全角英数字→半角英数字, 半角カタカナ→全角カタカナ等の変換を行うツール* —zen2han—

古川 正恵†
Tadamegu Furukawa

平成 18 年 9 月 9 日‡

概 要

TEX でタイプセットしていると, テキストで頂いたファイルを TEX ファイル用にマークアップする必要が生じたりワープロで作成した文書をテキストファイルに変換 (あるいは保存) したものを TEX ファイル用にマークアップする必要が生じる場合があります。

テキストファイルに TEX のコマンドをマークアップする処理は, それほど大変な作業ではありませんが, マークアップしていてうんざりすることは

- 英数字に全角文字を用いているケースが多々ある。(例えば 「This is a pen.」を「Ｔｈｉｓ　ｉｓ　ａ　ｐｅｎ.」としている等)
- 半角カタカナを用いている。(TEX ではエラーになります。)
- JIS の規格外の漢字コードを使用している。(某社特有の罫線コードや○の中に数字が入った文字等)
- カタカナの長音記号 (ー) とマイナス記号 (−) を誤って使用している場合が結構ある。

といった誤りが結構あるということです。このような誤り (必ずしも誤りとは言えないかもしれませんが) をいちいち手で直す事に嫌気がさして

- 全角英数字を半角英数字に変換
- 半角カタカナを全角カタカナに変換
- 長音記号 (ー) とマイナス記号 (−) の使用誤りを検出して自動訂正 (100% とはいえませんが)

等の処理を行うツールを UNIX 上で作成していました。今回は UNIX で動作していたツールを MS-DOS(WIN32, djgpp[1]) で動作するように修正しました。また, 機能の一部を拡張して全角文字の変換テーブルを外部ファイルに持たせる事によって, ある程度柔軟な変換を行えるようにしました (例えば○の中に数字の入った文字を TEX のマクロに置き換える事も可能です)。

1 機能概要

zen2han はテキストファイル中の特定のコードを変換するツールです。元々は全角の英数字を半角に変換する為に UNIX 上で awk + csh のスクリプトで作成したものです。そのため zenkaku to(2) hankaku という名前になっています。その後, 半角カタカナを全角カタカナに変換する機能等を追加すると同時に C 言語[2]で作り直しました。現在は概ね以下の機能を有しています。

---

*半角とか全角という記述はある意味では正しくありませんが, このドキュメントでは 1 バイトコードを「半角」, 2 バイトコードを「全角」と記述します。(但し EUC では半角のカタカナは 2 バイトコードなので, 正確にはこの記述も正しくはありません。)

†tfuruka1@nifty.com
http://www.vector.co.jp/authors/VA001687/

‡本ドキュメントは 1997 年頃に記述したものです。その為、現在の環境とは大きく異なっている部分があります。本来であれば、大幅に修正及び加筆を行なうべきなのですが、時間的な制約の為、必要最低限な修正のみ行なっています。

[1]現在は WIN32 のみです。

[2]現在は Emacs Lisp で作成したものもあります。

- 全角英数字を半角英数字に変換します。オプションスイッチによって, 無効にする事も可能です。
- 半角カタカナを全角のカタカナに変換する。この時に濁音や半濁音の処理も行います。
- カタカナの長音記号「ー」とマイナス記号「−」の使用誤りを推測して適切なコードに置き換えます (100%とはいえません)。この機能はオプションスイッチによって無効にすることも出来ます。
- 半角の英数字と全角文字の間に空白文字がなかった場合には空白文字を挿入する。この処理はおそらく嫌いな方もいらっしゃると思いますので[3], オプションスイッチによって無効にする事が出来ます。
- “,” の後に空白文字がない場合に空白文字を挿入する。この機能も TEX ファイルに対して処理を行うと, 不具合が生じる可能性がありますので[4], オプションスイッチによって無効にする事が出来ます。
- 全角変換定義ファイルを記述する事によって, ある程度柔軟な変換を行うことが可能。例えば, デフォルトの定義ファイルでは句読点は半角のカンマ「,」と全角の白丸「。」を使用するように定義しています。これは私の好みですが, 他の設定に変える事も出来ます[5]。

2 インストール方法

このドキュメントを読んでいるという事はアーカイブを展開したという事だと思います。本ドキュメントが存在しているディレクトリに `bin` という名前のディレクトリがあるはずです。

```
cd bin
```

としてディレクトリを移動して下さい。`bin` ディレクトリには更に入手したプラットホームによってディレクトリ名は異なりますが, プラットホームと同じ名前のディレクトリ名があるはずです。

`cd win32` まはた `cd dos` または `cd djgpp`

のいずれかのコマンドを入力してディレクトリを移動して下さい。するとカレントディレクトリに “`zen2han.exe`”, “`z2h_inst.exe`”, “`zen2han.def`” というファイルがあります。ここから先のインストール方法は二つの方法があります。一つは環境変数を設定する方法, もう一つは専用のインストーラ (`z2h_inst.exe`) を使用する方法です。

2.1 環境変数を設定する場合 (専用インストーラを使用しない場合)

先ず, “`zen2han.exe`” をパスを張ってある適当なディレクトリにコピーして下さい。次に “`zen2han.def` を適当なディレクトリにコピーして下さい。通常は “`zen2han.exe`” と同一のディレクトリと同じで構わないと思います。次に環境変数 `ZHDEF` に “`zen2han.def`” の格納場所をフルパスで指定します。今,

```
c:\user\local\bin\zen2han.def
```

が “`zen2han.def`” のフルパスだと仮定します。WIN32 と DOS の場合は “`autoexec.bat`” 等に

```
set ZHDEF=c:\user\local\bin\zen2han.def
```

[3]私は NTT 系の TEX も使用するのですが, NTT の場合, 漢字と ASCII の間に空白文字を入れないと Overfull hbox が多発する場合があるので, このようになっています。
[4]例えば, マクロの引数で “,” をセパレータとして使用している場合。
[5]私が普段書いている文書は英語と日本語が混在していますので, “,” で統一したほうが見やすいからです。

と記述して下さい。djgpp の場合は若干方法が異なります。先ず, 既に gcc 等がインストール済み等の理由で環境変数 DJGPP が設定されている場合には, 環境変数 DJGPP が示している環境変数ファイルに

```
+ZHDEF=c:\user\local\bin\zen2han.def
```

の一行を追加して下さい。環境変数 DJGPP が設定されていない場合には適当なファイルを適当なディレクトリに作成して, ファイルの内容を環境変数 DJGPP が設定してある場合と同じ一行にして下さい。ここではこのファイル名を

```
c:\djgpp\djgpp.env
```

と仮定します。そして, このファイルを環境変数 DJGPP に設定します。この例の場合は

```
set DJGPP=c:\djgpp\djgpp.env
```

の記述を autoexec.bat 等に追加します。これで環境変数を設定する場合のインストール作業は終了です。

2.2 専用インストーラを使用する場合

環境変数の設定が面倒であるとか, 環境変数領域があまりないのでもったいないという方は専用インストーラを使用してインストールする事が出来ます。専用インストーラは zen2han.def の場所を zen2han.exe に直接書き込みますので, 環境変数は必要ありません。例えば,

```
c:\usr\local\bin
```

ディレクトリにインストールしたい場合はコマンドラインから

```
z2h_inst c:\usr\local\bin
```

と入力するだけでインストールが完了します。この時にインストール先ディレクトリは必ずフルパスで指定して下さい。相対パスでは不具合が生じます。またディレクトリの最後の文字にはディレクトリセパレータ文字 (“\”, “/” 等) を指定しないで下さい。

専用インストーラでインストールした場合は, “zen2han.def” の格納ディレクトリを変更した場合に再度 “z2h_inst” によるインストールが必要になります。カレントディレクトリを “zen2han.exe” と “zen2han.def” が存在しているディレクトリにして再度, 同様に “z2h_inst” を実行すれば, インストール出来ます。“zen2han.exe” は何度でも z2h_inst.exe” によって書き換える事が出来ます。

また, インストーラを使用してインストールした場合でも, 環境変数 ZHDEF が設定されていた場合には, 環境変数を優先します。

3 使い方

コマンドの形式は以下の通りです。

zen2han [-9] [-X][-s] [-d] [-a] [-c] [-t*DefFile*] {*filename* ... | -}

filename で指定されたテキストファイルの内容を 1 節で記述した通りに変換します。変換した結果は *filename* に-を指定していない場合及び-s オプションを指定していない場合は同じファイルに対して書き込みます。変換前のファイルは拡張子を “bak” に変換して保存します。従って拡張子が “bak” のファイルに対しては変換作業を行う事は出来ません。*filename* の指定は UNIX ライクなワイルドカードを使用する事が出来ます。例えば, ファイル名に数字が含まれているファイルを対象にする場合には

```
zen2han *[0-9]*
```

等と指定します。詳しくは他の文献を参考にして下さい。また, オプションスイッチを指定する事によって, 変換の方法を制御するする事が出来ます。使用出来るオプションスイッチは以下の通りです。

`-9` 全角の数字を半角数字へ変換しません。

`-X` 全角のアルファベットを半角数字へ変換しません。

`-s` 変換結果を同一ファイルではなく, 標準出力へ出力します。

`-d` カタカナの長音記号「ー」とマイナス記号「−」の推論訂正を行いません。意図的にこれらの記号を入れ換えて使用している時などに指定して下さい。

`-a` ASCII 文字と漢字の間にスペースが無い場合でもスペースを挿入しません。デフォルトでは私の好みでスペースが一個挿入されます。

`-c` “,”(Comma) の次の文字スペースでない場合でもスペースを挿入しません。デフォルトでは “,” の後には一個のスペースが挿入されます。LaTeX ファイルに対して実行すると, マクロの引数のセパレータの “,” の後ろにスペースが挿入されてしまい, 都合が悪くなる場合があります。

`-t`*Def* *Def* で指定されたファイルを全角変換定義ファイルとして使用します。定義ファイルを複数個持っていて, 用途によって使い分ける時等に指定します。尚, `-t` と *Def* の間にはスペースを入れないで指定して下さい。

filename 変換するファイル名を指定します。UNIX ライクなワイルドカードを指定出来ます。“`-`” を指定した場合は標準入力から読み込みます。

“`test.txt`” というファイルをコピーしてコピーしたファイルに対して zen2han を実行してみる事によって概ねの動作を理解して頂けると思います。

4 定義ファイルの記述方法

`zen2han` は全角英数字から半角英数字への変換, 半角カタカナから全角カタカナへの変換, カタカナの長音記号「ー」とマイナス記号「−」の推論訂正は, 内部で行いますが, その他の全角文字の変換の定義は「全角変換定義ファイル」と呼ばれるファイルに記述します (デフォルトのファイル名は “`zen2han.def`”)。

この定義ファイルの書き方は, 基本的には

/from/to/

のように全角文字と文字列を `/` (スラッシュ) で区切って記述します。`zen2han` では *from* と一致する全角文字を見つけた時に *from* を *to* に変換します。この時,

- *from* は全角文字 (2 バイトコード) 又は半角文字。但し一文字だけ指定可能です。
- *to* は 39 バイト以内の文字列

である必要があります。 *to* を 記述しなかった場合には *from* を削除する事になります。例えば PC98 ○1 特有の○の中に数字が入ったコードは TeX ではエラーになりますので,

```
/(1)//
```

のように記述[6]した場合は○の中に 1 と入っているコードを見つけるとそのコードを削除します。また, この記述は, 必ず先頭カラムから記述して下さい。先頭に空白を空けないで下さい。

もし, *to* に /(スラッシュ) を含ませたい場合はセパレータを+ にして下さい。例えば, 全角のスラッシュを半角のスラッシュに変換する場合には以下のように記述します。

```
+／+/+
```

これを

```
//／///
```

と記述するのは間違いです。/ も + も *to* に含めたい場合には !, ", %, &, *, - 等の記号をセパレータに使用して下さい。

もう少し正確に記述すると, UNIX(EUC の場合) の場合はセパレータが *from* にも *to* にも現れなければ, どんな文字 (ASCII-CODE) を使用してもかまいません。zen2han は先頭の 1 バイト目をセパレータ文字と認識します。MS-DOS(SHIFT-JIS) の場合は 2 バイトコードの 2 バイト目のコードが ASCII-CODE と同じになる可能性がありますので, 0x40 未満のコードである必要があります。取り合えず, 意味が分からない場合は上に示したコードを使用して下さい。

先頭カラムが # で始まる行はコメント行として処理しますので, 自由に注釈等を記述する事が出来ます。

また, この定義ファイルはサンプルなので, LaTeX の事を考慮した変換は行っていません。

例えば, 漢字の「Σ」や「∫」を数式モードの $\sum$ や $\int$ に変換するように

```
/Σ/$\displaystyle\sum$/
/∫/$\displaystyle\int$/
```

というような記述をしておくのも場合によっては便利かもしれません。また, ○の中に数字の入った文字も

```
/(1)/\○\llap{１}/
```

のような記述[7]をしておくと, TeX でコンパイルした時にそれなりに表示されます[8]。

5 その他

5.1 ワイルドカード展開ルーチンに関して

zen2han はファイル名にワイルドカードを指定出来ますが, このワイルドカード展開ルーチンは MS-DOS や WIN32 API で提供している FindFile 系のワイルドカード展開ルーチンよりも, 柔軟な指定が可能です。UNIX ライクなワイルドカードの展開を行えます。もともとこのルーチンを作成したのは私ではありません。記憶が定かではないのですが, 多分 1990 年代前半の C マガジンの付録に添付されていた PDS(日本国内では PDS は有り得ないようですが, PDS と明記されていました。) です[9]。そのソースを元に私が WIN32 と djgpp に対応させたものです。また, UNIX に於いても WIN32, MS-DOS, djgpp とのファイルの共用を考慮して同様な動作をするようにしました。(実際, UNIX の場合は何も処理を行っていません。受け取った引数をそのままリターンするようになっています。)

[6] (1) と記述したのは本当は○の中に 1 と入っているものと思って下さい

[7] くどいようですが, (1) は○の中に 1 という数字が入っている文字だと思って下さい。

[8] 丸数字に関してはこんなダサダサな定義ではなく, もっと素晴らしい方法が発表されています。

[9] たしか, どこかの大学のプロジェクトだったと思います。

5.2 Windows95 と GO32.EXE に関して

私の使用しているマシン (PC-856RV) 固有の問題かもしれませんが, Windows95 の DOS 窓と GO32 の相性が非常に悪いです。GO32 を実行すると, 不正なアクセスを検出して GO32 が異常終了することが多々あります。そのような場合には「MS-DOS プロンプト」のプロパティの詳細設定で「MS-DOS プログラムに Windows を検出させない」ように設定することにより, 回避出来ます。但しこの設定を行ってしまうと, DOS 窓から WIN32 アプリケーションが動作しなくなります。因みに私は, WIN32 を実行出来る DOS 窓と出来ない DOS 窓をショートカットで作成しておいて, それぞれを使い分けています。

6 コンパイル方法

コンパイルの確認は VC++2.0, MSC Version 5.1, djgpp の gcc Version 2.7.1, EWS4800 320EX の cc コマンド, 同マシンで gcc Version 2.7.2, FreeBSD の gcc Version 2.6.8 で行っています。make コマンドは MicroSoft の nmake, GNU の gmake, UNIX の make の何れでも動作する事を確認しています。

基本的なコンパイル方法は `makefile` の当該個所のコメントを外して (またはコメントアウトして), インストールディレクトリ等を自分の環境にあわせて書き換えた後に “make install” するだけです (または “make” 後, “make install”)。殆どの機種で, これだけの処理を行う事によってコンパイル出来ると思います。但し私は BSD 系のコンパイラを所有していませんので[10], BSD 系のマシンの場合は若干, ソースファイルの修正が必要かもしれません[11]。

本アーカイブのソースファイルは SHIFT-JIS で記述していますので UNIX でコンパイルする場合には, 全てのソースファイルと `makefile`, `zen2han.def` ファイルを何らかの方法で EUC に変換する必要があります。

コンパイルオプションや, その他の情報に関しては `makefile` に記述していますので, そちらを参照して下さい[12]。

7 最後に

本ツールは FSW です。入手したアーカイブのままであれば, 転載及び頒布は自由に行って下さってかまいません。また, 新たに作成した「全角変換定義ファイル」であれば, アーカイブに追加する形で含めても構いません。

また, 本ツールを使用した事によって発生した如何なる問題や障害に対しても作者は一切の責任を負いません。あくまで使用者の責任に於いて使用して下さい。

バグレポート, 使用レポート, 改善要求等は以下のメールアドレスへお願いします。

tfuruka1@nifty.com

[10] FreeBSD は使用していますが, コンパイラは gcc なので, BSD 系のコンパイラではありません。

[11] `string.h` と `strings.h` の違い等

[12] 最初のヴァージョンを作成した頃はあらゆるマシンを使用出来る環境にあったので, 厳密にコンパイルチェックを行う事が出来たのですが, 現在は Win32(MSVC) でのコンパイルしか確認できていません。