Kouzla s programem MNU

Do nového balíku $\mathcal{CS}$TEX byl zahrnut program MNU, který řídí režim nabídek. Nabídka je za pomocí tohoto programu plně konfigurovatelná. V rámci „reklamací a připomínek“ k $\mathcal{CS}$TEXu zaznělo několik návrhů na mírnou úpravu konfigurace nabídek. Rozhodl jsem se žádnému z těchto návrhů nevyhovět a do inovované verze změny v této oblasti nezanášet. Místo toho chci zde v tomto článku ukázat, jak se takové změny dělají. Každý si je může udělat sám podle svých představ. Začnu nejprve popisem vývoje (nebo spíše „nevývoje“) vlastního programu MNU.

Zhruba před dvěma lety jsem při pohledu na nabídkový program, který byl tehdy zahrnutý do instalace TEXu, došel k závěru, že by stálo za to napsat něco jiného. Nejprve jsem si zformuloval, co vše by program měl umět – tj. napsal jsem dokumentaci k neexistujícímu programu.[1] Pak jsem věc konzultoval s tehdejším výborem $\mathcal{CS}$TUGu a po předběžném souhlasu, že by takový program mohl být užitečný, jsem se pustil do díla. Tak vznikl program MNU. S první verzí jsem byl hotov do začátku EuroTEXu-92, kde jsem ji též zveřejnil. Pak jsem ještě zpracoval některé návrhy a připomínky a výsledkem byla verze z ledna r. 1993, kterou už považuji za definitivní. Tato verze je zařazena do balíku $\mathcal{CS}$TEX.

Za celý rok provozu programu MNU v nové instalaci $\mathcal{CS}$TEXu jsem nezaznamenal k tomuto programu jedinou připomínku nebo reklamaci a to mě vede k rozhodnutí dále program pokud možno neměnit. Můj názor je, že velkým přínosem každého programu je jeho stálost (viz například program TEX), protože uživatel se nemusí topit v džungli nejrůznějších čísel verzí a být neustále ve střehu, zda nepřišel náhodou o nejnovější verzi a zda ty nové verze ještě bude umět používat.

Samozřejmě svědomí mi nedovolí nechat v programu objevené chyby. Ačkoli jsem nedostal zprávu o nalezení chyby od žádného uživatele,

[1] Musím přiznat, že tento obrácený postup – nejprve dokumentace a potom teprve program, se mi osvědčil i při tvorbě jiného software a všem jej vřele doporučuji. Je-li totiž jasně zformulovaná myšlenka, je pak další programovací práce většinou záležitostí pro „cvičenou programovací opici“.

sám jsem za ten rok jednu chybu objevil. Jedná se o nesprávnou lokalizaci aktivní editované položky v případě, že se tato položka nalézá v okně bez rámu. Při opravě této chyby jsem program obohatil o jednu maličkost – schopnost provést definovanou akci po daném časovém limitu „nic-nedělání“ (dá se nakonfigurovat například zhasnutí obrazovky po daném čase). Inovovaný program jsem zařadil do $\mathcal{CS}$TEXu 94 a věřím, že to byly skutečně poslední úpravy programu.

Program MNU má na rozdíl od programu TEX omezenou životnost. Program totiž ryze účelově rozšiřuje možnosti „dávkového programování“ pod operačním systémem DOS. Ačkoli je DOS velmi rozšířen, nevěštím mu dlouhou budoucnost. Větší vyhlídky do budoucna mají systémy, v jejichž jádru je řešena problematika paralelního zpracování procesů.[2] V současné době jsou už i stolní počítače pro tyto účely dostatečně výkonné. Pro uživatele TEXu jsou systémy s více procesy přímo nutností, neboť vypisovat si poznámky na papír v okamžiku, kdy mám na obrazovce „View“, abych mohl tyto poznámky použít hned, jakmile se vrátím do editoru, to je metoda, které se budeme zanedlouho (doufám) pouze smát. V DOSu přitom není příliš mnoho jiných možností.[3]

Dnes bych program MNU asi naprogramoval jinak a možná lépe (například jazyk konfiguračního souboru by nebyl tak kryptický), ale z důvodů uvedených výše nebudu program dále rozvíjet. DOS ale zatím u nás vládne, a proto si myslím, že některé triky, které se dají s programem MNU a s DOSem provádět a které uvádím v tomto článku, může ještě plno lidí využít.

Program MNU je „in public domain“. Můžete jej získat ve dvou podobách (v obou případech se jedná o naprosto stejný program). První možností je získat jej jako samostatný balík (ftp z archívu `ftp.tex.ac.uk`, adresář `support`). Zde najdete anglickou dokumentaci a dva příklady použití programu. Oba příklady pracují pouze „teoreticky“, tj. v místě volání zvoleného programu z nabídky se ve skutečnosti neus-

[2] Nemám na mysli DOSovskou nadstavbu WINDOWS. Ta není příliš vydařeným pokusem dát DOSu náplast ve formě vize z paralelního zpracování procesů. Přitom zůstává jen u té vize a skutek utek'.

[3] Ani rezidentní editory typu Side Kick apod. příliš nepomohou. Jsou s nimi spíš problémy – motají grafiku prohlížeče a paměť počítače a jejich užitečnost je sporná.

kuteční nic. První příklad je jednoduchý a druhý demonstruje konfiguraci $\mathcal{CS}$TEXu.

Program můžete také získat jako součást $\mathcal{CS}$TEXu. Naleznete zde českou dokumentaci k programu (v souboru `\doc\programs\mnu.tex`) a rovněž tu najdete dva příklady – v tomto případě ovšem plně funkční. Prvním příkladem je samotná konfigurace nabídek v $\mathcal{CS}$TEXu a druhým instalační program umožňující pohodlnou instalaci $\mathcal{CS}$TEXu z disket. Dokumentace k prvnímu příkladu je uložena v souboru `\doc\cfg.doc` a k druhému příkladu v souboru `install.doc` na první instalační disketě. Nechci zde opakovat, co je řečeno v dokumentaci. Proto se raději věnujme jednotlivým ukázkám. Začneme tím nejjednodušším.

Jak přidat novou položku do systému nabídek?

Z veřejných archívů jsme například získali program GhostScript a chceme ho zařadit do systému nabídek $\mathcal{CS}$TEXu tak, aby se jeho vyvoláním spustila grafická interpretace PostScriptového souboru na obrazovce. Položku nazveme třeba `GS-View` a zařadíme ji do okénka `Others`. Nejprve provedeme změny v souboru `\cfg\ram\cfg.mnu`:

```
^131  132  133  134  135  136  [37]  ; ******* Others *******
~param {31, 16, 14, 8}
| DVI-O^ut    |
| DVIP^S      |
| MakeI^ndex  |
| B^ibTeX     |
| T^eXCad     |
| G^S-View    |
```

Pro položku `GS-View` jsme zvolili číslo 136 (viz první řádek) a výšku okénka `Others` jsme o jedničku zvětšili (viz druhý řádek, číslo 8). Text nové položky je napsán na posledním řádku. Volba čísla položky je trošku problematická záležitost. Je třeba nejprve zjistit, zda zvolené číslo nekoliduje s již existujícími položkami – seznam všech položek najdeme v souboru `cfg.doc`. Pokud chceme s číslem položky pracovat v řídicí dávce prostřednictvím error-kódu, jsme navíc omezeni číslem 255. Jistou nevýhodou programu MNU tedy je, že nelze pro položky použít symbolických názvů.

Při vyvolání programu MNU na změněný soubor `cfg.mnu` se změní okénko `Others`. Především je o jeden řádek větší a v tomto řádku se nabízí položka `GS-View`. Po jejím potvrzení vrátí program MNU dávce error-kód číslo 136. Pokud jsme v řídicích dávkách neudělali žádné změny, vyvolá se program stejný, jako při volbě položky `TeXCad` (error-kód nejbližší nižší; zde č. 135). K tomu, abychom spustili jiný program, potřebujeme ještě editovat dávku `\cfg\others.bat`:

```
if errorlevel 136 goto gsview
if errorlevel 135 goto texcad
if errorlevel 134 goto bibtex
...
:gsview
  %TEXDIR%\gs -I%TEXDIR%\gslib %MAIN%.ps > gs.mes
  goto end
```

V této dávce jsme přidali skok na návěští `:gsview` (první řádek dávky) a pak jsme přidali řádky, týkající se našeho nového programu. Zde se konkrétně předpokládá, že program `gs.exe` je uložen v adresáři `%TEXDIR%`, tj. v „hlavním" adresáři $\mathcal{CS}$TeXu. Knihovní soubory pro program jsou v podadresáři `gslib` a vstupem pro program je soubor s názvem hlavního souboru a s příponou `ps`. Přesměrování textového výstupu je provedeno proto, že program jinak hloupě míchá grafický výstup se svými textovými poznámkami a je to rušivé.

V našem příkladě jsme využili toho, že v hlavní řídicí dávce `texbat.bat` je zaneseno vyvolání dávky `others.bat` v případě, že je error-kód v intervalu $\langle 131, 140\rangle$ (viz příkaz `if errorlevel 131 goto others` v dávce `texbat.bat`). V případě, že jsme zvolili jiné číslo položky, musíme většinou editovat přímo dávku `texbat.bat`.

Přidáme nové okénko

Rozhodli jsme se například usnadnit uživatelům „útěk" z nabídkového systému přidáním položky `Quit` do nejčastěji používaného okénka `TeX`. Uživatel nyní může rovnou odejít zmáčknutím klávesy Q. Aby to ale neměl tak jednoduché, nechť se po volbě této položky zobrazí nové okénko, které nabídne dvě možnosti – buď odejít z TeXovského systému za současného smazání „nepotřebných" souborů, (tj. `dvi`, `bak`, `aux` apod.), nebo odejít bez mazání těchto souborů. V souboru `cfg.mnu` provedeme úpravy následujícím způsobem (srovnejte s originálním obsahem souboru).

```
^31, 32, 33, 34, 34, 35, 36, 37, 38, 39, 30, 330 [3] <21, 41>; *TeX menu*
~param  {19, 7, 15, 16}
| E^dit        |
| [%FFMT%]        |
| V^iew        |
...
| C^StoCS...    |{141}
| Q^uit...      |{331}

^331 332 [330] ; **** Quit menu ****
~param  {31, 20, 23, 4}
| Q^uit plus clear dvi |{$25}
| Quit -- L^eave dvi   |{$29}
```

Na prvním řádku ukázky jsme přidali číslo 330, což bude číslo nové položky. Na druhém řádku jsme zvětšili výšku okénka `TeX` na 16 a dále jsme přidali text nové položky, viz řádek `Q^uit`. Na tomto řádku je vidět, že volba položky `Quit` neukončí systém nabídek, ale způsobí otevření okna s položkou 331, přičemž tato položka bude aktivní.

Okénko s položkou 331 jsme definovali na posledních čtyřech řádcích ukázky. Nejprve jsme napsali řádek s čísly položek: 331 a 332. Na tomto řádku je ještě v hranaté závorce číslo položky, kam se program vrátí v případě stisku klávesy `Esc`. V našem případě se jedná o položku, z níž je naše okénko vyvoláno. To má svoji logiku, ovšem dají se udělat věci, které logiku nemají.

Nové okénko obsahuje dvě položky s textem `Quit plus clear dvi` a `Quit -- Leave dvi`. První položka vrací kód 25 (viz `$25`), což zatím dávka `texbat.bat` neumí správně zpracovat a budeme se tím za chvíli zabývat. Volbou druhé položky se ukončí program MNU s kódem 29, což je kód, který odpovídá požadavku ukončení systému nabídek (viz soubor `texbat.bat`).

Poznamenejme ještě, že je velice výhodné měnit konfigurační soubor MNU jakoby rovnou ze systému nabídek voláním editoru. Po ukončení editoru okamžitě vidíme všechny spáchané změny, takže výsledný vzhled a umístění okének můžeme průběžně „ladit“. Takové úpravy musíme ale dělat v kopii souboru `cfg.mnu`, kterou obvykle najdeme na RAM disku. Právě tato kopie je totiž „v akci“. Po ukončení práce pak nesmíme zapomenout naše změny uložit do souboru na fyzickém disku, protože na RAM disku nám po vypnutí počítače nebudou příliš platné. Navíc, pokud jsme konfigurovali $\mathcal{CS}$TEX v době instalace tak, že po ukončení systému nabídek se vymažou pomocné soubory z RAM disku, musíme být nanejvýš opatrní a systém nabídek neukončovat před uložením změn na disk.

Nyní ukončíme naši práci editováním souboru `texbat.bat`. Změny mohou vypadat následovně:

```
...
if errorlevel  26 goto shell
if errorlevel  25 goto clearq
if errorlevel  24 goto clear
...
:clearq
  for %%p in (%CLFILES%) do del %MAIN%.%%p
  if exist %WORK%.bak del %WORK%.bak
  goto quit
```

V této ukázce jsme zavedli mezi příkazy `if errorlevel` odskok na návěští `clearq` při kódu 25. V místě tohoto návěští je realizováno mazání souborů s příponami obsaženými v proměnné `CLFILES`. Je vhodné do inicializačního souboru `texset.bat` zařadit naplnění této proměnné třeba takto:

```
set CLFILES=dvi bak aux
```

a zařadit obsah proměnné `CLFILES` do okna `Envir`, aby si rozsah a množství mazaných souborů mohl uživatel sám nastavit. Výšku okénka `Envir` je proto nutné zvětšit o jeden řádek a přidat tam editovatelnou položku. Jak se to dělá, nebudeme znova opakovat.

Poznamenejme ještě, že chceme-li být důslední, měli bychom se zabývat i texty helpů pro naše nové položky. Například pro všechny tři nové položky 330, 331 a 332 připravíme společný text helpu a zařadíme jej někam na konec souboru `mnu.cfg`. Takový text může vypadat následovně.

```
^(330, 331, 332)
 Quit plus clear dvi
 ===================
 Po volbě položky Quit budete ještě tázáni, zda si přejete smazat
 nepotřebné soubory, či nikoli. Za nepotřebné se považují soubory
 [%MAIN%] s příponami: [%CLFILES%].
 Množinu těchto přípon lze měnit pomocí | environment proměnné |{\1163}
 CLFILES.
```

Poznamenejme ještě, že zde máme „proměnlivý" obsah helpu v závislosti na obsahu proměnné `CLFILES` a `MAIN`. To je celkem sympatická vlastnost programu MNU. Také zde vidíte odkaz na další okno helpu s položkou 1163.

Barevná nebo černobílá nabídka

Smyslem tohoto článku není opakovat to, co bylo řečeno v dokumentaci k programu MNU nebo co je předvedeno ve stávající konfiguraci CSTEXu. Více bych se chtěl zaměřit na triky, které možná nejsou při povrchním přečtení dokumentace a příručky DOSu zcela patrné. Tento článek bych přirovnal ke kapitole „Dirty Tricks" v *TEXbook*u. Jen jsem opomněl vložit na toto místo příslušné množství „dangerous bendů".

Zajímavým příkladem může být vlastnost nabídky „dynamicky" měnit svůj vzhled v závislosti na provedené volbě. Třeba položka `TeX` v okně `TeX` má vzhled závislý na zrovna zvoleném formátu (tj. `PlainTeX`, `LaTeX` apod.). Je to zařízeno tak, že v místě textu položky je požadavek substituce textu ve tvaru `[%...]`. V tomto případě se text nahrazuje hodnotou proměnné DOSu.

Další možností je nabídnout programu MNU různé konfigurační soubory. Tuto možnost si předvedeme podrobněji. V souboru `cfg.zip` na první instalační disketě najdete dva skoro stejné konfigurační soubory. Jeden se jmenuje `cfg.mnu` a konfiguruje barevnou nabídku pro CSTEX, zatímco druhý se jmenuje `cfgmono.mnu` a konfiguruje černobílou (ale jinak stejnou) nabídku. V instalačním programu je otázka, zda se má nabídka chovat barevně nebo černobíle a na základě toho se jeden z těchto dvou souborů použije jako definitivní.

Zkusme použít oba soubory naráz. Přidejme si do prvního souboru na nějaké vhodné místo položku `Mono` ve tvaru

```
| Mono  |{"set CFGMNU=cfgmono.mnu">2}
```

V řídicí dávce nahraďme ve volání programu MNU název konfiguračního souboru hodnotou proměnné `CFGMNU` takto:

```
%RAM%\mnu %RAM%\%CFGMNU% %RAM%\envir.bat %RAM%\dos.bat %RAM%\mfbat.bat
call %RAM%\envir
```

a postarejme se o to, aby po volbě položky `Mono` dávka pouze načetla soubor `envir.bat` a dále se vrátila na volání menu. Pokud v souboru `cfgmono.mnu` vytvoříme analogickou položku s názvem `Color`, můžeme si celý zázrak vyzkoušet. Nesmíme ale zapomenout v souboru `texset.bat` definovat implicitní hodnotu proměnné `CFGMNU` například jako `cfg.mnu`.

Po startu nám naskočí barevná nabídka obsahující položku `Mono`. Po volbě této položky se ihned celá nabídka stane pouze černobílou a bude obsahovat položku `Color`. Po volbě této položky zase dostane nabídka barvu.

Tento příklad s barvami není příliš efektivní. Potřebujeme mít totiž v akci dva skoro stejné soubory. Navíc, pokud bychom měnili strukturu nabídek, musíme tak učinit dvojmo. Asi lepší řešení je použít pouze v místech, kde se zmíněné dva soubory liší, substituční konstrukce typu `[%...]`, přitom zde bude asi vhodné substituovat krátkými soubory.

Podobně můžeme střídat třeba jazyky. Vytvořme si soubor s názvem `cfgcz.mnu`, který bude překladem souboru `cfg.mnu` do češtiny. Zařadíme si na vhodná místa položky `Anglicky` a `Czech` a můžeme se bavit tím, jak bude uživatel zmaten, když se texty všech položek jedním rázem změní.

Uživatelské konfigurační soubory trochu jinak

Zaměřme se na problém uživatelských konfiguračních souborů, které mají v CSTeXu název `texcfg.bat`. Příkazy uvedené v takovém souboru se provedou na konci inicializační dávky `texset.bat`. Například příkazy typu `set` v konfiguračním souboru mají vyšší prioritu, než tytéž příkazy `set` nastavující „společnou konfiguraci“ v inicializační dávce. Současný návrh systému nabídek umožňuje sice automaticky vytvořit soubor `texcfg.bat`, ale ukládají se do něj *všechny* hodnoty proměnných DOSu. To má plno nepříjemných důsledků. Například pokud uživatel zkopíruje obsah svého adresáře do jiného počítače (včetně souboru `texcfg.bat`), pak se může stát, že bude mít nepředstavitelné problémy. V tomto jiném počítači může například být jinak nastavena hodnota `%COMSPEC%`, nebo třeba vlastní TeX může být v jiném adresáři, takže se liší hodnota proměnné `TEXDIR`. Také se v souboru `texcfg.bat` ukládá hodnota proměnné `MAIN` a `WORK` a uživatel se pak může strašně divit, že hodnoty těchto proměnných přestávají být závislé na obsahu příkazového řádku, kterým je nabídkový systém vyvolán.

Pokusíme se předvést možnost, jak automaticky vytvářet uživatelské konfigurační soubory, které obsahují nastavení *pouze některých* zvolených proměnných DOSu. Chtěl bych poznamenat, že při vývoji nabídkového systému pro $\mathcal{CS}$TEX jsem měl toto řešení původně navrženo, ale pak jsem od něj ustoupil, protože tyto „statické“ soubory `texcfg.bat` samozřejmě nebudou obsahovat důležité hodnoty proměnných DOSu, které souvisejí s novými programy nezařazenými do původního balíku.

Podívejme se nejprve, jakým způsobem se soubor `texcfg.bat` vytváří ve stávající instalaci $\mathcal{CS}$TEXu. Při volbě položky `Save` se v dávce `texbat.bat` provedou následující příkazy:

```
:save
  set > %RAM%\file.mnu
  echo echo **** CONFIGURATION from file %MARK%.bat **** > %MARK%.bat
  %TEXDIR%\dupcent %RAM%\file.mnu set >> %MARK%.bat
  echo set MNU=31 >> %MARK%.bat
  del %RAM%\file.mnu
  set mnu=65
  goto menu
```

Zkusme si to vysvětlit. Nejprve se uloží výpis všech proměnných DOSu ve formátu `NAZEV=hodnota` do pomocného souboru `file.mnu`. Pak se začne vytvářet vlastní soubor `%MARK%.bat`. První řádek tohoto souboru bude obsahovat text

```
echo **** CONFIGURATION from file texcfg.bat ****
```

a další řádky budou obsahovat postupně všechny použité proměnné DOSu ve formátu `set NAZEV=hodnota`. Je zde použit program `dupcent.exe`, který čte pracovní soubor `file.mnu` a před každý řádek přidá slovo `set` a navíc duplikuje případná procenta uvnitř hodnot proměnných DOSu. Poslední řádek v souboru `%MARK%.bat` bude obsahovat příkaz `set MNU=31`, tj. systém nabídek po zpětném načtení tohoto souboru naskočí na položku s číslem 31, což je položka `Edit`.

Pokud změníme výše uvedené příkazy v dávce `texbat.bat` následujícím způsobem, dostáváme „statické“ konfigurační soubory:

```
:save
  %RAM%\mnu %TEXDIR%\cfg\maskcfg.mnu > %RAM%\file.mnu
  %TEXDIR%\dupcent %RAM%\file.mnu > %MARK%.bat
  del %RAM%\file.mnu
  set mnu=65
  goto menu
```

Program MNU je zde použit jako filtr, který čte soubor `maskcfg.mnu` a na základě něj vytváří pracovní soubor `file.mnu`. V tomto pracovním souboru ještě zdvojíme procenta programem `dupcent.exe` (tentokrát nepřidáváme před řádky slovo `set`) a výsledek uložíme do souboru s názvem `%MARK%.bat`. Obsah výchozího souboru `maskcfg.mnu` může být třeba následující:

```
***** Mask for configuration files *.bat ****
~copy
echo **** CONFIGURATION from file [%MARK%].bat ****
set EDIT=[%EDIT%]
set EMTEXED=[%EMTEXED%]
set SHELL=[%SHELL%]
...
set MFINPUT=[%MFINPUT%]
```

Příkaz `~copy` nastavuje program MNU do „kopírovacího režimu“ (viz dokumentace `mnu.tex`, str. 6 dole), tj. program lze použít jako filtr, přičemž se aplikuje náhrada textu podle pravidel pro konfigurační soubory programu MNU. Výstupem z tohoto filtru bude soubor třeba s tímto obsahem (to samozřejmě závisí na aktuálním stavu proměnných DOSu):

```
echo **** CONFIGURATION from file texcfg.bat ****
set EDIT=call e:\textmp\qedit
set EMTEXED=call e:\textmp\qedit %2 %3 -n%1
set SHELL=c:\nc\nc
...
set MFINPUT=c:\emtex\mfinput;.
```

Program `dupcent` ještě zdvojí procenta, takže ve výsledném souboru `texcfg.bat` máme např. řádek:

```
set EMTEXED=call e:\textmp\qedit %%2 %%3 -n%%1
```

Pokud se někdy zpětně vyvolá soubor `texcfg.bat`, pak se zdvojená procenta interpretují jako jedno „obyčejné“ procento a vše bude pracovat tak, jak chceme.

Je tedy vidět, že obsah uživatelských konfiguračních souborů typu `texcfg.bat` můžeme mít zcela pod kontrolou. Tento obsah určíme pomocí souborů typu `maskcfg.mnu`. Můžeme mít samozřejmě takových

„maskových souborů“ více; například roztřídíme proměnné DOSu do tématických okruhů. Nabídku týkající se možnosti uložení hodnot proměnných do souboru pak musíme příslušným způsobem rozšířit. Uživatel si může vybrat, zda uloží třeba stav proměnných týkajících se editoru a shellu nebo stav proměnných týkajících se třeba TEXu, ovladačů, nebo názvů pracovního a hlavního souboru.

Staré menu pomocí nového MNU

Následující příklad má dokumentovat flexibilitu programu MNU. Ukážeme, že starý známý systém nabídek z předchozích verzí CSTEXu lze pomocí programu MNU snadno simulovat. Užitečnost tohoto příkladu se může zdát sporná. Je třeba si ale uvědomit, že někdy je velmi nebezpečné dát lidem k dispozici jiný systém nabídek, třebaže má více možností. Už jiný vzhled systému nabídek na obrazovce dokáže mnohé uživatele odradit. Lidé se začnou rozčilovat, že se musí učit *zase* něco jiného. Proto například O. Ulrych raději poněkud vylepšil staré menu, než aby uživatelům na jeho pracovišti nabídl nový systém nabídek z CSTEXu. Chápu ho – uživatelé jsou totiž velmi konzervativní.

Pokud si vzpomínám, tak staré menu nabízelo vpravo od názvu položky jakési editační pole, kde se psaly parametry pro jednotlivé programy. Myslím si, že tomu nejvíce odpovídá následující konfigurační soubor (označme ho např. `oldmenu.mnu`).

```
******* Staré menu pomocí MNU *******
~start {!>2}
~final {"set MNU=#">2}
^1 2 3 4 5 6 7 8 9
~param {30, 1, 50, 15, 1, 7, 7, 7, 112, 112}
                CSTeX - menu
================================================
▷ Edit   ◁ |#[%EDITPAR%37]|{"set EDITPAR=#">%2}
▷ TeX    ◁ |#[%TEXPAR%37]|{"set TEXPAR=#">%2}
▷ View   ◁ |#[%VIEWPAR%37]|{"set VIEWPAR=#">%2}
================================================
▷ Vlnka  ◁ |#[%TIEPAR%37]|{"set TIEPAR=#">%2}
▷ Spell  ◁ |#[%SPELLPAR%37]|{"set SPELLPAR=#">%2}
▷ Matrix ◁ |#[%MATRPAR%37]|{"set MATRPAR=#">%2}
▷ Laser  ◁ |#[%LASERPAR%37]|{"set LASERPAR=#">%2}
================================================
▷ DOS    ◁ |#[%DOSPAR%37]|{"set DOSPAR=#">%2}
▷ Quit   ◁ |#[%QUITPAR%37]|{"set QUITPAR=#">%2}
```

Symboly ▷ a ◁ zde označují znaky s kódem 16 a 17, což se na obrazovce jeví jako trojúhelníčky. Menu vyvoláme například dávkou `oldtex.bat`, která obsahuje tyto příkazy:

```
@echo off
if '%1==' goto err
%COMSPEC% /e:1300 /c oldmenu %1
goto end
:err
echo Chybí název souboru (bez přípony) jako parametr.
:end
```

Takovou věc asi znáte z $\mathcal{CS}$TEXu. Rozšiřuje se zde prostor pro proměnné DOSu a vyvolá se další dávka, zde nazvaná `oldmenu.bat`. Tato dávka vypadá následovně (místo teček si doplňte analogicky podobné příkazy pro další položky):

```
@echo off
set EDITPAR=%1.tex
set TEXPAR=%1.tex
set VIEWPAR=@scr.cnf %1.dvi
...
set QUITPAR=echo ahoj!
set MNU=1

:menu
mnu oldmenu.mnu envir.bat
call envir
if errorlevel 9 goto quit
if errorlevel 8 goto dos
...
if errorlevel 1 goto edit
pause > nul
goto menu

:edit
  call edit %EDITPAR%
  goto menu
  set MNU=2
:tex
  ...
:dos
  %DOSPAR%
  goto menu
:quit
  %QUITPAR%
```

Poznamenejme, že zde nejsou řešeny zdaleka všechny problémy související s TEXem. Například zde nenajdete nastavení některých proměnných DOSu, které jsou typické pro emTEX. Berte to proto jako příklad konfigurace nabídky. Pokud ji budete chtít skutečně použít, bude potřeba ji v některých ohledech upravit.

Nyní si shrneme, jak se takto konfigurovaný systém nabídek chová. Vidíme, že všechny položky v menu mají typ editovatelné položky. Pokud se zmáčkne `Enter` (nezáleží na tom, zda se editovalo či nikoli), potvrdí se hodnota položky (tj. parametry volaného programu) a navíc se požadovaný program vyvolá. Položku volíme šipkami a jakmile začneme psát, ukládá se text v editačním poli do parametrů programu. Klávesa `Esc` má dva významy. Pokud se už začalo editovat, pak vrací editační pole do původního stavu, jinak skrývá celé okénko s nabídkou. V tomto případě totiž program MNU ukončí činnost s kódem 0 a v dávce `oldmenu.bat` se provede příkaz `pause > nul`, tj., čeká se na libovolnou klávesu, aby se proces mohl vrátit zpět do nabídky. Myslím, že tak nějak to v tom starém menu fungovalo.

Nevýhoda takovéto konstrukce nabídek je v tom, že nám nedovoluje používat zvýrazněná písmena v textu položek pro rychlý výběr položky. Jakmile totiž zmáčkneme písmeno, nabídkový systém to interpretuje jako požadavek na změnu parametrů programu a odstartuje editaci v editačním poli. Nevzpomínám si, že by ve starém menu šlo volit položku rovnou písmenem, takže se domnívám, že to pracuje přesně tak, jak byli uživatelé starého menu zvyklí. Jednu věc programem MNU simulovat nelze. Jde o to, že při startu editace se celé editační pole zaplnilo jakýmisi nesmyslnými znaky vypadajícími přibližně takto: ≪. Tato skutečnost ale spíše rušila a mátla, takže z toho, že to nelze simulovat, nemusíme být smutní.

Mezi znaky, které by se neměly objevit v textu parametrů patří `|`, `=`, `<` a `>`. Tyto znaky totiž mají v DOSu speciální význam a z těchto důvodů jimi nelze naplnit proměnnou DOSu. Tuto věc můžeme obejít použitím pomocných souborů. Stačí se inspirovat řešením položky DOS v novém systému nabídek CSTEXu.

Namítnete ještě, že čáry oddělující jednotlivá pole v původním programu pěkně navazovaly na vnější rámeček. Pokud chcete, můžete toho pomocí MNU snadno dosáhnout. Stačí volit v příkazu `~param` velikost rámečku rovnou nule a rámeček si do konfiguračního souboru `oldmenu.mnu` zanést explicitně pomocí semigrafických symbolů včetně všech spojujících mezičar. Pokusíte-li se ale o spojení dvojité čáry s jednoduchou, nebude vám to fungovat v kódování PC Latin2, protože tam požadované semigrafické znaky nejsou.

Od jednoduchých formulářů k databázím

Uděláme si „uživatelské rozhraní“ pomocí programu MNU, které umožní snadno vkládat údaje, které je třeba často vyplňovat do stanoveného formuláře. Vlastní formulář nakonec vysází TEX. Uživatel přitom vůbec nemusí vědět, že používá TEX.

Nechť je celá procedura přípravy formuláře vyvolána z DOSu dávkou `form.bat`. Ta bude obsahovat pouze řádek, kterým se zvětší prostor pro proměnné DOSu a vyvolá se řídicí dávka (podobně, jako v dávce `oldtex.bat` z předchozího odstavce). Řídicí dávka se může jmenovat třeba `formbat.bat` a její obsah je následující:

```
@echo off
keybcs
set JMENO=Ferdinand
set PRIJMENI=Mravenec
set ULICE=Ondřeje Sekory 13
set POVOLANI=Práce všeho druhu
:menu
  set MNU=1
  mnu form.mnu envir.bat
  if errorlevel 1 goto quit
  call envir
  mnu masktex.mnu > form.tex
  call texrun form.tex
  call view form.dvi
  set MNU=10
  mnu form.mnu envir.bat
  if errorlevel 2 goto menu
  if errorlevel 1 goto quit
  call print form.dvi
:quit
  del envir.bat
  keybcs -u
```

Popišme si její funkci. Nejprve se vyvolá ovladač pro českou (slovenskou) klávesnici. Předpokládáme, že ovladač obrazovky je řešen globálně (např. v `autoexecu`). Pak se naplní proměnné DOSu, které odpovídají jednotlivým položkám formuláře, jistými „výchozími hodnotami“. Pak se poprvé spustí program MNU s výchozí položkou č. 1. V tomto místě provede uživatel případné změny výchozích hodnot proměnných, které se pak aktualizují voláním přechodné dávky `envir.bat`. Poté se za pomocí programu MNU jako filtru vytvoří zdrojový soubor TEXu s názvem `form.tex`. Výchozí soubor `masktex.mnu` může vypadat třeba takto:

```
~copy
\input makra
\JMENO    {[%JMENO%]}
\PRIJMENI {[%PRIJMENI%]}
\ULICE    {[%ULICE%]}
\POVOLANI {[%POVOLANI%]}
\makeform
```

Pokud například uživatel změnil pouze první dva údaje, pak po provedení příkazu `mnu masktex.mnu > form.tex` dostáváme soubor `form.tex` třeba ve tvaru:

```
\input makra
\JMENO    {Petr}
\PRIJMENI {Olšák}
\ULICE    {Ondřeje Sekory 13}
\POVOLANI {Práce všeho druhu}
\makeform
```

Jak vypadá soubor `makra.tex` zde nebudeme uvádět, protože to závisí na vlastním formuláři a na tom, v jaké podobě ho chceme vytisknout. Pokračujme v trasování dávky `formbat.bat`. Po vytvoření souboru `form.tex` se tento soubor zpracuje TEXem a hned se spustí prohlížeč. Po ukončení prohlížeče se program MNU zeptá uživatele, zda to chce vytisknout nebo opravit (položka 10). Další věci v dávce `formbat.bat` jsou už zřejmé.

Nakonec si uvedeme, jak vypadá soubor `form.mnu`, v němž konfigurujeme vlastní uživatelské rozhraní. Může mít třeba tento obsah:

```
******* Uživatelské rozhraní "formulář" pomocí programu MNU *******
~start {!>2}
^1 2 3 4 5 6
~param {5, 2, 72, 13}
                Formulář pro Velkého Byrokrata
                ==============================
 Následující systém nabídek Vám umožní formulář vyplnit a vytisknout
--------------------------------------------------------------------
```

```
 Jméno:     |#[%JMENO%64]|{    "set JMENO=#">%2,    "#">JMENO,    2}
 Příjmení: |#[%PRIJMENI%64]|{"set PRIJMENI=#">%2,"#">PRIJMENI,3}
 Ulice:     |#[%ULICE%64]|{    "set ULICE=#">%2,    "#">ULICE,    4}
 Povolání: |#[%POVOLANI%64]|{"set POVOLANI=#">%2,"#">POVOLANI,5}

               | OK    |{$0} (Start zpracování)
               | Konec |{$1} (Nechci žádný formulář)
^10
~param {5, 15, 72, 5}
 | Vytisknout |{$0} (Formulář se vytiskne na tiskárně, připravte ji)
 | Vrátit     |{$2} (Návrat do systému nabídek, provedu opravy)
 | Konec      |{$1} (Rozmyslel jsem si to, nechci žádný formulář)

^0
~param {20, 20, 42, 4}
 | Konec |{$1} (Nechci žádný formulář)
 | Dále  |{\[%MNU%]} (Pokračuji)
```

Zde samozřejmě záleží na naši tvořivosti a fantazii, ale také na obsahu skutečného formuláře. Vše můžeme rozvést do více okének a přidat systém nápověd (helpů). Uvedená ukázka je pokud možno co nejjednodušší. Všimneme si některých triků. Za prvé v jediném souboru je řešen vzhled nabídky pro obě volání programu mnu (jak s výchozí položkou 1, tak s položkou 10). Také je zde konfigurována reakce na klávesu `Esc` (položka 0), přitom položka `Dále` způsobí skok na položku 1 nebo 10 v závislosti na současném obsahu proměnné MNU.

Program MNU neumožní editovat položku většího rozsahu, než je jeden řádek. Pokud potřebujeme rozvést některou položku do více řádků, musíme ji buď rozdělit do samostatných řádků, nebo vyvolat na vyplnění takové položky externí editor. První případ použijeme jen výjimečně, protože zde narazíme na omezené editovací možnosti řádkového editoru vestavěného do programu MNU. Stručně řečeno, řádkový editor aplikovaný opakovaně na řádky umístěné pod sebou zdaleka nevytvoří komfort celoobrazovkového editování. Je skutečně lepší pro vyplnění obsáhlých položek vyvolat externí editor. V prvních několika řádcích „předzpracovaného textu“ může být komentář, který například říká, kam se má text napsat a jak se z editoru odchází. Do souboru `form.tex` pak můžeme zařadit vytvořený text buď TEXovským `\input`, nebo jej zařadíme fyzicky v souboru `masktex.mnu` pomocí substituční konstrukce. Program MNU totiž umí expandovat do daného místa obsah jiného souboru.

Nedávno jsem měl možnost vyzkoušet program W94, který vytvořil ve výpočetním centru ČVUT pan ing. Strejc. Program má usnadnit uživatelům vytvoření LaTeXovského souboru, který obsahuje příspěvek do sborníku. Autor programu totiž každoročně formátuje sborník výsledků odborné činnosti na naší škole za pomocí LaTeXu. Nejvíce práce přitom má s vytvořením hlaviček k příspěvkům tak, aby to bylo podřízeno jednotnému formátu sborníku. Navíc mu příspěvky docházejí v rozličných podobách (napsané v různých programech typu `word`-*cokoli* apod.). Rozhodl se proto distribuovat všem přispěvatelům prográmek W94 společně se stylem pro sborník a značně zúženým a jednoúčelově upraveným balíkem TeXu. Prográmek přitom řeší naprosto analogickou věc, jaká byla předvedena zde. Vytváří uživatelsky přítulné rozhraní mezi autorem příspěvku a vlastním LaTeXovým souborem. Nejprve uživatel vloží prostřednictvím okének text názvu příspěvku, jména autora apod. Jedná se vlastně o strukturu hlavičky příspěvku. Potom se vyvolá editor a je třeba napsat vlastní tělo příspěvku. Vše je doprovázeno velkým množstvím příkladů. Z předchozí ukázky vidíme, že by mohl být systém W94 vytvořen pomocí programu MNU.

Komplikovanější je případ, kdy potřebujeme zpracovávat více formulářů současně. Každý formulář je vlastně jeden údaj v jakési databázi. Je nesmyslné, aby se všechny formuláře zpracovávaly naráz v proměnných DOSu – bude nám stačit, když tam budeme mít obsah všech položek jednoho formuláře. V takovém případě už nevystačíme jen s možnostmi programu MNU a DOSu, ale budeme muset vytvořit jednoduché podpůrné prográmky, které umožní například vybrat z databáze jeden formulář a hodnoty jeho položek uložit do proměnných DOSu.

Z mého laického pohledu (s databázemi nemám nic společného) zabezpečuje databázový systém tři druhy služeb. Za prvé vytváří přítulné rozhraní mezi uživatelem, který vkládá nové údaje, a vlastním databázovým souborem. To se dá vesměs řešit programem MNU. Za druhé umožní tisk údajů v nejrůznějších formátech. V tom je zase nepřekonatelný TeX nejen možností definovat jakékoli výstupní formáty pomocí maker, ale samozřejmě též kvalitou tisku. Třetí funkcí databázových systémů jsou nejrůznější třídící a vyhledávací algoritmy, pracující nad databázovým souborem. Aby byly tyto algoritmy dostatečně výkonné, je většinou formát databázového souboru netriviální. Pokud ovšem pracujeme jen s malými databázemi, postačí nám databázový soubor v obyčejném textovém formátu a můžeme si naprogramovat jednoduché vyhledávací prográmky (v Céčku se jedná o pár desítek řádků). Můžeme si tak vytvořit „na míru“ databázový systém za pomocí programu MNU a TEXu. Tento systém sice pracuje pomalu a jen s malými a ne příliš složitými databázemi, zato komfort vkládání údajů je dostatečný a výstup na tiskárnu je bezkonkurenční. Navíc je to celé zadarmo.

Příklad tohoto typu dnes nebudu uvádět. Chystám se ale v nejbližší době řešit problém vyplňování školních vysvědčení za pomocí TEXu. Přitom se využije buď výstup z databázového systému, který má škola už zakoupený, nebo se pro vkládání údajů použije program MNU. Pokud při řešení tohoto úkolu narazím na nějaké zajímavosti, zmíním se o tom někdy příště na stránkách tohoto časopisu.

Okno pro přihlášení uživatele do sítě

Na naší katedře používáme pro připojení do lokální sítě systém PC NFS, který umožňuje přihlášení do sítě pouze z řádky DOSu (příkazem `net init` nebo `net login`. Pomocí programu MNU jsem zvýšil bezpečnost sítě i uživatelský komfort při přihlašování. V této ukázce je plno „špinavých triků“. Kdo nemá takové postupy rád, nechť tento odstavec přeskočí. Také je to ukázka, která nemá nic společného s TEXem. Má pouze dokumentovat další možné využití programu, který se čtenářům dává k dispozici jako součást CSTEXu.

Začneme konfiguračním souborem pro program MNU. Jeho název je `log.mnu` a zde je jeho obsah:

```
******* Login promtp *******
~timeout {20, *"scrblank">0}
~mouse {65535, 0}
^1
~param {5, 2, 72, 16, 1, 112, 30, 30, 14, 14}
     Přihlášení do lokální sítě na katedře matematiky FEL ČVUT
     =======================================================

 Uživatelské jméno (user): |#        |{   "#">USER, 2}

 Heslo (password):

    Po přihlášení do sítě máte k~dispozici lokální disky:
         K: ... s~uživatelskými soubory
         M: ... s~chráněným softwarem
    a na síť připojené tiskárny:
         LPT1 ... jehličková (Epson)
```

```
        LPT2 ... laserová
^2 [1]
~param {26, 8, 20, 1, 0, 0, 0, 0, 0}
|#                  |{*"trylogin [%USER%] #">0}

^10 [1]
~param {20, 14, 40, 7, 2, 78} " !! CHYBA !! " ;

         Chybné přihlášení.
     Údaje nebyly vloženy správně.
          Zmáčkněte  Esc.
^0 [1]
~param {30, 14, 21, 3, 2, 78} " ?? Únik ?? "
   Zmáčkněte Esc
```

Celá procedura se vyvolá dávkou `login.bat` s tímto obsahem:

```
@echo off
call initnet
set MNU=1
:znova
mnu log.mnu
set MNU=10
if not exist OK.bat goto znova
call OK
del OK.bat
```

Po uvedení síťových programů do provozu (dávka `initnet.bat`) naskočí program MNU na položku s číslem 1. Zde uživatel edituje své jméno. Po zmáčknutí `Enter` se objeví v místě pro napsání hesla černé okénko o výšce jednoho řádku, obsahující „neviditelnou“ editovatelnou položku (s číslem 2). Její neviditelnost je dána tím, že má barevné atributy rovny nule, tj. černý text na černém pozadí. Tato položka musí být popsána v samostatném okně, protože barevné atributy pro položky jsou společné pro celé okno; přitom my chceme text polořky `user` vidět, zatímco `password` nikoli.

Toto řešení sice nezobrazuje v místě hesla vůbec nic (tedy ani žádné hvězdičky či čtverečky), ale myslím si, že to je spíš k dobru věci – zvyšuje to bezpečnost uživatelských hesel.

Po zadání hesla se vyvolá *zevnitř* (z programu MNU) dávka `trylogin.bat`. Tato dávka obdrží dva parametry. Jedním je uživatelské jméno a druhým je heslo. Dávka zavolá přihlašovací program sítě s těmito dvěma parametry a vytvoří soubor `OK.bat` v případě, že bylo přihlášení úspěšné. Proč voláme dávku zevitř programu MNU? Je to proto, že informaci o obsahu hesla nejsme schopni předat řídicí dávce jinak, než v souborech typu `envir.bat`. Potulování hesel po souborech (třebaže smazaných) by ale mohlo oslabit jejich bezpečnost.

Zpětnou informaci o úspěšnosti přihlášení zase nelze přenést jinak, než existencí souboru. Navíc nyní musíme program MNU opustit, protože v programu samotném nemůžeme zařídit větvení výpočtu závislé na existenci souboru.[4] V hlavní dávce se v případě, že neexistuje soubor OK.bat přejde znova na volání programu MNU, nyní ale má výchozí položka číslo 10. Na této položce nám program MNU vynadá a umožní po klávese `Esc` přihlašovací proceduru opakovat.

Pokud je přihlášení úspěšné, řídicí proces se ukončí vyvoláním dávky `OK.bat`, která obsahuje jediný řádek ve tvaru např. `set USER=vopicka`. Po ukončení přihlašovací procedury máme tedy v proměnné `USER` podchycené uživatelské jméno. To je užitečné pro další dávky, které mohou s touto hodnotou pracovat. Samotný PC NFS software nám takovou věc neumožní.

Pro zajímavost ještě uveďme, jak vypadá dávka `trylogin.bat`, která realizuje vlastní přihlášení. Zde je její obsah:

```
@echo off
if exist OK.bat del OK.bat
if '%2==' exit
if '%1==' exit
net init %1 %2 < letter.a
if errorlevel 1 exit
echo set USER=%1 > OK.bat
```

Zde je použit ještě jeden nehezký trik. Program `net init` při neúspěšném přihlášení vydá zprávu typu `Abort, Retry, Ignore?`, přičemž my nechceme, aby uživatel měl možnost na tuto zprávu reagovat. Volbou `Ignore` by totiž mohl z přihlašování vyklouznout. Soubor `letter.a`, na který je přesměrován vstup, obsahuje písmeno `a`. Program `net init` tedy při neúspěšném přihlášení reaguje, jakoby bylo řečeno `Abort`, a v takovém případě vrátí nenulový error kód, což my v zápětí testujeme.

[4] Není to tak docela pravda; lze využít konstrukce typu `[%...]` například v definičním řádku okna a tím ovlivnit chování programu v závislosti na obsahu souboru. Není to ale příliš elegantní

Celý systém dávek je navržen tak, aby uživatel neměl šanci přihlášení nijak obejít. Při té příležitosti podotkněme, že program MNU je odolný proti kombinacím typu `Ctrl-Break` apod.

V konfiguračním souboru jsou dvě zajímavé globální definice. První z nich (`~timeout`) říká, že po dvaceti vteřinách klidu klávesnice se vyvolá program `scrblank`, který zhasíná obrazovku. Takový program udělá šikovný programátor za pár minut. Nemáte-li program tohoto typu po ruce, může posloužit DOSovské `cls`, ale toto řešení neskrývá kurzor. Šikovný programátor se přitom může vyřádit a udělat z obrazovky třeba „toulky vesmírem“ nebo „akvárium“ apod. Velice jednoduchý program `scrblank` jsem zařadil do $\mathcal{CS}$TEXu 94 (jedná se o pár řádků v jazyce C).

Druhou definicí, na niž bych chtěl upozornit, je `~mouse`. Parametry kurzoru myši jsou nastaveny tak, aby tento kurzor nebyl vidět. Přesněji, maska AND propustí všechno (hodnota FFFFh) a maska XOR nezmění nic (hodnota 0000h). To je důležité, protože například při implicitních hodnotách kurzoru myši by šlo pohybem kurzoru v poli pro heslo demaskovat postupně jednotlivá písmena hesla.

Nakonec ještě poznamenejme, že lokalizace některých souborů (například `OK.bat`) je ve skutečných dávkách provedena samozřejmě jinde, než v aktuálním adresáři. Pro stručnost jsem tyto věci nevypisoval. Pro vyšší bezpečnost proti virům a vlezlým uživatelům je navíc vše, co se týká konfigurace tohoto přihlášení, umístěno na takové části lokálního disku, která je zabezpečena proti zápisu.

Závěrem

Tímto článkem jsem chtěl naznačit, že lze program MNU použít při rozličných příležitostech. Zdaleka jsem nevyčerpal všechny možnosti programu. Pokud vás článek zaujal, mužete si přímo jednotlivé ukázky vyzkoušet. Všechny byly testovány.

Připravuji se provést nějaké úpravy v $\mathcal{CS}$TEXu, týkající se specifik naší lokální sítě. Právě proto, že vše stojí na řídicích dávkách DOSu, budou se takové změny dělat celkem snadno. Do všeho je totiž vidět; nic není skryto před všetečnými zásahy tzv. „MNU-inženýrů“, kterými se můžete stát i vy. Jediné, co je skryto, je zdrojový text programu MNU (asi 1600 řádků v jazyce C). Tento text nedám z ruky. Tím zabráním možnému šíření různých mutací se „zaručeně nejlepšími“ vylepšeními.

Ještě poslední příklad použití programu MNU. Asi před rokem jsem zařadil volání tohoto programu do `autoexec`u počítače naší paní sekretářky. Musel jsem samozřejmě pamatovat na všechny operace, které jsou na tomto počítači prováděny – od volání jednotlivých programových celků až po archivaci souborů na diskety. Tím se mi podařilo zcela izolovat uživatelku tohoto počítače od DOSovského promptu. Systém už nějakou dobu funguje a paní uživatelka si nestěžuje.

Na závěr bych chtěl popřát mnoho zdaru a trpělivosti všem odvážlivcům, kteří chtějí můj program potrápit nějakým netriviálním způsobem. Bohužel, jazyk konfiguračních souborů je dosti kryptický a možnosti DOSu a programu MNU jsou dost omezené. I za těchto omezení lze ale vytvořit neuvěřitelné věci.

Pokud někdo z vás přijde při konfiguraci TEXu na zajímavé použití programu MNU, velice mě potěší, když si o tom přečtu na stránkách tohoto časopisu.

Literatura

Kernighan, B. W. – Ritchie, D. M.: *The C Programming Language*, Englewood Cliffs, Prentice-Hall 1978.

Knuth D. E.: *The Art of Computer Programming*, vol. I–∞, Addison Wesley, 1972–...

Knuth D. E.: *The TEXbook*, vol. A of *Computers & typesetting*, Addison Wesley, 1986

Olšák P.: *Program MNU, Konfigurovatelné menu pro spouštění aplikací pod DOSem* (`mnu.tex` – dokumentace zahrnutá do CSTEXu), 1993

`cfg.doc`, `install.doc` – dokumentační soubory k příkladům použití programu MNU v CSTEXu.

MS DOS: Dokumentace k systému.