.

`\Ikbd{`⟨*cnt*⟩`}` **Klávesnicový vstup celého čísla do čítače**

Po vyvolání makra se na obrazovce objeví výzva se jménem čítače, který se vytvoří, pokud nebyl již předem definován:

```
******* Integer: ⟨cnt⟩ *******
  \integ=_
```

Do místa kurzoru se postupným stlačováním číslicových kláves vloží žádaný údaj a stiskne `Enter`. Nato se posloupnost vložených číslic zkonvertuje na celé číslo a uloží do žádaného čítače.

Příklady:

```
\Ikbd \n%    vytvoření a naplnění čítače ⟨T-cnt⟩
\Ikbd {n}%   vytvoření a naplnění čítače ⟨L-cnt⟩
```

`\Ckbd{`⟨*cmd*⟩`}` **Klávesnicový vstup řetězce do povelu**

Po vyvolání tohoto makra vystoupí na obrazovce text

```
******* Command: ⟨cmd⟩ *******
\cmd=_
```

a čeká se na vložení libovolného řetězce znaků zakončeného stisknutím `Enter` z klávesnice. Tato posloupnost znaků se uloží do nově vytvořeného povelu ⟨*cmd*⟩, jehož jméno si zvolil uživatel sám. Kdykoliv se v budoucnosti užije v dokumentu tento povel, vystoupí vložená posloupnost znaků, pokud neobsahovala další povely (neterminální symboly). V případě, že je obsahovala, dojde k dalšímu rozkladu a výstupu konečného textu. Speciálním případem možného řetězce je i reálné číslo (s desetinnou tečkou).

Příklad:

```
\Ckbd \vaha%  vytvoření a naplnění povelu \vaha
              reálným číslem z klávesnice
```

Kontrolní výstupy na obrazovku a do `*.log`

Pro operativní výstup na obrazovku umožňuje TEX použít příkaz

`\typeout{`⟨*libovolná zpráva*⟩`}`

Text ⟨*libovolná zpráva*⟩ může obsahovat i povely, které se však před výstupem na obrazovku rozvinou.

Při práci na nových makrech i na složitějším dokumentu je pro uživatele často důležité znát okamžité hodnoty veličin, se kterými se pracuje, případně místo, které se právě zpracovává. Pro tyto účely lze sice užít již zmíněný příkaz `\typeout`, avšak pro častěji se vyskytující typy výstupů je výhodnější sestavit účelová makra. Dočasný výstup informace o ladění do cílového dokumentu není vhodný, protože právě stránka, na níž došlo k chybě nevystoupí a tedy ani nejdůležitější informace o stavu před chybou nejsou k dispozici. Ani v případech, kdy je objem výstupů

na obrazovku velký, není výstupní informace ztracena, protože se ukládá do souboru `*.log`, který lze dodatečně prohlížet. Pro ladění se osvědčila makra, která posílají svůj argument na novou řádku obrazovky.

`\wri{`⟨*reg*⟩`}` **Kontrolní výstup obsahu registru**

Jde o velice univerzální makro, které lze použít pro výstup jak obecného čítače, tak i délkového registru, který obsahuje informaci o délce v bodech s rozměrem `pt`. Jeho vyvolání má při překladu za následek výstup textu `***` ⟨*jméno registru*⟩ `=` ⟨*hodnota*⟩ na obrazovku a do `*.log` souboru. Podle vystupujících hodnot lze usuzovat na správnost výsledků při ladění nebo na hloubku zpracování (např. u cyklů).

Příklady:

```
\wri{page}%          *** page = 49
\wri\baselineskip%   *** \baselineskip = 11.77008pt
```

`\wric{`⟨*cmd*⟩`}` **Kontrolní výstup obsahu povelu**

Makro lze užít pouze pro povely ⟨*cmd*⟩ které obsahují znakovou informaci, nebo i pro skupinu povelů uzavřenou ve složených závorkách, která je schopna se okamžitě rozvinout v řetěz znaků. Výstup má na obrazovce tvar

`***` ⟨*cmd*⟩ `=` ⟨*obsah povelu*⟩

Příklady:

```
\wric\FPpi          *** \FPpi = 3.141592653589793238
\wric{\the\hsize}   *** \the \hsize = 312.9803pt
```

Pro poslední příklad by ovšem bylo vhodnější použít `\wri\hsize`.

`\wris{`⟨*poznámka*⟩`}` **Výstup řetězce znaků**

Toto makro je výhodné pro výstup libovolné zprávy na obrazovku během překladu. Zpráva je zde chápána jako libovolná posloupnost znaků, jejichž původ se na rozdíl od `\wri` a `\wric` neidentifikuje názvem (protože řetězec žádný název nemá). Obecný tvar výstupu je

`*** řetězec znaků po případném rozvoji`

Odtud je patrno, že případné povely, které mohl původní řetězec obsahovat, se nejdříve rozvinou a vystoupí až výsledná posloupnost znaků.

Příklady:

```
\wris{--------------}     →
  *** --------------   čára pro oddělování výstupů v cyklech
\wris{cislo e = \FPe}     →
  *** cislo e = 2.718281828459045235
```

Práce se seznamy položek

Pod pojmem *seznam hodnot* chápeme množinu položek navzájem oddělených čárkami. Obvykle jsou uloženy v povelech, ale pokud jsou delší, bývají uloženy ve vlastních souborech, které je třeba ke zpracování nejdříve přečíst.

Standardně umí TEX i LATEX přečíst informace ze samostatných souborů pomocí příkazů `\input jméno_souboru`. Toho se běžně využívá při tvorbě rozsáhlých dokumentů. Pokud vstupní posloupnost znaků neobsahuje příkazy, posílá se do výstupního dokumentu. V opačném případě se příkazy vykonají a do dokumentu se pošlou až výsledné sekvence. Jiná situace nastává, chce-li uživatel LATEXu přečíst data, která se teprve budou zpracovávat, a tudíž nesmějí být zaslána do výsledného dokumentu přímo, jako je tomu u příkazu `\input`. Za tím účelem byla sestavena makra, která umožňují číst data ze vstupních souborů a z požadovaných vytvářet seznamy, které lze potom analyzovat a výsledky nakonec použít k tvorbě dokumentu. K tomu se užijí dále uvedená makra:

Vytváření seznamů

`\loadgf` ⟨*cmd*⟩`={`⟨*jméno souboru*⟩`}` **Čtení seznamu ze souboru**
Makro otevře deklarovaný `soubor` pro čtení. Soubor musí být znakový a musí obsahovat seznam položek navzájem oddělených čárkami. Soubor se čte po textových řádkách (ke znaku nové řádky) a každá nově načtená řádka se připojí na konec uživatelem požadovaného povelu `\cmd`, který se tímto příkazem i vytvoří. Po přečtení všech řádek textu se vstupní soubor uzavře. Implicitní koncovkou jména souboru je `.tex`. V tom případě lze toto rozšíření jména vynechat. Je-li rozšíření jiné, musí se uvést plné jméno souboru, případně i s cestou, pokud soubor leží v jiném adresáři než v aktuálním.

Příklady:

```
\loadgf \matx={data.asc}%     vytvoření příkazu \matx s daty
\loadgf \xy={funkce.dat}%     Vytvoření příkazu \xy
```

`\getxy` ⟨cmd_1⟩`=`⟨cmd_2⟩`(`⟨*ix*⟩`,`⟨*iy*⟩`)` **Výběr dvojice řádek z matice**
Toto makro je obvykle následovníkem předcházejícího makra `\loadgf`, kterým se přečetl seznam do povelu ⟨cmd_2⟩, např. `\matx` obsahující strukturu složenou z následujících údajů:

⟨*počet řádek*⟩, ⟨*počet sloupců*⟩, ⟨*prvky matice po sloupcích*⟩

Vstupní soubor může ale obsahovat souřadnice více závislostí, z nichž

v daném okamžiku chce uživatel pracovat pouze s některými. Makro `\getxy` je navrženo právě pro výběr souřadnic bodů jedné závislosti `y(x)` z matice. Makro vybírá dvojice hodnot souřadnic x_j = $v_{ix,j}$ a y_j = $v_{iy,j}$ z povelu ⟨*cmd*$_2$⟩ a vytváří z nich nový povel ⟨*cmd*$_1$⟩, např. `\xy`. Povel ⟨*cmd*$_1$⟩, který mohl být vytvořen příkazem `\loadgf`, obsahuje matici dat jako seznam položek oddělených čárkami. Celočíselné argumenty ⟨*ix*⟩, ⟨*iy*⟩ jsou indexy řádek matice dat.

Příklad:

```
\Cmd\xyz={3,4, 10,20,30, 40,50,60, 70,80,90, 100,110,120}%
\getxy \xy=\xyz(1,3)%         *** Matrix type (3,4)
\wric\xy% *** \xy = 10,30, 40,60, 70,90, 100,120
```

`\Appitem` ⟨*cmd*⟩+⟨*arg*⟩ **Přidání položky na konec seznamu**

Makro slouží k postupnému vytváření seznamu v uživatelem zvoleném příkazu ⟨*cmd*⟩, který se musí předem nadefinovat jako prázdný. Při každém vyvolání se na konec seznamu přidá za oddělovací čárkou (,) nová položka ⟨*arg*⟩, případně nové položky {⟨*arg*$_1$⟩,⟨*arg*$_2$⟩,...}.

Příklad:

```
\Cmd\str={}%
\Appitem\str+{1}\Appitem\str+{a}%
\Appitem\str+{2}\Appitem\str+{b}%
\wric\str%            *** \str = 1,a,2,b
```

`\Apptab` ⟨*cmd*⟩+(⟨*row*⟩) **Přidání řádky do tabulky**

Makro je určeno pro vytváření seznamu vhodného pro užití v tabulkách. Po vyvolání přidá řádku ⟨*row*⟩ do příkazu ⟨*cmd*⟩, který byl před prvním použitím prázdný. Řádka ⟨*row*⟩ je běžným seznamem s prvky oddělenými čárkami, který mohl být vytvořen např. příkazem `\Appitem`. Ve výsledném řetězci ⟨*cmd*⟩ jsou jednotlivé položky oddělovány znakem `&` a řádky pomocí znaků nové řádky tak, jak je žádáno pro výstup v prostředích `tabular` nebo `array`.

Příklad:

```
\Cmd\Tab={} \Cmd\rowa={1,10,100}%
\Apptab\Tab+(\rowa) \Apptab\Tab+(2,20,200)
\begin{tabular}{rrr}
   \Tab
\end{tabular}
```

s výsledkem

1	10	100
2	20	200

Manipulace s položkami seznamu

V řadě případů nepotřebuje uživatel pracovat s celým souborem, ale s jeho částmi. To byl již případ výběru určitých vektorů dat z matice, o kterém bylo pojednáno v předcházejícím odstavci. Někdy je ovšem zapotřebí pracovat dokonce i s jednotlivými položkami přečteného seznamu. K tomu slouží dále uvedená makra.

Pracovat se v nich bude se seznamy, které budou opět uzavírány do kulatých závorek. Dále uvedená makra použijeme v případech, že budeme potřebovat jakkoliv pracovat s položkami. Jejich pořadí je dáno indexem, který roste po jedničce ve sloupcích řazených za sebou.

`\nextitem` ⟨*cmd*$_1$⟩`=(`⟨*cmd*⟩`)` **Odebrání první položky ze seznamu**
Makrem se odtrhne první položka ze seznamu, který je obsahem příkazu ⟨*cmd*⟩, a uloží se do příkazu ⟨*cmd*$_1$⟩. O tuto položku se původní seznam zkrátí.

Příklad:

```
\Cmd \seznam={1, a, bc, 3.14}%
\nextitem \prvni=(\seznam)%
\wric\prvni%           *** \prvni = 1
\wric\seznam%          *** \seznam = a, bc, 3.14
```

`\getitem` ⟨*cmd*⟩`=(`⟨*seznam*⟩`){`⟨*idx*⟩`}` **Kopie položky ze seznamu**
Makro okopíruje položku uvedenou v seznamu ⟨*seznam*⟩na místě o indexu ⟨*idx*⟩ a vloží ji do příkazu ⟨*cmd*⟩, jehož jméno zvolí uživatel. Původní seznam zůstane beze změny.

Příklad:

```
\getitem \treti=(1, a, bc, 3.14){3}%
\wric\treti%           *** \treti = bc
```

`\putitem` ⟨*cmd*⟩`=(`⟨*seznam*⟩`){`⟨*idx*⟩`}` **Přepsání položky v seznamu**
Položka s indexem ⟨*idx*⟩ v seznamu ⟨*seznam*⟩ se nahradí novou položkou – příkazem ⟨*cmd*⟩.

Příklad:

```
\wric\seznam%          *** \seznam = 1, a, bc, 3.14
\putitem {1234}=(\seznam){3}%
\wric\seznam%          *** \seznam = 1, a,1234, 3.14
```

`\skipitems {`⟨*low*⟩`}(`⟨*seznam*⟩`){`⟨*high*⟩`}` **Vynechání konců seznamu**
Makro se užije vždy, když se na začátku a na konci seznamu mají jisté

položky vynechat. Stává se to dosti často u měření, kdy zajímavý interval dat leží někde uprostřed seznamu, kdežto začátek a konec měření jsou nadbytečné. Parametry makra jsou:

⟨*low*⟩ kladný index, do kterého včetně budou všechny položky vynechány;

⟨*seznam*⟩ běžný seznam položek oddělených čárkami, v němž je uložen jak vstupní, tak i výsledný seznam. Z tohoto důvodu *musí* být ⟨*seznam*⟩ příkazem!

⟨*high*⟩ kladný index, od kterého včetně budou všechny položky až do konce původního seznamu vynechány. Při tom musí platit, že ⟨*low*⟩ ≤ ⟨*high*⟩.

Veličiny ⟨*low*⟩ a ⟨*high*⟩ jsou celočíselnými konstantami, tedy celými čísly, celočíselnými povely nebo jmény čítačů.

Příklad:

```
\Cmd \xyz={3,4, 10,20,30, 40,50,60, 70,80,90, 100,110,120}
\skipitems{2}(\xyz){12}
\wric\xyz%        *** \xyz = 10,20,30, 40,50,60, 70,80,90
```

\getmn{⟨*m*⟩}{⟨*n*⟩}=(⟨*seznam*⟩) **Zjištění typu matice**

Jde o makro, které lze aplikovat na seznamy získané přečtením souborů určených pro makro `\getxy`. Makro zjistí ze seznamu hodnotu prvních dvou položek, totiž počtu řádek `m` a počtu sloupců `n` matice, která vyplňuje zbytek seznamu. Vstupní seznam zůstane zachován beze změny. Parametry makra jsou:

⟨*m*⟩, ⟨*n*⟩ jména čítačů pro uložení počtu řádek rep. sloupců matice, která v seznamu následuje;

⟨*seznam*⟩ běžný seznam položek oddělených čárkami, v němž první dvě položky udávají typ matice, která následuje., ale nikam se neukládá.

Příklad:

```
\newcnt{m} \newcnt{n}
\Cmd \xyz={3,4, 10,20,30, 40,50,60, 70,80,90, 100,110,120}
\getmn{m}{n}=(\xyz)
\wris{m=\them, n=\then}%     *** m=3, n=4
```

Práce s řetězci znaků

Dále uvedená makra mohou být užita k analýze předem neznámých řetězců znaků. K jinému účelu patrně nebudou sloužit, protože by bylo zbytečné analýzovat známý řetězec. Zčásti se podobají makrům pro práci s položkami seznamů a proto zde uvedeme jen jejich krátký přehled:

`\nextchr` ⟨*cmd*$_1$⟩`=(`⟨*cmd*⟩`)` **První znak v řetězci**

Makro uloží do povelu ⟨*cmd*$_1$⟩ první znak z řetězce ⟨*cmd*⟩, který zároveň o tento znak zkrátí. Původní řetězec proto *musí* být uložen v příkazu, zde označeném ⟨*cmd*⟩! Uživatel si však jeho jméno může zvolit libovolně.

Příklad:

```
\Cmd \abc={abcdefghijklmnopqrstuvwxyz}
\nextchr \ch=(\abc)
\wric\ch%          *** \ch = a
\wric\abc%         *** \abc = bcdefghijklmnopqrstuvwxyz
```

`\getchr` ⟨*cmd*⟩`=(`⟨*string*⟩`){`⟨*index*⟩`}` **Znak daný pořadím v řetězci**

Po vyvolání makra bude příkaz ⟨*cmd*⟩ obsahovat ⟨*index*⟩tý znak řetězce. Řetězec sám zůstane vyvoláním makra nezměněn. Nemusí proto být (na rozdíl od příkazu `\nextchr`) obsahem povelu. Parametrem ⟨*index*⟩ může být libovolná celočíselná konstanta.

Příklad:

```
\Cmd \abc={abcdefghijklmnopqrstuvwxyz}
\getchr \ch=(\abc){14}
\wric \ch          *** \ch = n
```

`\findchr` ⟨*znak*⟩`=(`⟨*string*⟩`){`⟨*cnt*⟩`}` **První výskyt znaku v řetězci**

Makrem se prohledává řetězec ⟨*string*⟩ a každý jeho znak se srovnává s daným vzorem ⟨*znak*⟩, který je libovolným znakem množiny ASCII. V případě, že dojde ke shodě, vloží se do čítače ⟨*cnt*⟩ celé číslo rovné pořadí nalezeného znaku od počátku řetězce. Neobsahuje-li řetězec hledaný znak, dosadí se do ⟨*cnt*⟩ záporně vzatá délka řetězce.

Příklad:

```
\newcnt{np}
\Cmd \real={123.456}%                 reálné číslo
\findchr {.}(\real){np} \wri{np}      *** np = 4%
\findchr {a}(123.456){np} \wri{np}    *** np = -7%
```

Závěry

V předcházejících odstavcích byl učiněn pokus o seznámení příznivců LaTeXu s možnostmi, které je možno využívat zejména pro vytváření diagramů v čase tvorby dokumentu bez nutnosti vyvolávat další externí prostředky pro jejich tvorbu. Jde o postup, který má jak řadu výhod tak i nevýhod. Především uživatel musí zvládnout alespoň rámcově vyvolávání maker z jejich knihoven – packages. Zadávání parametrů u složitějších úloh může dělat potíže, ale ty při využití vzorů z příkladů nemusí být enormní. Naproti tomu značnou výhodou je možnost vytvoření dokumentu nebo prezentace v reálném čase tvorby dokumentu. To má cenu u úloh, které by se neměly tvořit přetržitě. Dobrým příkladem jsou protokoly, u nichž lze po odladění a ověření vzorového zadání garantovat věrohodnost výsledku např. při akreditacích.

Celý systém se aktivuje po deklaraci knihovny pro vytváření diagramů jediným povelem `\usepackage{diagram}` před zahájením dokumentu. Tento balík maker si sám zavede další styly, jako jsou `fp.sty` [4] a jeho rozšíření `fp-contrib.sty` pro výpočty v pevné řádové čárce, `curves.sty` [2] pro grafické práce s křivkami a `support.sty` s řadou užitečných maker, které sám využívá a které jsou k dispozici i pokročilému uživateli. V balíku `diagram.sty` jsou jak makra rozšiřující možnosti poskytované stylem `curves.sty`, tak i makra pro vytváření složitých grafických papírů a diagramů. Je pamatováno i na prezentace, při nichž lze postup vytváření diagramu krokovat, což prezentační programy umožňují.

Popis systému je doprovázen řadou jednoduchých ukázek, které mohou být i vodítkem pro uživatele při vytváření vlastních grafů. Použití systému pro složitější úlohu je ukázáno na zpracování měření únavové životnosti materiálů do protokolu. Protože jde o dosti náročnou úlohu, je rozdělena do etap, které tvoří rozsáhlá makra, mezi kterými je zřetelná dělba práce na výpočet, postupné vytváření diagramu s možnou intervencí uživatele pro prezentace, vynášení složitých závislostí a nakonec i tisk tabulky výsledků. Takto specializovaná makra jsou zahrnuta do stylu `SNcurv.sty`. Soubory všech knihoven jsou k dispozici jako příloha tohoto čísla Zpravodaje ČSTUGu na `http://bulletin.cstug.cz/`.

Při sestavování nových povelů se v LaTeXu užívá příkaz `\newcommand` a pro změnu existujícího povelu příkaz `\renewcommand`. Je však zapotřebí méně zkušeného čtenáře upozornit na nebezpečí jejich používání

pro definování argumentů jiného makra. Oba zmíněné příkazy totiž definují *makro*, tedy podprogram, který se při vyvolání vždy znovu vyhodnocuje! Je-li argumentem takto vytvořeného makra nějaký pracovní povel, který se může v průběhu výstavby dokumentu měnit, bude se měnit i hodnota příkazů definovaných v LaTeXu výše uvedeným standardním způsobem podle jeho aktuálního obsahu. Tato chyba se velmi špatně hledá, a proto se v tomto dokumentu preferují pro přiřazení příkazy `\edef, \xdef, \FPset` nebo `\Cmd`. Ty totiž argument napřed vyhodnotí a výsledek tohoto rozvoje teprve přiřadí. Vzniklý příkaz má potom konstantní hodnotu, která platila v době jeho definice.

Makra, jejichž realizace může trvat déle, jsou v nových knihovnách maker `diagram.sty` a `SNcurv.sty` vybavena operativními výstupy názvu makra na obrazovku a do souboru `*.log`. Tato informace jednak ukazuje místo, ve kterém se převod zdroje dokumentu právě nachází, jednak dává uživateli klid, že systém „nezamrznul“. Tímto hlášením nejsou vybavena všechna makra, ale jen potenciálně časově náročná.

Platí-li Murphyho zákon o programování říkající, že v každém programu je alespoň jedna chyba, potom ani předkládané soubory maker nebudou výjimkou, třebaže jejich sestavení byla věnována maximální péče. Prosím proto laskavé čtenáře o shovívavost a o sdělení nalezených chyb a slabin na moji dále uvedenou adresu. Doufám jen, že jich nebude mnoho a že všechna makra pomohou budoucím uživatelům zrychlit práci na jejich dokumentech s diagramy.

Závěrem chci poděkovat všem autorům publikací o TeXu a LaTeXu, zejména pak TeXbook naruby [8], LaTeXové kuchařky [9] a LaTeX pro začátečníky [10] za popularizaci a předávání cenného know-how. Můj dík patří i panu Petru Aubrechtovi za řadu cenných připomínek k rukopisu.

Summary: Calculations and diagrams in LaTeX

The article deals with the title problem from the point of view of a common user of LaTeX. It describes a way of using standard packages `fp.sty` and `curves.sty`, and their new extensions `fp-contrib.sty` and `diagram.sty` with an auxiliary package `support.sty`. The complex allows to solve rather complicated tasks in one run of the LaTeX compiler. A solution of processing fatigue data into SN-curve, bands of confidence

intervals, plots and a table of results is presented as an example. The system is suitable also for presentation purposes.

Key words: LaTeX, calculations, diagrams

Reference

[1] M. Balda: Uživatelské nadstavby sázecího systému LaTeX 2.09 Západočeská univerzita, Fakulta aplikovaných věd, Plzeň, 1997; viz i `http://www.cdm.cas.cz` pod publikacemi autora.

[2] Ian Maclaine-cross: Curves in LaTeX Pictures. A manual for `curves.sty` and `curvesls.sty`. Sydney, 2000, In: A subdirectory `texmf\source\latex\curves`

[3] F. Buchholz: `realcalc.tex` – Real arithmetic with big values and high precision. DANTE, 1993

[4] M. Mehlich: Fixed point arithmetic for TeX – `fp-package`, 1996, In: A subdirectory `texmf\source\latex\fp`

[5] ČSN 42 0368: Zkoušení kovů. Zkoušky únavy kovů. Statistické vyhodnocování výsledků zkoušek únavy kovů. Československá státní norma, Vyd. ÚNM, Praha, 1974

[6] J. Antoch, D. Vorlíčková: Vybrané metody statistické analýzy dat. Academia, Praha, 1992

[7] K. Rektorys & kol.: Přehled užité matematiky. SNTL, Praha, 1988

[8] P. Olšák: TeXbook naruby. Konvoj, Brno, 1997

[9] Z. Wagner: LaTeXová kuchařka. Zpravodaj CSTUG. /1 (2/1996), /2 (4/1996), /3 (3-4/1997), /4 (2/1998), /5 (3/1999)

[10] J. Rybička: LaTeX pro začátečníky, 3. vyd.. Konvoj & CSTUG, Brno, 2003

Miroslav Balda
Veleslavínova 11
301 14 Plzeň
`balda@cdm.it.cas.cz`

TUGboat 22(4), December 2001

TUGboat 23(1), 2002 — TUG 2002 proceedings

TUGboat 23(2), 2002

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
Počet výtisků:	650
Uzávěrka:	22. října 2000
Odpovědný redaktor:	Zdeněk Wagner
Redakční rada:	Petr Aubrecht, Matěj Cepl, Jiří Demel, Jana Chlebíková, Jiří Kosek, Jaromír Kuben, Petr Sojka, Martin Tkadlčík
Tisk a distribuce:	KONVOJ, spol. s r. o., Berkova 22, 612 00 Brno, tel. +420 549 240 233
Adresa:	CSTUG, c/o FEL ČVUT, Technická 2, 166 27 Praha 6
Tel:	+420 224 353 611
Fax:	+420 233 332 938
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 CD-ROM

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

Uzávěrka příštího čísla: 9. prosince 2004