Zpravodaj Československého sdružení uživatelů TeXu

Jan Šustek

Zašifrování zdrojového textu při zachování jeho funkčnosti

Zpravodaj Československého sdružení uživatelů TeXu, Vol. 19 (2009), No. 4, 201–211

Persistent URL: http://dml.cz/dmlcz/150095

Zašifrování zdrojového textu při zachování jeho funkčnosti

Jan Šustek

Abstrakt

Článek ukazuje možnost, jak zašifrovat TEXový soubor tak, aby se navenek choval stejně jako původní soubor. Toho je možné využít například, pokud chceme druhému uživateli poslat balík maker, ale nechceme mu z nějakého důvodu prozradit zdrojový kód těchto maker.

Klíčová slova: Zašifrování souboru, `\uppercase`, Caesarova šifra.

1. Úvod

Při práci v TEXu můžeme stejného výsledku dosáhnout mnoha různými způsoby. Můžeme napsat text bez použití jediného makra. Nebo si naopak práci můžeme zautomatizovat definováním a použitím maker. Tato makra se mohou jmenovat různě a také míra jejich použití může být různá. Makra mohou být napsána čitelně i nečitelně. Zdrojový kód [3] je ukázkou extrémně nečitelně definovaných maker. Po zhlédnutí výstupu čtenář určitě najde mnoho způsobů, jak čitelněji napsat zdrojový text, aby výsledek zůstal stejný.

Tento článek ukazuje možnost, jak jednoduše zašifrovat zdrojový text do nečitelné podoby tak, abychom po přeložení TEXem dostali stejný výstup. Původní (čitelný) zdrojový text budu označovat jako *původní soubor* a výsledný (zašifrovaný) text jako *nový soubor*. Struktura *nového souboru* bude taková, že na prvním řádku budou definována makra, jejichž aplikací na zbývající (zašifrované) řádky se soubor rozšifruje.

Na několika místech budu makra popisovat hlouběji. Čtenář toto může brát jako cvičení z fungování token procesoru a expand procesoru. Proto tato místa budu označovat jako cvičení. Konec cvičení budu označovat symbolem //.

2. Použitá šifra

K zašifrování *původního souboru* je použita Caesarova šifra [4]. Na každém řádku je posun šifry jiný. Pro přesný popis je nutné zavést několik parametrů a označení. V textu budu pracovat pouze s celými čísly a tuto skutečnost nebudu dále zdůrazňovat.

Předpokládejme, že budeme šifrovat pouze znaky s ASCII kódem z intervalu $[K, L]$. Označme symbolem $\diamond$ operaci

$$x \diamond [K, L] := x - \lambda_x (L - K + 1)\,,$$

kde λ_x je celé číslo takové, že výsledek patří do intervalu $[K, L]$. V tomto článku bude vždy $\lambda_x \in \{-1, 0, 1\}$, což zjednoduší použitá makra. Šifra použitá na n-tém řádku bude

$$\varphi_n(c) = \begin{cases} (c + \beta_n) \diamond [K, L] & \text{pro } c \in [K, L]\,, \\ c & \text{jinak.} \end{cases} \tag{1}$$

Je zřejmé, že φ_n je bijekce na množině znaků. Její inverze je

$$\varphi_n^{-1}(c) = \begin{cases} (c - \beta_n) \diamond [K, L] & \text{pro } c \in [K, L]\,, \\ c & \text{jinak.} \end{cases} \tag{2}$$

Posun β_n určíme následovně. Označme $[\alpha, \gamma]$ interval, v němž se může posun nacházet. Dále vezměme parametry $\beta_0, \delta \in [\alpha, \gamma]$. Posun β_n definujme vztahem

$$\beta_{n+1} = (\beta_n + \delta) \diamond [\alpha, \gamma]\,. \tag{3}$$

Použitá šifra tedy závisí na šesti parametrech $K, L, \alpha, \beta_0, \gamma, \delta$. Aby vše fungovalo správně, musí být splněny podmínky[1] $K < L$, $0 < \alpha \leq \gamma \leq L - K$, $\alpha - \delta \leq \beta_0 \leq \gamma$ a $\alpha + \delta \leq \gamma$. Je vhodné, aby čísla δ a $L - K + 1$ byla nesoudělná. Dále je nutné, aby pro všechna $c \in [K, L]$ a pro všechna $\beta_n \in [\alpha, \gamma]$ platilo `\catcode` $\varphi_n(c) \notin \{9, 15\}$, přičemž se jedná o hodnotu `\catcode` platnou v době dešifrování. Také je nutné, aby TEX zapsal znaky $\varphi_n(c)$ do *nového souboru* přímo a ne pomocí dvojité stříšky. S ohledem na [1], sekci 49, je třeba položit $32 \leq K < L \leq 126$.

Hlavní mechanismus pro zašifrování tvoří primitiv `\uppercase`. Hodnota $\varphi_n(c)$ je uložena v registru `\uccode` c. Při práci jsou použity tři čítače:

- `\^^E` je aktuální hodnota β_n;
- `\^^F` je hodnota `\uccode\^^G`;
- `\^^G` je řídicí proměnná cyklu při generování `\uccode`.

V sekci 6 je popsáno, proč jsou tyto čítače pojmenovány tímto způsobem.

Samozřejmě by bylo možné vhodnou úpravou maker typ šifry změnit.

3. Zašifrování souboru

Zašifrování souboru se provede makrem `\zasifruj`, majícím deset parametrů,

`\zasifruj{`*původní soubor*`}{`*nový soubor*`}{`*pomocný soubor*`}`
`{`K`,`L`}{`α`,`β_0`,`γ`,`δ`}{`*poslední příkaz*`}`

[1] Vysvětlení těchto podmínek přenechávám na čtenáři.

Argumenty *pomocný soubor* a *poslední příkaz* budou popsány v sekci 4, o ostatních parametrech už byla řeč.

Makro `\zasifruj` nejdříve uloží hodnoty svých parametrů a nastaví pomocné čítače. Poté otevře používané soubory a makrem `\zapismakra` do *nového souboru* zapíše dešifrovací makra. Nastaví se hodnoty `\uccode` mimo interval $[K, L]$[2] a také počáteční hodnota β_n. Dále je nastaven `\endlinechar` na hodnotu -1, aby se při načítání řádků na konec nepřipojovaly nežádoucí mezery. Makrem `\dospecials` se nastaví kategorie speciálních znaků na hodnotu 12, aby při šifrování nedocházelo k nežádoucím efektům. Makrem `\zpracuj` se spustí šifrování jednotlivých řádků *původního souboru*. V závěru se použité soubory uzavřou.

```
\def\zasifruj#1#2#3#4#5#6{\def\puvodni{#1 }
  \def\novy{#2 }
  \strisky\novyS#2\^^H
  \def\pomocny{#3 }
  \strisky\pomocnyS#3\^^H
  \separujascii#4,
  \separujposun#5,
  \def\posledni{#6}
  \posunsirka\posunmax \advance\posunsirka-\posunmin
    \advance\posunsirka1
  \asciiminI\asciimin \advance\asciiminI-1
  \asciisirka\asciimax \advance\asciisirka-\asciiminI
  \openin\patnact\puvodni
  \immediate\openout\patnact\zasifrovany
  \zapismakra
  {\^^G0
  \loop \ifnum\^^G<255
    \advance\^^G1 \uccode\^^G\^^G
  \repeat
  \^^E\posunzacatek
  \endlinechar-1
  \def\do##1{\catcode`##112}
  \dospecials\zpracuj}
  \closein\patnact \immediate\closeout\patnact}
\def\separujascii#1,#2,{\def\asciimin{#1 }\def\asciimax{#2 }}
\def\separujposun#1,#2,#3,#4,{\def\posunmin{#1}
  \def\posunzacatek{#2 }\def\posunmax{#3 }\def\posunzmena{#4 }}
```

Makro `\zpracuj` načte jeden řádek *původního souboru*. Podle vzorce (3) nastaví β_n a poté podle vzorce (1) změní `\catcode` všech znaků z intervalu $[K, L]$.

[2]Kvůli zjednodušení maker jsou nastaveny hodnoty `\uccode` všech znaků.

Použitím primitivu `\uppercase` se řádek zašifruje a zapíše se do *nového souboru*.[3] Nakonec se makro `\zpracuj` zavolá na následující řádek.

```
\def\zpracuj{\read\patnact to\radek
  \ifeof\patnact\else
    \advance\^^E\posunzmena
    \ifnum\^^E>\posunmax \advance\^^E-\posunsirka\fi
    \^^G\asciiminI
    \loop
      \ifnum\^^G<\asciimax \advance\^^G1
      \^^F\^^G \advance\^^F\^^E
      \ifnum\^^F>\asciimax
        \advance\^^F-\asciisirka
      \fi
      \uccode\^^G\^^F
    \repeat
    \ifx\radek\byestring\else
    \ifx\radek\enddocstring\else
      \expandafter\uppercase\expandafter{\expandafter\immediate
        \expandafter\write\expandafter\patnact\expandafter
        {\radek}}%
    \fi\fi
    \expandafter\zpracuj
  \fi}
```

Cvičení Kdybychom nedefinovali řídicí sekvenci `\patnact` a uvnitř `\uppercase` na řádku 43 psali číslo `15`, nastal by problém s primitivem `\write`. Tokeny $\boxed{1}_{12}\boxed{5}_{12}$ by se změnily na $\boxed{\varphi_n(1)}_{12}\boxed{\varphi_n(5)}_{12}$ a výsledek `\uppercase` by byl

$$\boxed{\texttt{immediate}}\,\boxed{\texttt{write}}\,\boxed{\varphi_n(1)}_{12}\boxed{\varphi_n(5)}_{12}\boxed{\varphi_n(\{)}_1\,\varphi_n^*(\text{expanze }\texttt{\textbackslash radek})\,\boxed{\varphi_n(\})}_2$$

což by způsobilo chybu, protože primitiv `\write` očekává číslo souboru. //

Názvy použitých souborů je nutné do *nového souboru* zapsat. Není však vhodné zapsat tyto názvy přímo. Názvy se zašifrují tak, že se jednotlivé znaky zapíší pomocí dvojité stříšky. Rozšifrování poté provede token procesor. Viz například [2], strana 22. K tomuto účelu slouží makro `\strisky`, jehož použití je

`\strisky\cs` *text*`\^^H`

[3] Pokud je řádek složen z tokenů $\boxed{\backslash}_{12}\boxed{\texttt{b}}_{11}\boxed{\texttt{y}}_{11}\boxed{\texttt{e}}_{11}$ nebo $\boxed{\backslash}_{12}\boxed{\texttt{e}}_{11}\boxed{\texttt{n}}_{11}\boxed{\texttt{d}}_{11}\boxed{\{}_{12}\boxed{\texttt{d}}_{11}\boxed{\texttt{o}}_{11}\boxed{\texttt{c}}_{11}\boxed{\texttt{u}}_{11}\boxed{\texttt{m}}_{11}\boxed{\texttt{e}}_{11}\boxed{\texttt{n}}_{11}\boxed{\texttt{t}}_{11}\boxed{\}}_{12}$, nebude se do *nového souboru* zapisovat. Důvod bude vysvětlen později.

Zašifrovaný *text* se uloží do makra `\cs`. Například po použití

```
\strisky\sifra Ahoj\^^H
```

budou v makru `\sifra` uloženy tokeny

$$\boxed{\hat{}}_7\boxed{\hat{}}_7\boxed{4}_{12}\boxed{1}_{12}\boxed{\hat{}}_7\boxed{\hat{}}_7\boxed{6}_{12}\boxed{8}_{12}\boxed{\hat{}}_7\boxed{\hat{}}_7\boxed{6}_{12}\boxed{f}_{12}\boxed{\hat{}}_7\boxed{\hat{}}_7\boxed{6}_{12}\boxed{a}_{12}$$

Spoluautorem makra `\strisky` je Petr Březina.

```
\def\strisky#1{\def#1{}\striskyA#1}
\def\striskyA#1#2{\ifx#2\^^H\else
    \chardef\^^N=‘#2
    \expandafter\striskyB\meaning\^^N
    \edef#1{#1^\empty^\^^O}
    \expandafter\striskyA\expandafter#1\fi}
\def\striskyB#1"#2#3{\lowercase{\def\^^O{#2#3}}}
```

Cvičení Pro získání čísla znaku v šestnáctkové soustavě je použit primitiv `\meaning`. Po expanzi primitivu `\meaning` použitého na token, který je definován primitivem `\chardef` nebo `\mathchardef`, dostaneme tokeny

$$\boxed{\backslash}_{12}\Big(\boxed{m}_{12}\boxed{a}_{12}\boxed{t}_{12}\boxed{h}_{12}\Big)\boxed{c}_{12}\boxed{h}_{12}\boxed{a}_{12}\boxed{r}_{12}\boxed{"}_{12}$$

za nimiž následuje dané číslo v šestnáctkové soustavě reprezentované jako tokeny kategorie 12. Toto je jediná přímá cesta, jak lze v TEXu převést číslo do šestnáctkové soustavy. //

Cvičení Pokud bude text poslaný makru `\strisky` obsahovat znak s kódem menším než 16, bude mít jeho zápis v šestnáctkové soustavě pouze jeden znak. V takovém případě nebude makro `\striskyB` fungovat správně. Řešení tohoto problému přenechávám na čtenáři. //

Cvičení Konverze makrem `\strisky` nebude fungovat také v případě, že bude parametr obsahovat mezeru (kategorie 10).[4] Předpokládejme, že zavoláme

```
\strisky\sifra Z y\^^H
```

Po prvním provedení makra `\striskyA` budou ve čtecí frontě tokeny

$$\boxed{\texttt{striskyA}}\boxed{\texttt{sifra}}\boxed{\sqcup}_{10}\boxed{y}_{11}\boxed{\hat{}\hat{}H}$$

[4] Tento případ nemůže nastat, pokud budeme konvertovat názvy vkládaných souborů. Podle [1], sekce 516, nemůže totiž název vkládaného souboru obsahovat mezeru.

Expand procesor při načítání neseparovaných parametrů přeskakuje tokeny kategorie 10. Proto se makro `\striskyA` zavolá s parametry `#1` = $\boxed{\texttt{sifra}}$ a `#2` = $\boxed{\texttt{y}}_{11}$. V důsledku toho nakonec budou v makru `\sifra` uloženy tokeny

$$\boxed{\hat{}}_7\boxed{\hat{}}_7\boxed{5}_{12}\boxed{a}_{12}\boxed{\hat{}}_7\boxed{\hat{}}_7\boxed{7}_{12}\boxed{9}_{12}$$

namísto správných tokenů

$$\boxed{\hat{}}_7\boxed{\hat{}}_7\boxed{5}_{12}\boxed{a}_{12}\boxed{\hat{}}_7\boxed{\hat{}}_7\boxed{2}_{12}\boxed{0}_{12}\boxed{\hat{}}_7\boxed{\hat{}}_7\boxed{7}_{12}\boxed{9}_{12}$$ //

Cvičení TEX interpretuje dvojitou stříšku pouze na úrovni token procesoru. Proto se na řádku 53 znaky

$$\hat{}_7\backslash_0 e_{11} m_{11} p_{11} t_{11} y_{11} \hat{}_7\backslash_0\hat{}_7\hat{}_7 0_{11}$$

interpretují jako

$$\boxed{\hat{}}_7\boxed{\texttt{empty}}\ \boxed{\hat{}}_7\boxed{\hat{}\hat{}0}$$

což po expanzi dává

$$\boxed{\hat{}}_7\boxed{\hat{}}_7(\text{expanze } \texttt{\^\^0})$$

V dalším zpracování už se tokeny $\boxed{\hat{}}_7\boxed{\hat{}}_7$ jako dvojitá stříška neinterpretují. //

Nesmíme zapomenout na deklaraci použitých registrů.

```
\newcount\asciiminI \newcount\posunsirka \newcount\asciisirka
\newcount\^^E \newcount\^^F \newcount\^^G \chardef\patnact15
```

4. Rozšifrování souboru

Makra podobná výše uvedeným rozšifrují všechny řádky *nového souboru* s výjimkou prvního řádku do *pomocného souboru*. Rozšifrování se provede podle vzorce (2). Poté se *pomocný soubor* zavře a načte. Protože je v *pomocném souboru* obsažen nezašifrovaný text *původního souboru*, bylo by vhodné *pomocný soubor* smazat. Toto však lze udělat pouze při zapnutém `\write18` s použitím prostředků operačního systému. Protože nemáme zaručeno, že uživatel bude mít zapnuto `\write18`, je třeba si pomoci jinak. *Pomocný soubor* otevřeme pro zápis (tím se jeho obsah vymaže) a ihned zavřeme.

V závěru je třeba zajistit, aby se *nový soubor* nedočetl do konce. K tomu slouží *poslední příkaz*. Pokud šifrujeme soubor, který se vkládá do jiného souboru (například balíček LATEXu), použijeme `endinput`, pokud se jedná o hlavní soubor v plainu, použijeme `bye`.

Je důležité, aby všechna makra z prvního odstavce této sekce byla v *novém souboru* na prvním řádku. O to se postará makro `\zapismakra`. Aby byl *nový soubor* hůře čitelný, jsou použitá makra nazvána `\^^A` až `\^^D` a použité čítače `\^^E` až `\^^G`.

```
\begingroup
  \catcode`\|0 \catcode`(1 \catcode`)2 \catcode`\^12
  \catcode`\\12 |catcode`|{12 |catcode`|}12 |catcode`|#12
  |gdef|byestring(\bye)
  |gdef|enddocstring(\end{document})
  |gdef|zapismakra(|immediate|write|patnact(%
    {\chardef\^^A15%
    \countdef\^^E221\countdef\^^F222\countdef\^^G223%
    \def\^^B{\read\^^Ato\^^C%
      \ifeof\^^A\else%
        \ifnum\^^E=0%
          \^^E|posunzacatek
        \else%
          \advance\^^E|posunzmena
          \ifnum\^^E>|posunmax
            \advance\^^E-|the|posunsirka
          \fi%
          \^^G|the|asciiminI
          \loop%
            \ifnum\^^G<|asciimax \advance\^^G1%
            \^^F\^^G\advance\^^F-\^^E%
            \ifnum\^^F<|asciimin
              \advance\^^F|the|asciisirka
            \fi%
            \uccode\^^G\^^F%
          \repeat%
          \expandafter\uppercase\expandafter%
            {\expandafter\immediate\expandafter%
            \write\expandafter\^^A\expandafter{\^^C}}%
        \fi%
        \expandafter\^^B%
      \fi}%
    \def\^^D{{\def\do##1{\catcode`##112}\endlinechar-1%
      \^^G0%
      \loop \ifnum\^^G<255%
        \advance\^^G1\uccode\^^G\^^G%
      \repeat%
```

```
95        \^^E0\dospecials\^^B}}%
96      \openin\^^A|novyS \immediate\openout\^^A|pomocnyS
97      |space\^^D%
98      \closein\^^A\immediate\closeout\^^A}%
99      \input |pomocnyS
100     \immediate\openout15|pomocnyS \immediate\closeout15%
101     \|posledni))
102 |endgroup
```

Cvičení Při načítání názvu souboru TEX provádí expanzi, dokud nenarazí na mezeru nebo neexpandovatelnou řídicí sekvenci. Kdyby na řádku 97 nebyla mezera vložená makrem `\space`, TEX by při načítání názvu souboru (vloženého makrem `\pomocnyS`) pokračoval expandováním makra `\^^D`. Tam by narazil na token $\boxed{\texttt{\{}}_1$. Protože se tokeny kategorie 1 při načítání názvu souboru interpretují jako běžné znaky, chápal by TEX závorku `{` jako součást názvu souboru. Teprve token $\boxed{\texttt{def}}$ by sloužil jako oddělovač. Proto je nutné na řádku 97 mezeru explicitně vložit. //

Cvičení Kdybychom při šifrování *původního souboru* zašifrovali i tokeny $\boxed{\texttt{\textbackslash}}_{12}\boxed{\texttt{b}}_{11}\boxed{\texttt{y}}_{11}\boxed{\texttt{e}}_{11}$, pak by se TEX ukončil při vložení *pomocného souboru* na řádku 99 a k důležitému řádku 100 už by se nedostal. //

Cvičení Pokud v době dešifrování nebude mít znak `^` kategorii 7, token procesor název souboru nerozšifruje. Například soubor SLOVAK.LDF nastavuje v místě `\begin{document}` znak `^` aktivní. To znamená, že pokud používáme `\usepackage[slovak]{babel}` a *původní soubor* obsahuje `\begin{document}`, pak se názvy souborů vložené na řádku 96 rozšifrují správně, ale název vložený na řádku 100 se nerozšifruje. V důsledku toho se *pomocný soubor* nevymaže. Čtenář určitě najde jednoduché řešení tohoto problému. //

5. Ukázka

Zkopírujme řádky 1–102 do souboru s názvem A.TEX.[5] Poté můžeme libovolný soubor zašifrovat, pokud TEXem přeložíme soubor obsahující jediný řádek

```
\input a \zasifruj... \bye
```

Soubor A.TEX je relativně čitelný. Nyní vytvoříme soubor B.TEX, který bude dělat totéž, ale bude nečitelný. Tento soubor vytvoříme zašifrováním souboru A.TEX, například s nastavením konstant $K = 33$, $L = 126$, $\alpha = 37$, $\beta_0 = 41$,

[5] Soubor je umístěn na adrese `http://www1.osu.cz/~sustek/TeX/zasifruj.tex`.

$\gamma = 47$ a $\delta = 3$. Protože soubor B.TEX budeme primitivem `\input` vkládat do jiného souboru, ukončíme jeho čtení sekvencí `endinput`. Jako *pomocný soubor* vytvoříme soubor B.TXT.

Vytvořme nový soubor, zapišme do něj

```
\input a \zasifruj{a.tex}{b.tex}{b.txt}{33,126}{37,41,47,3}{endinput} \bye
```

a přeložme tento soubor TEXem. Dostaneme soubor B.TEX, jehož první řádky jsou[6]

```
{\chardef\^^A15\countdef\^^E221\countdef\^^F222\countdef\^^G223\def\^^B{\read\^^Ato\^^C
\ifeof\^^A\else\ifnum\^^E=0\^^E41 \else\advance\^^E3 \ifnum\^^E>47 \advance\^^E-11\fi\^
^G32\loop\ifnum\^^G<126 \advance\^^G1\^^F\^^G\advance\^^F-\^^E\ifnum\^^F<33 \advance\^^
F94\fi\uccode\^^G\^^F\repeat\expandafter\uppercase\expandafter{\expandafter\immediate\e
xpandafter\write\expandafter\^^A\expandafter{\^^C}}\fi\expandafter\^^B\fi}\def\^^D{{\de
f\do##1{\catcode`##112}\endlinechar-1\^^G0\loop \ifnum\^^G<255\advance\^^G1\uccode\^^G\
^^G\repeat\^^E0\dospecials\^^B}}\openin\^^A^^62^^2e^^74^^65^^78\immediate\openout\^^A^^
62^^2e^^74^^78^^74 \^^D\closein\^^A\immediate\closeout\^^A}\input ^^62^^2e^^74^^78^^74\
immediate\openout15^^62^^2e^^74^^78^^74\immediate\closeout15\endinput
*234*H/9=2C8O]O^O_O‘OaObI*234*>CD=2<7IO] K
  -567-K2<@5@G2?JLRa N
  %<=;2<4B%C*48-8?*7BzJY%’’o
  (O12(<;9;/:EGM] I
  +BCA8B:H+?><>2=H"P‘+--u
  #:,7(9<1(:*OOHYQ
  &=/:+<?4:9=?8K]T
  )123)=<@921;6HNaJ
  ,@?CE>C9B;1,@?CE>=1H ,14F1>35,@?CE>C9B;1[,@?CE>=9>
    $),>)6+-$87;=6;1:3)W
  ’,>.44849r’,>.44849 ’,/A,9.O’,>.44849rVZ
  */A177A7@9/*/A177;/F */2D/<13*/A177A7@9/Y*/A177;7<u
  -@A6?:?-A2E?24E-AFG@5?: -:>>65:2E6-@A6?@FE-A2E?24E-K2<@5@G2?J
  %C*92<6*4;*
  G(**qZ
  +;>>? +85=D<+--ti_bb
    #(+=(5*,#%%lV #<**6+,#%%l#%%l
  &</:/+>
  )++p)=<@B;G.O.A28
  ,5>4<9>5381B[_
  $,-.$,7IIWC$+)<+7,-(IIWWXE
  ’/:>;O.4,7>’E;=,.@5H
  *1:=A37<*>/B</1B *7;;327/B3*1:=A3=CB*>/B</1BK
-567-D6A2CF;2D4::R‘[Ra[L-567-2D4::>:?LR‘ N-567-2D4::>2ILRa NN
%-./%<.9*;>398<>7JXSJYSJZSJ[SD%-./%98<>7627DJXF
  (O12(<;?A:F-/-@17GM\ I(O12(<;?A:9-DGM] I(O12(<;?A:F91:-GM^ II
+345+I?A02D9J+A403+?OC=O2C C>+A034:
...
```

Soubor B.TEX můžeme používat stejným způsobem, jako se používá soubor A.TEX. Libovolný soubor můžeme zašifrovat, pokud TEXem přeložíme soubor obsahující jediný řádek

```
\input b \zasifruj... \bye
```

[6] Ve skutečnosti prvních devět řádků výpisu tvoří jediný řádek souboru B.TEX.

6. Možné problémy

Pokud bude *nový soubor* načítán uvnitř jiného souboru, může dojít ke kolizím s makry definovanými před načtením *nového souboru*. Aby se pravděpodobnost těchto kolizí minimalizovala, jsou provedena následující opatření.

- Všechna nová makra jsou definována uvnitř skupiny.[7]
- Nové řídicí sekvence mají „nepravděpodobné“ (a zároveň nečitelné) názvy `\^^A` až `\^^G`.
- Jsou použity registry `\count221` až `\count223`, jejichž pravděpodobnost použití je malá. Tyto registry nejsou deklarovány makrem `\newcount`, ale je přímo použit primitiv `\countdef`.
- Je použito nejvyšší možné číslo souboru 15, které není deklarováno makry `\newread` a `\newwrite`, ale je přímo použit primitiv `\chardef`.

Pro správné rozšifrování je třeba, aby všechny použité primitivy a makra měly svůj původní význam. Pokud je některý z nich před načtením *nového souboru* předefinován, nelze zaručit správné fungování.

Slabým místem použitého postupu je použití *pomocného souboru*. Kdyby nějaký hacker na vhodném místě změnil *nový soubor* tak, aby se *pomocný soubor* nevymazal, mohl by z *pomocného souboru* jednoduše získat *původní soubor*. Otázkou je, zda je možné vše udělat bez použití *pomocného souboru*.

Závěr

V TEXu je možné napsat dokument mnoha různými způsoby. Stejně tak je možné různými způsoby napsat balík maker. Existují způsoby čitelné a způsoby nečitelné. Pokud chceme, aby druhý uživatel používal naše makra, musíme mu poslat zdrojový kód těchto maker. Pokud však nechceme, aby se uživatel k tomuto zdrojovému kódu dostal, je třeba jej zašifrovat a tento zašifrovaný soubor uživateli poslat.

Článek ukazuje jednu z možností, jak toho dosáhnout. Soubor je zašifrován použitím primitivu `\uppercase`. Na prvním řádku zašifrovaného souboru jsou definována makra, která zbytek souboru rozšifrují. Tím je zajištěna původní funkčnost souboru.

Poděkování

Rád bych poděkoval Petru Březinovi, Petře Konečné, Pavlu Střížovi a Janu Štěpničkovi za pročtení článku a cenné připomínky a nápady vedoucí ke zkvalitnění článku.

[7] Pokud je *nový soubor* načítán při nastaveném `\globaldefs=1`, zavedení skupiny nepomůže. Další opatření sníží pravděpodobnost kolize. Největší problém pak mohou způsobit jinak nastavené hodnoty `\uccode`.

Seznam literatury

[1] Knuth, Donald. *TEX: The Program* (Computers and Typesetting, Volume B). Reading, Massachusetts: Addison-Wesley, 1986. ISBN 0-201-13437-3. Dostupné též online ze serveru: `ftp://tug.ctan.org/pub/tex-archive/systems/knuth/dist/tex/tex.web`

[2] Olšák, Petr. *TEXbook naruby*. [TEXbook Inside Out.] 2. vyd. Brno, nakladatelství Konvoj, 2001. ISBN 80-7302-007-6. Dostupné na `ftp://math.feld.cvut.cz/pub/olsak/tbn/tbn.pdf`

[3] Carlisle, David. *obscure.tex*. Dostupné na `http://www.maths.abdn.ac.uk/tex/latex_course/obscure.tex`

[4] Wikipedie: Otevřená encyklopedie: *Caesar Cipher* [online]. [citováno 2. října 2009]. Dostupné na `http://en.wikipedia.org/wiki/Caesar_cipher`

Summary: On Enciphering a Source Code and Preserving Its Functionality

It is possible to write a document in many different ways using TEX. Similarly, one can write a macro package in many different ways. Some ways are readable and some are not. If we want another user to use our macros, we have to send him the source code of those macros. But if we don't want him to see the source code, we have to encipher it and to send the ciphertext only to the user.

This paper shows one possibility of how to achieve this goal. The original file is enciphered using the `\uppercase` primitive. The first line of the ciphered file contains macros for deciphering the other lines. This ensures that the ciphered file has the same functionality as the original file.

Key words: File enciphering, `\uppercase`, Caesar cipher.

Jan Šustek, jan.sustek@osu.cz
Ostravská univerzita, Přírodovědecká fakulta, Katedra matematiky
30. dubna 22, CZ-701 03 Ostrava, Czech Republic