Zpravodaj Československého sdružení uživatelů TeXu

Petr Olšák

Nový encTeX - kódování UTF-8 v TeXu

Zpravodaj Československého sdružení uživatelů TeXu, Vol. 13 (2003), No. 2, 98–106

Persistent URL: http://dml.cz/dmlcz/149928

přepínat mezi různě kódovanými vzory dělení stejného jazyka. Přitom členové LaTeX-teamu jsou toho názoru, že to možná bude zapracováno až do LaTeXu 3.

6. Reference

[1] Donald Knuth. *The TeXbook*. Addison Wesley Publishing Company. Eleventh printing, revised, May 1991, ISBN 0-201-13447-0

[2] Petr Olšák. *Typografický systém TeX*. Konvoj, Brno 2000, ISBN 80-85615-91-6

[3] Petr Olšák. *TeXbook naruby*. Konvoj, Brno 2001, ISBN 80-7302-007-6.

[4] Petr Olšák. *První setkání s TeXem*. Volně šířený dokument ve formátech `tex`, `ps`, `pdf` na `ftp://math.feld.cvut.cz/pub/cstex/doc/prvni.*`, 22 stran.

[5] Petr Olšák. *Putování písmene ř z klávesy na papír*. Zpravodaj Československého sdružení uživatelů TeXu, **7** (3), 109–118 (1997)

[6] Petr Olšák. *Rozšíření TeXu encTeX*. Rozšíření ve formě změnového souboru k `tex.web` je volně šířeno na `ftp://math.feld.cvut.cz/pub/olsak/enctex/`.

[7] Petr Olšák. *Program* `a2ac`. Program včetně zdrojových kódů v jazyce C je volně šířen na `ftp://math.feld.cvut.cz/pub/olsak/a2ac/`.

[8] Petr Olšák. *Test* `cstrip`. Test je nepovinnou součástí $\mathcal{CS}$TeXu. Je volně šířen na `ftp://math.feld.cvut.cz/pub/cstex/base/cstrip.tar.gz`.

[9] Petr Olšák. *Makro OFS*. TeXové makro OFS pro práci s rozsáhlými kolekcemi fontů je volně šířeno na `ftp://math.feld.cvut.cz/pub/olsak/ofs/`.

Nový encTeX – kódování UTF-8 v TeXu

Petr Olšák

EncTeX je rozšíření TeXu, které zavádí nové primitivy TeXu pro nastavování konverzního algoritmu na úrovni input procesoru nebo při výstupu na terminál, do logu a do `\write` souborů. První verzi encTeXu jsem napsal v roce 1997. Tato verze umožňovala pomocí primitivů `\xordcode`, `\xchrcode` a `\xprncode` číst a nastavovat konverzní vektory TeXu *xord* a *xchr* a nastavit „tisknutelnost“ každého znaku na terminál, do logu a do `\write` souborů. O této verzi jsem podrobně psal v [7].

V roce 2002 prosákla na diskusních skupinách o TeXu potřeba používat ve zdrojových textech pro TeX kódování Unicode, přesněji ve formě UTF-8. Toto

kódování bylo nastaveno jako výchozí pro vesměs všechny textové editory například v nových distribucích RedHat. Anglicky mluvící uživatelé nepociťují žádný problém, protože 96 tisknutelných znaků z ASCII je v UTF-8 kódováno naprosto shodně jako ve standardu ASCII samotném. Znaky z jiných abeced jsou ovšem kódovány ve dvou nebo více bytech. Tím je umožněno nekonfliktně mít v jednom dokumentu současně miliony znaků z abeced celého světa. O to hlavně v tomto kódování jde.

Programové aplikace, které interně rezervují pro každý znak 16 bitů, nemají se čtením UTF-8 skoro[1] žádné problémy, protože existuje jednoduchý algoritmus pro konverzi z 16bitového kódování podle Unicode do UTF-8 nebo obráceně. TEX problémy má, protože nepracuje interně v šestnácti bitech. Z této úvahy asi vyplynula na diskusní skupině mylná představa, že jediným možným řešením na přímé zpracování textu v UTF-8 je ΩTEX.

Už v této diskusi jsem naznačoval, že konverzní algoritmus umožňující čtení z UTF-8 je možné implementovat i do jednoduchého rozšíření TEXu na úrovni input procesoru. Nemusíme hned kvůli tomu použít ΩTEX.

Donald Knuth použil v input procesoru TEXu *xord* a *xchr* vektory hlavně proto, aby odstínil na systému a vstupním kódování nezávislé prostředí „uvnitř“ TEXu (na které navazuje na systému nezávislá implementace fontů) od vstupního kódování textů, které může být závislé na použitém operačním prostředí. Naše abeceda bývá v různých prostředích kódována různě (existuje aspoň 6 různých standardů nebo pseudostandardů). Díky *xord* a *xchr* vektorům jsme odfiltrovali tento kódovací chaos různých systémů od vnitřních záležitostí TEXu. Například formát `csplain` je postaven na vnitřním kódování ISO-8859-2 a chová se uvnitř zcela stejně ve všech dostupných operačních prostředích.

Z tohoto pohledu na věc je zřejmé, že UTF-8 je jen další z řady standardů, které kóduje (mimo jiné) naši abecedu. Pro bezproblémový provoz `csplainu` i na starších dokumentech, které jsou nově překódovány do UTF-8, se tedy hodí mít konverzní algoritmus v input procesoru TEXu. Bohužel, protože se jedná o vícebytové kódování, nebude ten algoritmus tak úplně jednoduchý, jako bylo použití stávajících *xord* a *xchr* vektorů. Přesto jsem se o implementaci takového algoritmu pokusil a výsledek nabídl TEXové veřejnosti ve formě nového encTEXu.

Chtěl bych ještě upozornit na balíček `ucs.sty`, který se také snaží řešit čtení dokumentů kódovaných v UTF-8. Jedná se ale jen o balíček maker, který spolupracuje s LATEXem, a konverzi provádí pomocí aktivních znaků na úrovni expand procesoru. To je věc pro `csplain` naprosto nepoužitelná, protože `cstrip` (test pro `csplain`, viz [5]) explicitně vyžaduje, aby všechny znaky české a slovenské abecedy měly kategorii 11 (písmeno). Také je v tomto testu ověřován zpětný výstup do logu a `\write` souborů, který musí kódovat českou a slovenskou abecedu

[1] Problémy mohou nastat, pokud sada znaků použitá v UTF-8 kódovaném dokumentu přesáhne možnosti 16bitového standardu, což kódování UTF-8 principiálně umožňuje.

stejně jako vstupní soubory (tj. závisle na použitém operačním prostředí). Důležité je, aby se v použitém operačním prostředí daly české a slovenské texty v logu a `\write` souborech přímo číst člověkem nebo znovu načíst TEXem.

Také je pro bezkonfliktní chod některých maker nutné, aby jeden znak naší abecedy byl skutečně jedním znakem (tokenem) a nebylo jej nutno na úrovni makrojazyka TEXu skládat ze dvou nebo více tokenů. Přesně v duchu zásady, že záležitosti vstupních kódování mají být odstíněny od záležitostí, které řešíme uvnitř TEXu. Například makro

```
\def\testujznak{\futurelet\znak \badejnadznakem}
```

musí testovat celistvé znaky naší abecedy, které si makro uloží do proměnné `\znak`. Není možné, aby tam byla uložena při kódování UTF-8 jen „půlka znaku“ zatímco při ISO-8859-2 kódování znak celý. Nebo makro

```
\everypar{\vetsiprvnipismeno}
\def\vetsiprvnipismeno#1{{\vetsifont#1}}
```

musí zvětšit první písmeno každého odstavce i za situace, že toto písmeno je třeba „Ř“. Snad se mi podařilo přesvědčit čtenáře, že překódování vstupních textů je kvůli odstínění od vnitřních záležitostí nutné řešit na úrovni input procesoru.

Nedávno byla ohlášena Frankem Mittelbachem příprava podpory UTF-8 přímo v balíčku `inputenc`. Je potřeba si uvědomit, že princip toho balíčku je stejný jako `ucs.sty`, tj. konverze probíhá až na úrovni expand procesoru. Jednoduchá makra `\testujznak` a `\vetsiprvnipismeno` samozřejmě nebudete moci použít ani tam. Podrobněji o problematice vztahu input procesoru a balíčku `inputenc` se zmiňuji v článku [8].

Základní vlastnosti

EncTEX v nové verzi umožňuje primitivem `\mubyte` nastavit konverzi více bytů na jeden byte nebo kontrolní sekvenci. Můžeme tedy zachovat interní 8bitové kódování v TEXu, které dosud používáme (například v csplainu je to ISO-8859-2), a mapovat pomocí encTEXu vícebytové kódy z UTF-8 do tohoto kódování. Mnoho dalších kódů z UTF-8 můžeme pak mapovat přímo na zvolenou kontrolní sekvenci, kterou definujeme například jako makro. Makro se postará o tisk požadovaného znaku (například sestavením kompozitu nebo přechodným přepnutím do jiného fontu). Počet kontrolních sekvencí není v TEXu významně omezen, takže tímto způsobem můžeme obsloužit vesměs všechny znaky, které kódování UTF-8 nabízí, pokud pro tisk těchto znaků máme nějaké řešení. Pokud ale řešení pro tisk některých znaků neexistuje, pak nám ani ΩTEX nepomůže.

Za zmínku stojí, že token procesor tokenizuje konvertované byty běžným způsobem (tj. podle kategorie znaku), zatímco kontrolní sekvence, které jsou

výsledkem multibytové konverze vstupního procesoru, nejsou token procesorem nijak měněny. Po přečtení takové kontrolní sekvence zůstává token procesor ve stavu neignorování mezer.

Příklady

```
\mubyte  ^^c1      ^^c3^^81\endmubyte % Á
\mubyte  ^^e1      ^^c3^^a1\endmubyte % á
% atd. -- implementace UTF8 do ISO8859-2

% příklady na kontrolní sekvenci:
\mubyte \endash    ^^c4^^f6\endmubyte %
\mubyte \alpha     ^^ce^^b1\endmubyte %
\mubyte \beta      ^^ce^^b2\endmubyte %

\mubyte \leftarrow ^^e2^^86^^90\endmubyte
\mubyte \uparrow   ^^e2^^86^^91\endmubyte
```

První token za `\mubyte` je byte nebo kontrolní sekvence, na kterou se konvertuje následující sekvence bytů (po přeskočení případné mezery). Deklarace vstupní sekvence bytů je ukončená primitivem `\endmubyte`. Tímto způsobem se postupně naplní konverzní tabulka. Poté je možno tabulku aktivovat nastavením registru `\mubytein` na kladnou hodnotu.

Považuji za zbytečné zde podrobně rozepisovat vlastnosti algoritmu a parametry nově zavedených primitivů, protože tyto věci jsou důkladně popsány v dokumentaci [2].

Výchozí hodnoty všech registrů encTEXu jsou nastaveny tak, že se encTEX chová naprosto stejně, jako standardní TEX.

EncTEX se stará i o zpětnou konverzi při zápisu do logu, `\write` souborů a na terminál. Tuto konverzi zapínáme nastavením `\mubyteout` a `\mubytelog` na kladnou hodnotu.

Pokud chceme konvertovat zpětně do kódu UTF-8 i takový znak, který jsme mapovali na kontrolní sekvenci, nesmí se nám tato sekvence při výstupu do souboru expandovat. EncTEX se sám dokáže postarat o potlačení takové expanze – stačí nastavit hodnotu `\mubyteout` na trojku.

EncTEX rovněž dokáže deklarovat nezávisle vstupní konverzi na výstupní konverzi (pokud by to bylo potřeba). Také může být vypnutá nebo zapnutá zpětná konverze nezávisle do jednotlivých `\write` souborů. To se hodí tehdy, když výstupní soubor budeme chtít předložit nějakému programu (třeba na seřazování rejstříků), který se zatím neumí vyrovnat s kódováním UTF-8.

Příklady nesouvisející s kódováním

Vedlejším efektem encTEXu (pro mnohé možná zajímavým) je možnost zpracovávat jednoduše jisté struktury ve vstupním souboru, které by šly ošetřit běžnými TEXovými makry jen velice obtížně. Uvedu jednoduchý příklad:

```
\bgroup \uccode‘X=\endlinechar
\uppercase{\gdef\echar{X}}\egroup
\def \setmubyte #1#2{\mubyte #1 #2\endmubyte}
\setmubyte \pomlka {\space-\space}
\setmubyte \pomlka {\echar-\space}
\setmubyte \pomlka {\space-\echar}
\def\pomlka {\leavevmode\unskip~-- \ignorespaces}
\mubytein=1
```

Od této chvíle se každý znak mínus ve vstupním souboru, který má z obou stran mezery, promění na pomlčku obklopenou zleva nezlomitelnou mezerou a zprava běžnou mezerou. Kromě toho se znak mínus může vyskytovat na začátku nebo na konci řádku. I v tomto případě se promění na pomlčku.

Je zřejmé, že tento trik bychom mohli udělat na úrovni klasických TEXových maker jen za cenu aktivní mezery, která by nám asi dělala paseku v jiných částech dokumentu. V našem příkladě ale mezera zůstává obyčejnou mezerou (kategorie 10) a starost o sledování výskytu „mezera, mínus, mezera“ přebírá input procesor encTEXu.

Jiný příklad (viz též [3]) ukazuje možnost přehledného psaní znaku ©:

```
\mubyte \copyright (C)\endmubyte  \mubytein=1
Za znakem (C) se mezera objeví, zatímco
za znakem \copyright se mezera ztratí.
```

Za zajímavou aplikaci encTEXu považuji implementaci programu vlna (automatické doplňování nezlomitelných mezer za neslabičné předložky) do input procesoru. Najdete ji v balíku encTEXu v souboru `vlna.tex`.

Dovedu si představit, že půjde daleko pohodlněji encTEXem nastavit čtení XML souborů, než když se o to pokoušíme jen pomocí TEXových maker.

Historie a vývoj encTEXu

EncTEX byl a je distribuován formou záplaty souboru `tex.ch`, který se používá při kompilaci TEXu. Záplata je připravena pro `tex.ch` z distribuce web2c. To je distribuce, ze které je odvozena jednak distribuce teTEXu (hojně používaná v UNIXech) a jednak v fpTEXu (pro systémy MS Windows). Kromě toho je k dispozici přesný seznam změn souboru `tex.ch`, které jsou nezávislé na použité TEXové distribuci, a je tedy aplikovatelný na jakýkoli TEX.

Aplikováním záplaty encTEXu a následným překladem binárních programů se encTEXem pozmění programy TEX, eTEX, pdfTEX a pdfeTEX. Ve všech těchto programech můžete pak využívat rozšiřující primitivy z encTEXu.

Jak již bylo řečeno, první verzi encTEXu jsem napsal už před mnoha lety. Následovala poněkud rozsáhlejší diskuse v časopise TUGboat, ve kterém jsem tuto věc publikoval. Diskuse se točila kolem toho, zda rozšíření TEXu směrem k encTEXu je ještě možné nazývat TEX. Samozřejmě, že ne, i když (bez použití nových primitivů) se ten program chová absolutně stejně jako TEX. Skutečnost, že to není TEX, mnohé odradila od jeho nasazení. Pak Poláci oprášili TCX tabulky[2] a použití encTEXu se stalo zbytečným. Nechtěl jsem jej vedle TCX tabulek dále propagovat, protože to tehdy dělalo víceméně totéž. Více nezávislých možností na překódování by jen pletlo uživatele.

Na základě požadavků na kódování UTF-8 pro TEX jsem na přelomu roku 2002 a 2003 naprogramoval novou verzi encTEXu. Kdo si přečetl pozorně úvodník Zpravodaje 1/2003, ten ví naprosto přesně, kdy ten program vznikl. V této verzi jsem se rozhodl encTEX znovu začít propagovat, protože se domnívám, že takové konverzní možnosti TCX tabulky nikdy nebudou mít.

Jsem zvyklý dělat programy tak, že si stanovím cíl, pak podle toho napíši dokumentaci a nakonec podle dokumentace naprogramuji samotný program. Tím jsem definitivně hotov. Na výsledném programu nemám potřebu většinou nic měnit. S výjimkou drobné úpravy encTEXu z února 2003 předpokládám, že tomu tak bude i v tomto případě. Považuji tedy projekt za skončený a úkol splněný a budu se pouze starat o to, aby encTEX zůstal v tomto stavu zachován i nadále, maje tím na zřeteli makra uživatelů, kteří to začali používat a už nebudou chtít ve svých dílech kvůli přechodům na novější verze nic měnit.

V této souvislosti bych chtěl připomenout, že přechod z původní verze encTEXu (z roku 1997) na současnou verzi je stoprocentně zpětně kompatibilní. Kdo si zvykl encTEXem nastavovat hodnoty *xord* a *xchr* vektorů, ten může tuto vlastnost naprosto stejně využívat i nadále.

V lednu 2003 jsem nabídl encTEX na českém diskusním listu. Na základě tohoto oznámení začal se mnou velice ochotně spolupracovat pan David Nečas (Yeti). Jednak encTEX testoval na systému, kde textové editory skutečně pracují s kódováním UTF-8. Podotýkám, že já takový systém zatím nepoužívám, takže jsem encTEX dělal jen „teoreticky“.

Pan David Nečas doplnil pro distribuci encTEXu kódovací tabulky UTF-8 do kódování T1 (podle Corku, používané hlavně v LATEXu) a dále doplnil kódovací tabulky velkého množství matematických a jiných znaků, které mapoval na kontrolní sekvence. Vytvořil si pro tyto potřeby pythonský skript, který čte definici znaků podle Unicode [4] a vytváří tabulku používající primitivy `\mubyte` vhodnou pro encTEX. Tento skript je v distribuci encTEXu také obsažen.

[2] To je ovšem taky rozšíření TEXu, které vede k programu, jejž už nesmíme nazývat TEX. Aktéři té diskuse si to asi moc neuvědomovali.

Panu Nečasovi tímto velmi děkuji za příkladnou spolupráci. Velmi mile mě též potěšily jeho html stránky [3], které se podrobně věnují encTEXu a dají se použít jako on-line dokumentace. Výhoda těchto stránek je mimo jiné v tom, že jsou psány ve výrazně lepší angličtině, než bych byl schopen stvořit já, samouk, který angličtině nikdy moc nerozuměl.

V únoru 2003 jsem nabídl encTEX mezinárodní TEXové komunitě. Rozporuplné reakce včetně laického názoru, že konverze přes aktivní znaky je dostačující, jsem tak trochu čekal. Po určité delší diskusi na `texlive@tug.org` byl nakonec encTEX akceptován Olafem Weberem, který dělá nové verze web2c TEXu. Odtud to přebírá Thomas Esser do teTEXu. Ten s encTEXem taky souhlasil. Fabrice Popineau to přebírá do fpTEXu a už prý udělal nějaké verze, které encTEX obsahují. Hans Hagen vyslovil přání, že by encTEX rád využil ve svém ConTEXtu.

Olaf Weber přislíbil, že encTEX zařadí do další verze web2c s označením 7.5.3. V únoru existovala verze 7.5.2 a podle ní jsem poslal Olafovi záplaty. Bohužel od února do konce května (kdy píšu tento článek) nová verze 7.5.3 ještě nezačala existovat.

Záplaty, které jsem poslal Olafovi, jsou součástí distribuce encTEXu na [1]. Kromě samotného encTEXu implementují ještě možnost inicializovat encTEX z příkazové řádky parametrem `-enc`. Bez použití tohoto parametru jsou primitivy encTEXu nepřístupné, což umožní konzervativcům pracovat s tímto programem jako s klasickým TEXem. Záplata samozřejmě obsahuje doplnění dokumentace a nápovědy ke změněným programům web2c distribuce. Také jsem v této záplatě řešil možnou spolupráci s TCX tabulkami, které, stejně jako encTEX, pracují se stejnými vektory *xord*, *xchr*. Je tedy možné encTEXem tyto vektory například při generování formátu nastavit, TCX tabulkami je pak třeba přepsat na jiné hodnoty a později zase encTEXem číst a nastavit třeba zase jinak.

Chystám se na konferenci euroTEX do Francie, kde bych se pokusil osobně představit encTEX mezinárodní TEXové komunitě.

Malé zamyšlení

Věřím, že encTEX si najde své uživatele a že má práce na něm vynaložená bude aspoň někomu užitečná.

Neočekávejte ale od encTEXu univerzální řešení, pomocí kterého byste mohli okamžitě a pohodlně používat v jednom dokumentu jazyky celého světa. Například vzory dělení pracují jen v 8bitovém kódování, fonty je možné používat jen 8bitové atd. Hlavním cílem encTEXu bylo zachovat důslednou zpětnou kompatibilitu s dokumenty, které byly dříve zpracovány 8bitovým TEXem, nyní jsou konvertovány do UTF-8 a je potřeba je zpracovat znovu. Toto řešení také umožňuje přežít různým balíčkům maker, které byly postaveny na nějakém pevně zvoleném interním 8bitovém kódování (`csplain` s ISO-8859-2, LATEX s kódo-

váním T1) i za situace, kdy se bude UTF-8 kódování používat stále častěji. V neposlední řadě encTEX umožní v souvislosti s nástupem UTF-8 standardu přežít samotnému 8bitovému TEXu. O to mi šlo především.

Mnohojazyčné řešení si ještě vyžádá dlouhou a náročnou práci a není to nic jednoduchého. Uživatelé často nabývají mylného dojmu, že když nyčkon mají v systému ten Unicode, tak mohou okamžitě sázet ve všech jazycích světa a hlavně *dobře* sázet. To vůbec není pravda. UTF-8 samotné neřeší jednotlivé fiškuntálie a typografické lahůdky různých jazyků, neřeší standardní obsazení znaků v běžných fontech, které by bylo možné pro případný přenos mnohojazyčných dokumentů očekávat na všech počítačích atd. Donald Knuth v jednom rozhovoru pro TUGboat [6] na otázku po „Unicode-TEXu“ říkal:

Yeah! It seems so difficult. I'm a great fan of Unicode, but I also know enough about it to know that it's incredibly complicated, and that I would never have the system based on Unicode that I would be able to say has no bugs in it because of the extra complexity. Well, I like TEX, I like having a program that it, if not 100% reliable, it's 99,9999—it's as realiable as anything...

As you know, Unicode 3.0 which wil be coming out early next year, really covers almost all the languages of everyone alive today; they have filled in the last gaps, they've got Burmese and the Maldive Islands, Sri Lanka, the places where the political difficulties were; they have several thousand extra Vietnamese and Chinese characters and so on. And Yi, and Mongolian, and native American—various Inuit and Algonquian languages—are all there now.

But each of these languages has special difficulties involved in the typesetting. *It is not just a matter of getting the symbols; all kinds of ligatures and things must go in, and rewriting of characters.* Many of the languages have no spaces between words, and special hyphenation conversions and a total user community of a few thousand...

So it's going to be hard to support this commercially; it's surely going to be a volunteer effort. The effort is not only an order of magnitude more difficult than what I had to do, but it also has to be done pretty much as labour of love, as I did it. So it looks to be a while before it could converge like that—not that it's impossible, but I myself wouldn't be in position to bless it. All I can do is provide an example of one of the world's nearly bug-free programs so that other people can try to emulate the good points and correct the bad points.

Reference

[1] `http://petr.olsak.net/enctex.html`
[2] `ftp://math.feld.cvut.cz/pub/olsak/enctex/encdoc.pdf`
[3] `http://trific.ath.cx/tex-mf/enctex/`
[4] `http://www.unicode.org/Public/UNIDATA/NamesList.txt`
[5] `ftp://math.feld.cvut.cz/pub/cstex/base/cstrip.tar.gz`

[6] U.K.TUG, Oxford, Sunday, 12 September 1999, *Question & Answer Session with Donald Knuth*, In: TUGboat, Volume 22 (2001), No. 1/2, pp: 15–19.
[7] Petr Olšák. *EncTEX – změny konverzních tabulek v TEXu.* Zpravodaj Československého sdružení uživatelů TEXu, **3** (7), 109–118 (1997)
[8] Petr Olšák. *Putování písmene ř z klávesy na papír.* Zpravodaj Československého sdružení uživatelů TEXu, **3** (7), 129–140 (1997)

Summary: New encTEX – the UTF-8 encoding in TEX

The UTF-8 encoding keeps the standard ASCII characters unchanged and encodes the accented letters of our alphabets in two bytes. The standard 8bit TEX is not ready for the UTF-8 input because it has to manage the single character as two tokens. It means you cannot set the `\catcode`, `\uccode`, etc. to these single characters and you cannot do `\futurelet` of the next character in normal sense. The second version of my encTEX solves these problems.

The encTEX is full backward compatible with the original TEX. It adds ten new primitives by which you can set or read the conversion tables used by input processor of TEX or used during output to the terminal, log and `\write` files.

The second version gives possibility to convert the multi-byte sequences to one byte or to a control sequence. You can implement up to 256 UTF-8 codes as one byte and unlimited number of other UTF-8 codes as control sequences. All internals in 8bit TEX are working in the same way as if "normal one byte encoding" of input files is used.

I think that the UTF-8 encoding will be in more common use. In such situation, there is no other way than to modify the input processor of TEX otherwise the 8bit TEX will die in a short time.