がらくたシリーズ: **ZEN2HAN**

JIS X0201(所謂半角カナ) → JIS X0208(全角カナ)
全角英数字→ ASCII
その他の文字列変換を行うツール

tfuruka1@nifty.com

このドキュメントは, 「がらくたシリーズ:ZEN2HAN」について記述しています。 ZEN2HAN は所謂半角カタカナ (JIS X0201) から所謂全角カタカナ (JIS X0208) への 変換を目的として作成した elisp です。

このドキュメントの版数は $Revision: 1.6 $ です。

Table of Contents

このドキュメントは, 「がらくたシリーズ:ZEN2HAN」について記述しています。 ZEN2HAN は所謂半角カタカナ (JIS X0201) から所謂全角カタカナ (JIS X0208) への 変換を目的として作成した elisp です。

このドキュメントの版数は $Revision: 1.6 $ です。

1 要旨

「がらくたシリーズ」は, 使い捨てで作っていたツールを捨てるのは勿体無いとの趣旨でパッケージ化したシリーズです。というか, 最近, 物忘れが激しくなってきて, 昔のように捨てては作るといった事が出来なくなってしまいました。‘zen2han.el’は, 元々は‘.emacs’に直接記述していたものを, 切り出して作成したものです。

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困る事は,

- 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 ではエラーになってしまいます。
- 機種依存文字を使用している。某社特有の罫線コードや○の中に数字が入った文字等です。
- 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。

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

等の処理を行うツールを UNIX 上で作成していました。そのうち, 主に使用するプラットフォームが Windows になってきたので, UNIX で動作していたツールを Win32 のコンソール (WIN32, djgpp) 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等)。これが, 1997 年の事です。

それと同時に, 当時覚えたての Emacs Lisp でも実現出来るといいな〜と思って, ‘.emacs’に直接記述して使用していました。Emacs は NEmacs の頃から使用していたので, その当時でも Emacs 暦は長かったと思いますが, Elisp を書いたのは, 初めてだったと思います (今も elisp はへなちょこですが, その当時の elisp はとても見られたもんじゃありませんでした)。その Elisp を焼きなおしたのが, 「がらくたシリーズ: ZEN2HAN」という事になります。

2 機能概要

zen2han はバッファ中の特定のコードを変換する関数の集まりです。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 言語で作り直したものを elisp に移植しました。現在は概ね以下の機能を有しています。

- 全角英数字を半角英数字に変換します。
- 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)。この機能はカスタマイズ変数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[1]。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, カスタマイズ変数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[2] 可能性がありますので, カスタマイズ変数によって無効にする事が出来ます。
- その他の変換も, カスタマイズ変数を記述する事によって, あらゆる変換が可能。例えば, デフォルトの定義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[3], 他の設定に変える事も出来ます。というか, 単に変換テーブルを持っているだけなので, どのようにも出来るという事です。

[1] 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, このようにしています。

[2] 例えば, TEX マクロの引数で,をセパレータとして使用している場合。

[3] 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。

3 インストール

3.1 ‘zen2han’のインストール

‘zen2han’のインストールは非常に容易です。‘zen2han.el’をを環境変数 EMACSLOADPATHに含まれるディレクトリにコピーするだけです。必ずしも, このディレクトリにインストールする必要はありませんが, 他のディレクトリにインストールした場合は, 後に説明する ‘~/.emacs’の設定を読み替えて下さい。

‘zen2han.el’のインストールが完了したら, ‘~/.emacs’のどこかに

```
(load "zen2han")
```

を追記して下さい。‘zen2han.el’を EMACSLOADPATH以外のディレクトリにインストールした場合は, パスを適切に指定して下さい。この設定ですと, zen2han を使用しない場合でも, 常にメモリに ‘zen2han.el’をロードします。zen2han を使用する時だけ, ロードする場合は, 上記の代わりに

```
(autoload 'zen2han-region "zen2han" "全⇔半" t)
(autoload 'zen2han-buffer "zen2han" "全⇔半" t)
(autoload 'zen2han-all-fill-paragraph-region "zen2han" "全⇔半" t)
(autoload 'zen2han-all-fill-paragraph-buffer "zen2han" "全⇔半" t)
(autoload 'zen2han-chop-line-end-space "zen2han" "全⇔半" t)
```

と, 追記して下さい。

3.2 info のインストール

これが読めているという事は, info のインストールが終わっているという事のような気がするのですが...

1. ‘zen2han.info’ファイルを ‘INFOPATH’に設定されているディレクトリにコピーしてください。
2. そのディレクトリの ‘dir’ファイルに以下の記述を追加してください。

   ```
   * zen2han: (zen2han).     Convert JIS X0201 to JIS X0208 etc.(ja)
   ```

   また, ‘install-info’でインストールする事も可能です。以下の例は, カレントディレクトリが ‘INFOPATH’に設定されているディレクトリの場合の例です。

   ```
   install-info ./zen2han.info ./dir
   ```

4 使い方

zen2han の使用方法は, 至って簡単です。任意のバッファで, `M-x`に続いて関数名の入力を行うだけです。

4.1 文字列変換系

文字列変換系の関数を以下に記述します。文字列変換の内容は, Chapter 2 [機能概要], page 3 に記述している通りです。

zen2han-region *start end* [Function]
指定リージョンの文字列の変換を行います。当たり前の事ですが, 意図して, 半角カナや, 全角英数字を記述している範囲にこの処理を行わないように注意して下さい。

zen2han-buffer [Function]
カレントバッファの文字列の変換を行います。

4.2 段落の文字詰め等

zen2han には, 文字列変換とは全く関係ありませんが, 段落の文字詰めを行う関数も用意しています。この関数は, 私の好みで, 本パッケージに入れています。*zen2han-region* で, 所謂全角英数字から ASCII 文字や, 所謂半角カタカナから全角カタカナに変換を行うと, 文字列長が変わるため, 段落の行長がバラバラになります。段落毎に *fill-paragraph* を行うのが面倒なので, 以下の関数を用意しています。

zen2han-all-fill-paragraph-region *start end* [Function]
指定リージョンの全ての段落の文字詰め (*fill-paragraph*) を行います。この関数を使用する場合は、「全て」の段落に対して処理を行う事に注意して下さい。目視の「段落」と Emacs の「段落」は違う場合があります。

zen2han-all-fill-paragraph-buffer [Function]
カレントバッファの全ての段落の文字詰め (*fill-paragraph*) を行います。その他は, *zen2han-all-fill-paragraph-region* と同様です。

zen2han-chop-line-end-space [Function]
段落の文字詰めとは, 趣旨が異なりますが, 行末の空白文字 (SPACE, TAB) を削除します。対象は, カレントバッファ全体です。Emacs では, 改行マークが見えない為, 行末に余分な空白があっても, 直ぐには判らない事があります。よっぽどの事がない限りは, 行末に意図して, 空白文字を置く事は無いと思います。

4.3 その他

その他に '`zen2han.el`'で定義している関数を以下に列挙します。

zen2han:rep-str *from flen to tlen* [Function]
zen2han-region, *zen2han-buffer* から呼び出され, カレントバッファの文字列の変換を行う下請け関数です。カレントバッファ全体に対して処理を行いますので, 範囲を指定する場合は, 呼び出し元で, ナローイングしておく必要があります。この関数はインタラクティブに呼び出す事は出来ません。引数の意味は以下の通りです。

`FROM` 変換元文字列の文字列の集合を文字列で表します。文書で書くと意味が判らないと思いますが, 例えば, Ａを`A`, Ｂを`B`, Ｃを`C`に変換する処理を考えた場合, この引数に,

ＡＢＣ

を指定します。

FLEN 変換元文字列の変換単位の文字列長を指定します。先と同じ変換を考えた場合この引数には

```
(length "Ａ")
```

等と指定します (即値で 1 を指定しても良いですが, 過去に痛い目にあった事があるので, 関数で文字列長を得た方が無難です)。つまり, この関数では, 異なる文字列長の集合を指定出来ないという事になります。

TO 変換先文字列の文字列の集合を文字列で現します。書式は, 変換元文字列と同様です。先と同じ変換を考えた場合, この引数には,

```
ABC
```

を指定する事になります。

TLEN 変換先文字列の変換単位の文字列長を指定します。先と同じ変換を考えた場合, この引数には

```
(length "A")
```

等と指定する事になります。

例えば, 全角文字列のﾊﾟをパに変換する場合は,

```
(zen2han:rep-str "ﾊﾟﾋﾟﾌﾟﾍﾟﾎﾟ"
                 (length "ﾊﾟ")
                 "パピプペポ"
                 (length "パ"))
```

と記述します。上記の例はハ行だけ記述していますが, 実際は全行記述すると思います。

zen2han-version-show [Function]

zen2han のヴァージョンをミニバッファに表示します。

zen2han-id-show [Function]

zen2han のヴァージョンコントロール ID をミニバッファに表示します。

5 カスタマイズ変数

zen2han の文字列変換は, 全てカスタマイズ変数で制御する事が可能です。以下に zen2han で用意しているカスタマイズ変数を列挙します。

zen2han-buffer, *zen2han-region* は次に列挙する変数を *zen2han-z2h-list*, *zen2han-h2z-list*, *zen2han-cnv-dullness-list*, *zen2han-cnv-kigou-list*, *zen2han-conv-other-alist*, *zen2han-conv-regexp-alist* の順番に処理しています。

zen2han-z2h-list [Variable]

所謂全角から半角へ変換する為のリストです。リストの要素は, 全角文字の集合, 全角文字の変換単位の長さ, 半角文字の集合, 半角文字の変換単位の長さです。この値に'()を指定した場合は変換を行いません。このリストの初期値は以下の通りです。

```
(list (concat "Ａ Ｂ Ｃ Ｄ Ｅ Ｆ Ｇ Ｈ Ｉ Ｊ Ｋ Ｌ Ｍ Ｎ Ｏ Ｐ Ｑ Ｒ Ｓ Ｔ Ｕ Ｖ Ｗ Ｘ Ｙ Ｚ"
              "ａ ｂ ｃ ｄ ｅ ｆ ｇ ｈ ｉ ｊ ｋ ｌ ｍ ｎ ｏ ｐ ｑ ｒ ｓ ｔ ｕ ｖ ｗ ｘ ｙ ｚ"
              "０ １ ２ ３ ４ ５ ６ ７ ８ ９")
      (length "Ａ")
      (concat "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
              "abcdefghijklmnopqrstuvwxyz"
              "0123456789")
      (length "A"))
```

zen2han-h2z-list [Variable]

半角から全角へ変換する為のリストです。リストの要素は, 半角文字の集合, 半角文字の変換単位の長さ, 全角文字の集合, 全角文字の変換単位の長さです。この値に'()を指定した場合は変換を行いません。このリストの初期値は以下の通りです。

```
(list (concat "。「」、・ヲァィゥェォャュョッｰ"
              "ｱｲｳｴｵｶｷｸｹｺｻｼｽｾｿﾀﾁﾂﾃﾄﾅﾆﾇﾈﾉﾊﾋﾌﾍﾎ"
              "ﾏﾐﾑﾒﾓﾔﾕﾖﾗﾘﾙﾚﾛﾜﾝﾞﾟ")
  (length "ｱ")
  (concat "。「」、・ヲァィゥェォャュョッーアイウエオカキクケコサシスセソタチツテトナニヌネノハヒフヘホマミムメモヤユヨラリルレロワンﾞﾟ"
          "ｱｲｳｴｵｶｷｸｹｺｻｼｽｾｿﾀﾁﾂﾃﾄﾅﾆﾇﾈﾉﾊﾋﾌﾍﾎﾏﾐﾑﾒﾓﾔﾕﾖﾗﾘﾙﾚﾛﾜﾝﾞﾟ")
  (length "ｱ"))
```

このリストには所謂半角カナが含まれています。

zen2han-cnv-dullness-list [Variable]

濁音及び, 半濁音を変換する為のリストです。リストの要素は, 変換元文字列の集合, 変換元文字列の変換単位の長さ, 変換先文字列の集合, 変換先文字列の変換単位の長さです。この値に'()を指定した場合は変換を行いません。このリストの初期値は以下の通りです。

```
(list (concat "ﾊﾟ ﾋﾟ ﾌﾟ ﾍﾟ ﾎﾟ "
              "ｶﾞ ｷﾞ ｸﾞ ｹﾞ ｺﾞ ｻﾞ ｼﾞ ｽﾞ ｾﾞ ｿﾞ "
              "ﾀﾞ ﾁﾞ ﾂﾞ ﾃﾞ ﾄﾞ ﾊﾞ ﾋﾞ ﾌﾞ ﾍﾞ ﾎﾞ ")
      (length "ﾊﾟ ")
      (concat "パピプペポ"
              "ガギグゲゴザジズゼゾ"
              "ダヂヅデドバビブベボ")
      (length "パ"))
```

この中には, 所謂半角カナ (JIS X0201) に対する変換は含まれていませんが, 先に半角カナは全角カナ (JIS X0208) に変換されている為です。

zen2han-cnv-kigou-list [Variable]

全角記号から半角記号への変換リストです。リストの要素は *zen2han-z2h-list* と同様です。この値に'()を指定した場合は変換を行いません。このリストの初期値は以下の通りです。

```
(list (concat "！＄｜＾＠｛［］｝＜＞＆／（） "
              "：－＃？＋＝．’＊；　、,. __｀")
      (length "！")
      (concat "!$|^@{[]}<>&/()"
              ":-#?+=.'*; ,,._‘")
      (length "!"))
```

このリストには全角スペースが含まれています。

zen2han-conv-other-alist [Variable]

変換元文字列と, 変換先字列の連想リストです。今まで説明した以外の汎用文字列の変換を行う為に使用します。例えば「(○の中に 1)」を「(1)」に変換する場合は,

```
'(("①" . "(1)"))
```

と指定します。Emacs 上では文字が見えないかもしれません (見えなくて正解なのですが)。機種依存文字を変換する場合に有効だと思います。また, TEX 用に

```
'(("Σ" . "$\\displaystyle\\sum$")
  ("∫" . "$\\displaystyle\\int$"))
```

等と記述すると良いでしょう。なお, この連想リストの値が'()の場合は, 変換を行いません。このリストの初期値は'()です。

zen2han-conv-regexp-alist [Variable]

正規表現で変換する文字列の連想リストです。検索文字列と, 置換文字列を指定します。この設定で,

1. 意味的におかしいーを-に変換しています。意味的におかしいという判断は全角の後の-, ーと半角の後のー等です。ASCII 文字直後のー は-に変換し、カタカナの後の-はーに変換するしています。
2. ,の後に空白の無いものは空白を付加する。
3. ASCII 文字の前後が漢字の場合はそれぞれスペースを挿入する
4. 後ろに空白があると変な物と, 前に空白が有ると変なものは空白を詰めます。これは, 判断が難しいので, 空白が一個のものだけを対象にしています。

このリストの初期値は以下の通りです。

```
'(
  ;; 意味的におかしい「ー」を「-」に変換する。意味的におかしいという判
  ;; 断は全角の後の-,ーと半角の後の「ー」等です。ASCII 文字直後の"ー"は
  ;; "-"に変換し、カタカナの後の"-"は"ー"に変換する
  ("\\([ -~]\\)ー" . "\\1-")
  ("\\([ア-ン]\\)-" . "\\1ー")

  ;; 「,」の後に空白の無いものは空白を付加する
  (",\\([^ ]\\)" . ", \\1")

  ;; ASCII 文字の前後が漢字の場合はそれぞれスペースを挿入する
  ("\\([  -龠]\\)\\([A-Za-z0-9]\\)" . "\\1 \\2")
```

```
("\\([A-Za-z]\\)\\([  -龠]\\)" . "\\1 \\2")

;; 後ろに空白があると変な物と、前に空白が有ると変なものは空白を詰め
;; る。これは、判断が難しいので、空白が一個のものだけを対象にしてい
;; ます。
("\\([〜「。]\\) \\([^ ]\\)" . "\\1\\2")
("\\([^ ]\\) \\([〜」]\\)" . "\\1\\2"))
```

なお, 初期値の値は, もしかすると ‘`zen2han.el`’と一致していない可能性がありますの (勿論, 最初は一致しているんですが, メンテナンスを重ねると一致しなくなる可能性があります ••• という意味です) で, 正確な初期値を知りたい場合は, *describe-variable* で確認して下さい。これらの変数の値を変更する事により, *zen2han-region*, *zen2han-buffer* の挙動を制御する事が可能になります。

6 蛇足

どうでも良い事ですが, 私の作成するソフトウエアには「がらくたシリーズ」と「へなちょこシリーズ」とそれ以外に分類されます。ここでは, その定義 (といっても, 私個人の主観なんですけど) について説明します。

6.1 がらくたシリーズ

がらくたシリーズとは, 所謂, 使い捨てのソフトウエアです。誰でも, 必要に応じて「ちょちょい」と作成して, 後は捨ててしまうソフトウエアを作成する事があると思います。よくあるのが, 「ちょっと試してみよう」的なソフトです。私も過去に何度もそのようなソフトを作ってきました。気が付いてみると …「あれ, これって, 前にも作ったような … 」と思う事がしばしばあります。若い時はそれでも良かったのですが, 最近, 年齢と供にプログラミング速度も低下してきて, 過去に自分で作成したソフトウエアのソースを追っていると「なんて凄いロジックなんだ … 」って思う事があります。

そんなわけで, 使い捨ては勿体無いと思い始めるようになり, それまで使い捨てだったソフトウエアも消さずに残しておくように心がけています。それらのソフトウエアを保存しておく場所を私は,

「がらくた置き場」

と, 呼んでいます。「がらくたシリーズ」は, これらのソフトウエアを少し焼きなおして作成されたものの総称なのです。

6.2 へなちょこシリーズ

「へなちょこシリーズ」は, もうはっきりいって私の主観以外の何者でもありません。私は, Emacs Lisp を始めたのが, 年齢的に若くなかったので, 他の言語 (C 言語とか, X86 Assembler) と比較して, あまり得意ではありません。それでも何とか「こんなことを Emacs でやりたいよ〜」という思いが強く, なんとか自分のやりたい事を実現する為に, 他の諸先輩方の Emacs Lisp を参考にして, それなりに扱えるようになりました。

しかし, lisp というものが良くわかっていない (自分ではわかっていないと思う) 為か,

な〜んか, へなちょこだな〜

と思ってしまうのです。しかも, 実際にへなちょこなのです。しかし,

へなちょこなもんは, へなちょこなんだ!!!。やりたい事が出来ているからいいじゃないか!!

と, 自分に逆切れして開き直っています (^_^;。と, いうわけで「へなちょこシリーズ」が出来たわけです。

… どうでも良い事でした。

7 取り扱い

本プログラムはフリーソフトウェアです。本プログラムを使用して生じたいかなる結果に対しても作者は責任を負わないこととします。個人の責任に於いて使用して下さい。入手したアーカイブのままの形式であれば, 再頒布, 転載は可能とします。常識的に扱ってください。

苦情, 希望, バグ報告, 感想等は歓迎いたします。以下迄お願い致します。

T.Furukawa tfuruka1@nifty.com

索引

Z

Z