JFP リファレンスマニュアル 4: ファイル形式

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 817–1785–11
2006 年 11 月

061212@15821

目次

はじめに

概要

SunOS リファレンスマニュアルは、初めて SunOS を使用するユーザーやすでにある程度の知識を持っているユーザーのどちらでも対応できるように解説されています。このマニュアルを構成するマニュアルページは一般に参照マニュアルとして作られており、チュートリアルな要素は含んでいません。それぞれのコマンドを実行すると、どのような結果が得られるかについて、詳しく説明されています。なお、各マニュアルページの内容はオンラインでも参照することができます。

このマニュアルは、マニュアルページの内容によっていくつかのセクションに分かれています。各セクションについて以下に簡単に説明します。

- セクション 1 は、オペレーティングシステムで使えるコマンドを説明します。
- セクション 1M は、システム保守や管理用として主に使われるコマンドを説明します。
- セクション 2 は、すべてのシステムコールについて説明します。ほとんどのシステムコールに 1 つまたは複数のエラーがあります。エラーの場合、通常ありえない戻り値が返されます。
- セクション 3 は、さまざまなライブラリ中の関数について説明します。ただし、UNIX システムプリミティブを直接呼び出す関数については、セクション 2 で説明しています。
- セクション 4 は、各種ファイルの形式について説明します。また、ファイル形式を宣言する C 構造体を適用できる場合には随時説明しています。
- セクション 5 は、文字セットテーブルなど他のセクションには該当しないものについて説明します。
- セクション 7 は、特殊なハードウェア周辺装置またはデバイスドライバに関するさまざまな特殊ファイルについて説明します。STREAMS ソフトウェアドライバ、モジュール、またはシステムコールの STREAMS 汎用セットについても説明します。
- セクション 9 は、カーネル環境でデバイスドライバを記述するのに必要な参照情報を提供します。ここでは、デバイスドライバインタフェース (DDI) とドライバ/カーネルインタフェース (DKI) という 2 つのデバイスドライバインタフェース仕様について説明します。

- セクション 9F は、デバイスドライバが使用できるカーネル関数について説明します。

以下に、このマニュアルの項目を表記されている順に説明します。ほとんどのマニュアルページが下記の項目からなる共通の書式で書かれていますが、必要でない項目については省略されています。たとえば、記述すべきバグがコマンドにない場合などは、「使用上の留意点」という項目はありません。各マニュアルページの詳細は各セクションの `intro` を、マニュアルページの一般的な情報については `man`(1) を参照してください。

名前　コマンドや関数の名称と概略が示されています。

形式　コマンドや関数の構文が示されています。標準パスにコマンドやファイルが存在しない場合は、フルパス名が示されます。字体は、コマンド、オプションなどの定数にはボールド体 (bold) を、引数、パラメータ、置換文字などの変数にはイタリック体 (Italic) または <日本語訳> を使用しています。オプションと引数の順番は、アルファベット順です。特別な指定が必要な場合を除いて、1 文字の引数、引数のついたオプションの順に書かれています。

以下の文字がそれぞれの項目で使われています。

[ ]　このかっこに囲まれたオプションや引数は省略できます。このかっこが付いていない場合には、引数を必ず指定する必要があります。

. . .　省略符号。前の引数に変数を付けたり、引数を複数指定したりできることを意味します (例: '`filename`...')。

|　区切り文字 (セパレータ)。この文字で分割されている引数のうち 1 つだけを指定できます。

{ }　この大かっこに囲まれた複数のオプションや引数は省略できます。かっこ内を 1 組として扱います。

プロトコル　この項が使われているのは、プロトコルが記述されているファイルを示すサブセクション 3R だけです。パス名は常にボールド体 (bold) で示されています。

機能説明　コマンドの機能とその動作について説明します。実行時の詳細を説明していますが、オプションの説明や使用例はここでは示されていません。対話形式のコマンド、サブコマンド、リクエスト、マクロ、関数などに関しては「使用法」で説明します。

IOCTL　セクション 7 だけに使用される項です。`ioctl`(2) システムコールへのパラメータは `ioctl` と呼ばれ、適切なパラメータを持つデバイスクラスのマニュアルページだけに記載されています。特定のデバイスに関する `ioctl` は、(そのデバイスのマニュアルページに) アルファベット順に記述されています。デバイスの特定のク

ラスに関する ioctl は、mtio(7I) のように io で終わる名前が付いているデバイスクラスのマニュアルページに記載されています。

オプション　各オプションがどのように実行されるかを説明しています。「形式」で示されている順に記述されています。オプションの引数はこの項目で説明され、必要な場合はデフォルト値を示します。

オペランド　コマンドのオペランドを一覧表示し、各オペランドがコマンドの動作にどのように影響を及ぼすかを説明しています。

出力　コマンドによって生成される出力 (標準出力、標準エラー、または出力ファイル) を説明しています。

戻り値　値を返す関数の場合、その値を示し、値が返される時の条件を説明しています。関数が 0 や -1 のような一定の値だけを返す場合は、値と説明の形で示され、その他の場合は各関数の戻り値について簡単に説明しています。void として宣言された関数はこの項では扱いません。

エラー　失敗の場合、ほとんどの関数はその理由を示すエラーコードを errno 変数の中に設定します。この項ではエラーコードをアルファベット順に記述し、各エラーの原因となる条件について説明します。同じエラーの原因となる条件が複数ある場合は、エラーコードの下にそれぞれの条件を別々のパラグラフで説明しています。

使用法　この項では、使用する際の手がかりとなる説明が示されています。特定の決まりや機能、詳しい説明の必要なコマンドなどが示されています。組み込み機能については、以下の小項目で説明しています。

コマンド
修飾子
変数
式
入力文法

使用例　コマンドや関数の使用例または使用方法を説明しています。できるだけ実際に入力するコマンド行とスクリーンに表示される内容を例にしています。例の中には必ず example% のプロンプトが出てきます。スーパーユーザーの場合は example# のプロンプトになります。例では、その説明、変数置換の方法、戻り値が示され、それらのほとんどが「形式」、「機能説明」、「オプション」、「使用法」の項からの実例となっています。

環境　コマンドや関数が影響を与える環境変数を記述し、その影響について簡単に説明しています。

終了ステータス	コマンドが呼び出しプログラムまたはシェルに返す値と、その状態を説明しています。通常、正常終了には 0 が返され、0 以外の値はそれぞれのエラー状態を示します。
ファイル	マニュアルページが参照するファイル、関連ファイル、およびコマンドが作成または必要とするファイルを示し、各ファイルについて簡単に説明しています。
属性	属性タイプとその対応する値を定義することにより、コマンド、ユーティリティ、およびデバイスドライバの特性を一覧しています。詳細は `attributes`(5) を参照してください。
関連項目	関連するマニュアルページ、当社のマニュアル、および一般の出版物が示されています。
診断	エラーの発生状況と診断メッセージが示されています。メッセージはボールド体 (bold) で、変数はイタリック体 (Italic) または <日本語訳> で示されており、C ロケール時の表示形式です。
警告	作業に支障を与えるような現象について説明しています。診断メッセージではありません。
注意事項	それぞれの項に該当しない追加情報が示されています。マニュアルページの内容とは直接関係のない事柄も参照用に扱っています。ここでは重要な情報については説明していません。
使用上の留意点	すでに発見されているバグについて説明しています。可能な場合は対処法も示しています。

参照

序章

名前

Intro_jfp, intro_jfp – JFP ファイル形式の序章

機能説明

本セクションでは、 JFP が提供する種々のファイルの形式について説明します。C 言語の構造体によるファイル形式の宣言も適宜示します。通常、C の構造体宣言を含むヘッダーは、 ディレクトリ `/usr/include` または `/usr/include/sys` にあります。 ただし、C プログラムへインクルードするには、`#include <`*filename.h*`>` または `#include <sys/`*filename.h*`>` という構文を使用する必要があります。

オペレーティングファイル上に複数の種類のファイルシステムが存在できるようになったので、名前が同じマニュアルページのインスタンスが複数存在することがあります。このようなマニュアルページでは、マニュアルページの冒頭に *name_fstype* という書式で、ファイルシステムタイプ名を示しています。

ファイル形式一覧

名前	説明
Intro_jfp(4)	JFP ファイル形式の序章
atok12wordlist(4)	ATOK12 辞書ユーティリティで使用するテキスト形式の単語ファイル
css.conf(4)	CS 起動情報ファイル
jserverrc(4)	Wnn6 かな漢字変換サーバーの初期化ファイル
sdtudc_map(4)	ユーザー定義文字変換マップ
uumkey(4)	Wnn6 かな漢字変換操作キー割り当て定義ファイル
uumrc(4)	xjsi、uum 初期化ファイル
wnnenvrc(4)	Wnn6 かな漢字変換辞書/変換パラメタ設定ファイル
wnnhosts(4)	Wnn6 かな漢字変換サーバー/辞書引きサーバー・アクセス制御ファイル
wnn_2A_CTRL(4)	入力変換モード切り替え定義表
wnn_2B_ROMKANA(4)	ローマ字かな変換定義表
wnn_automaton(4)	オートマトン
wnn_cvt_key_tbl(4)	かな漢字変換フロントエンドプロセッサ (uum) キーコード変換表ファイル
wnn_cvt_xim_tbl(4)	xjsi 用キー変換テーブル
wnn_hinsi.data(4)	Wnn6 品詞管理ファイル
wnn_mode(4)	モード定義表
wnn_serverdefs(4)	Wnn6 かな漢字変換サーバー接続パラメタ設定ファイル
wnn_ximrc(4)	xjsi 用環境設定ファイル

参照

ファイル形式

名前　　atok12wordlist – ATOK12 辞書ユーティリティで使用するテキスト形式の単語ファイル

形式　　**atok12wordlist**

機能説明　　ATOK for Solaris 日本語入力システムでは、ATOK パレットのメニューから [ATOKヘルプ] を起動し「よくある質問」→「(左フレーム)単語・辞書」→「単語ファイルを利用して、単語を一括して登録する」→「単語ファイルを作成する」を参照してください。

名前　　css.conf – CS 起動情報ファイル

形式　　`/etc/css.conf`

機能説明　　`css.conf` ファイルは `cssd`(1M) に対して、CS 起動スクリプトを格納する CS 起動情報ディレクトリを 指定するファイルです。ファイルは一般のテキストファイル形式になっており、通常のテキストエディタで編集可能です。

行の最初に `#` があるとその行を注釈行として解釈し、 読み飛ばします。

各行には CS 起動情報ディレクトリのパス名を記述します。 初期値は次の通りです。

```
/etc/css.d
/usr/lib/css.d
```

ファイル　　`/etc/css.conf`　　省略時 CS 起動情報ファイル

関連項目　　`cssd`(1M)

名前　jserverrc – 日本語マルチクライアントサーバー (jserver) の初期化ファイル

形式　`/usr/lib/wnn/ja_JP/jserverrc`

機能説明　`jserverrc` は、日本語マルチクライアントサーバーを使用する時の環境を設定するもので、`jserverrc` が起動する時に読み込まれます。設定できるものは、以下のとおりです。

1. 起動時に読み込む辞書

   `readfile` *dictionary_filename*
   サーバーが、立ち上がり時に読み込む辞書ファイル名を指定します。ここで、指定された辞書は、サーバーの立ち上がり時に読み込まれ、サーバープロセスが終了するまでサーバーが持ち続けます。これは、各クライアントが、起動する時に、辞書を読み込む時間を節約するために使用されます。

2. 固定化する環境の最大数

   `max_sticky_env` *maximum_number_of_fixed_environments*
   固定化する環境の最大数を指定します。固定化する環境とは、環境を終了させても、起動時の環境が記憶される環境です。これは、次にその環境を使用する時に、環境設定を省くため、立上りが早くなります。デフォルトは、`10` です。

3. サーバーが辞書を管理するディレクトリ

   `jserver_dir` *text_string*
   サーバーが、辞書を管理するディレクトリパスを指定します。ユーザーの頻度ファイルとユーザー辞書が、指定されたディレクトリの下で管理されます。デフォルトは、`/usr/local/lib/dic/` です。`@LIBDIR`、`@LANG` の記法が使用できます。

`@LIBDIR`	デフォルトの環境ファイルのディレクトリパス名 (`/usr/lib/wnn`)。
`@LANG`	`jserver` の作成時に決定される言語名。Solaris の日本語環境では ja_JP です。

4. かな漢字変換のパラメタ値

   `def_param` *number_0. . .number_16*
   かな漢字変換のパラメタ、および、疑似品詞の頻度を指定します。( ) 内はデフォルトの値です。

number_0	N (大) 文節解析の N (5)
number_1	大文節中の小文節の最大数 (10)
number_2	幹語の頻度のパラメタ (2)
number_3	小文節長のパラメタ (45)
number_4	幹語長のパラメタ (0)

number_5	直前に使ったことを示すビットのパラメタ (80)
number_6	辞書のパラメタ (5)
number_7	小文節の評価値のパラメタ (1)
number_8	大文節長のパラメタ (20)
number_9	小文節数のパラメタ (0)
number_10	疑似品詞「数字」の頻度 (400)
number_11	疑似品詞「カナ」の頻度 (–100)
number_12	疑似品詞「英数」の頻度 (400)
number_13	疑似品詞「記号」の頻度 (80)
number_14	疑似品詞「閉括弧」の頻度 (200)
number_15	疑似品詞「付属語」の頻度 (2)
number_16	疑似品詞「開括弧」の頻度 (200)

上記のパラメタには、整数値を指定してください。

下記は自動チューニングできるパラメタです。

- 幹語の頻度のパラメタ
- 小文節長のパラメタ
- 幹語長のパラメタ
- 直前に使ったことを示すビットのパラメタ
- 小文節の評価値のパラメタ
- 大文節長のパラメタ
- 小文節数のパラメタ

5. かな漢字変換のパラメタ上限値

`max_param` *number_0. . .number_16*
かな漢字変換のパラメータの上限値を指定します。各項目の意味、順序は `def_param` の指定と同じです。

自動チューニングできるパラメタの上限値はデフォルト値より小さい時、デフォルトと同じ値がセットされます。自動チューニングできないパラメタの上限値とデフォルト値が違うとき、デフォルトと同じ値がセットされます。上限値が 2 回以上セットされる時、最後にセットされた値が有効になります。

6. かな漢字変換のパラメタ下限値

`min_param` *number_0. . .number_16*
かな漢字変換のパラメタの下限値を指定します。各項目の意味、順序は `def_param` の指定と同じです。

自動チューニングできるパラメタの下限値はデフォルト値より大きい時、デフォルトと同じ値がセットされます。自動チューニングできないパラメタの下限値とデフォルト値が違うとき、デフォルトと同じ値がセットされます。下限値が 2 回以上セットされる時、最後にセットされた下限値が有効になります。

7. 疑似品詞の「英数」の文字コードの指定

`set_giji_eisuu` *character_code. . .*
指定した文字コードが、英数字に加えて疑似品詞「英数」として、疑似文節の変換に使用できます。

文字コードの指定では、16 進数、10 進数、8 進数で表記できます。

文字	表記法
CTRL_A	^A
' '(SPACE)	0x20 \x20 32 040 \o40
' '	' '
'_'	'_'

8. 辞書 OnDisk 機能

`ondisk_off`
辞書 OnDisk 機能を使用しない場合、記述します。この項目を記述しない場合は、辞書 OnDisk 機能を使用します。

辞書 OnDisk 機能を使用すると、メモリ使用量を削減することができます。

関連項目

`jserver`(1M)

名前　uumkey – かな漢字変換操作キーバインド定義ファイル

形式　/usr/lib/wnn/ja_JP/uumkey

機能説明　uumkey は、日本語入力時のキーの設定ファイルです。各ユーザーが uumkey を設定することで、ユーザー独自のキー操作が可能です。セミコロン (;) またはコロン (:) で始まる行は、コメントになります。

機能外エントリ　unset *function_entry_name*　指定エントリに対する既設定を取り消します。

機能エントリ設定形式　機能名　キーコード [キーコード ....]

キーコードの表記の仕方は、8 進数、10 進数、16 進数が使用できプログラミング言語 C における表記に準じます。また、コントロールキーについては、次のような略記法が使用できます。

文字	表記法
CTRL_A	^A

1 つの機能に対してコードを最大 10 個まで書くことができます。同じ機能を 2 度セットした場合には後のものが優先されます。1 つの機能を複数のキーにバインドする時は、1 度に設定してください。ただし、同じモードで作動する複数の機能を同じキーにバインドすることはできません。

機能エントリ　各機能エントリには、その機能が動作可能であるモードが決められています。モードには、モード番号に対応して、次のようなものがあります。

操作モード	内容
0	変換後の変換結果を修正しているモード
1	変換前の文字を入力しているモード
2	変換後に文節の長さを伸縮するモード
3	バッファーに何も文字が入っていないモード
4	カーソルを動かしながら、何か候補を選択する時のモード (すなわち、次候補選択や、ユーティリティーモード、登録時の品詞や辞書を選択する時のモード)
5	楽々入力のモード

エントリ名の最後に _e が付いたエントリは、 _e が付いていない同じ名前のエントリと同じ機能がバッファーが空の状態 (モード 3) においても作動します。rubout の働きをするキーの設定は、 キーバインドしなければ、uum を起動した tty の erase 文字の設定に従います。

以下に設定できるエントリ名とその機能を説明します。

表1基本機能エントリ

機能エントリ	操作モード	機能
henkan_on	01234	変換モードの on/off を変更する。変換モード off では、ローマ字かな変換オートマトンも同時に切り離される。henkan_on のエントリに設定するコードには、cvt_xim_tbl やオートマトンによるキーコード変換の出力コードを指定できない
quote_keyin	01234	次の入力文字をクオートする。henkan_on でバインドされたコードを入力するための非常手段として用意されている
send_string	012	変換行内の文字列に、この機能がバインドされているコードを付け加えた文字列を、アプリケーションに送る
kakutei	0125	変換行内の文字列をアプリケーションに送る
one_char_kakutei	012	先頭の 1 文字を確定する
one_char_no_henkan	01	先頭の 1 文字を変換しない
forward_char	1	1 文字右へカーソルを移動する
backward_char	1	1 文字左へカーソルを移動する。ただし、一度変換して途中から解除してある場合は、カーソルが解除してある部分の一番左にある状態でこの機能が呼ばれると、連文節変換をしてから 1 つ左の文節へ移る
goto_top_of_line	1	行の先頭の文字へカーソルを移動する
goto_end_of_line	1	行の後端の文字へカーソルを移動する
delete_char_at_cursor	15	カーソルの位置の文字消去。楽々入力モードの場合は、楽々入力の候補一覧からカーソル位置の候補を消去する

表 1 基本機能エントリ　(続き)

機能エントリ	操作モード	機能
kaijo	02	カーソルの位置の文節以降の変換された文字列を、変換される前の状態に戻す
henkan	1	連文節変換を行う
tan_henkan	1	単文節変換を行う
nobi_henkan	2	文節の伸縮時に、反転している部分を単文節変換して、それ以降を連文節変換する
jikouho	0	次候補を表示する
zenkouho	0	前候補を表示する
select_jikouho	0	次候補一覧を表示する
kana_henkan	1	漢字かな変換を行なう。ただし、逆引き形式辞書 (登録可能形式で、漢字→かな変換が行える) の使用時のみ有効
kill	1	カーソル以降 (カーソル位置を含む) の文字列を消去して、キルバッファーに蓄積する
yank	1	キルバッファーの内容を、カーソルの位置に挿入する
yank_e	13	キルバッファーの内容を、カーソルの位置に挿入する
bunsetu_nobasi	0	文節の長さを 1 文字長くする
bunsetu_chijime	0	文節の長さを 1 文字短くする
sainyuuryoku	1	前に入力されたかな文字列に置き換える
sainyuuryoku_e	13	前に入力されたかな文字列に置き換える
kuten	1	区点コード入力画面を表示する
kuten_e	13	区点コード入力画面を表示する
jis	1	JIS コード入力画面を表示する
jis_e	13	JIS コード入力画面を表示する
sjis	1	SJIS コード入力画面を表示する

表 1 基本機能エントリ　(続き)

機能エントリ	操作モード	機能
sjis_e	13	SJIS コード入力画面を表示する
kuten2	13	補助区点コード入力画面を表示する
jis2	13	補助 JIS コード入力画面を表示する
unicode_input	13	Unicode コード入力画面を表示する
next_ku_kuten	4	区点一覧表で次の区点を表示する
previous_ku_kuten	4	区点一覧表で前の区点を表示する
next_page	4	候補一覧 (区点一覧含む) 表示画面で次ページを表示する
previous_page	4	候補一覧 (区点一覧含む) 表示画面で前ページを表示する
redraw_line	0124	変換行の書き直しを行う
redraw_line_e	01234	変換行の書き直しを行う
previous_history	1	ヒストリに記憶されている 1 つ前の文字列に置き換える
previous_history_e	13	ヒストリに記憶されている 1 つ前の文字列に置き換える
next_history	1	ヒストリに記憶されている 1 つ後の文字列に置き換える
next_history_e	13	ヒストリに記憶されている 1 つ後の文字列に置き換える
all_history	1	ヒストリに蓄えられているすべての文字列を次候補として一覧を出す
all_history_e	13	ヒストリに蓄えられているすべての文字列を次候補として一覧を出す
quit	45	候補一覧などのウインドウを閉じる
change_to_insert_mode	0	変換された文字列を、もう一度編集できる状態にする。ここで、変換された漢字は、かなには戻せない
quote	1	次の入力文字が、henkan_on 以外の文字なら、それを quote する。即ち、次の入力文字を、ローマ字かな変換に通さずに、キーバインドされている機能があってもそれを無効として、直接変換行に入れる

表 1 基本機能エントリ　(続き)

機能エントリ	操作モード	機能
quote_e	13	次の入力文字が、henkan_on 以外の文字なら、それを quote する。即ち、次の入力文字を、ローマ字かな変換に通さずに、キーバインドされている機能があってもそれを無効として、直接変換行に入れる
forward_select	4	次候補など候補を選択するモードで、反転部分を右に移動させる
backward_select	4	次候補など候補を選択するモードで、反転部分を左に移動させる
next_select	4	次候補など候補を選択するモードで、次の画面に移動する
previous_select	4	次候補など候補を選択するモードで、前の画面に移動する
linestart_select	4	次候補など候補を選択するモードで、画面の先頭に移動する
lineend_select	4	次候補など候補を選択するモードで、画面の最後に移動する
select_select	4	次候補など候補を選択するモードで、候補選択と同時にそのモードから抜ける
send_ascii_char	01234	それ以降バッファーが空の時にアスキー文字が入力された時に、それをバッファーに蓄積しないようにする
not_send_ascii_char	01234	それ以降バッファーが空の時にアスキー文字が入力された時に、それをバッファーに蓄積するようにする
toggle_send_ascii_char	01234	それ以降バッファーが空の時にアスキー文字が入力された時に、動作をその時の動作と反対の状態にする
quote_send_ascii_char	3	send_ascii_char モードの時で、バッファーが空の時に次のアスキー文字をバッファーに蓄積する。それ以降のバッファーが空の時にはバッファーに蓄積しないようにする
reconnect_jserver	01234	jserver との再接続を行う

表2キーバインド細分化

機能エントリ	操作モード	機能
forward_bunsetsu	0	1 文節右へ移動する。未定義の場合は forward_char と同じキーコードにする
henkan_forward	2	反転している部分を単文節変換にして、それ以降を連文節変換して、1 つ右の文節に移動する。未定義の場合は forward_char と同じキーコードにする
backward_bunsetsu	0	1 文節左へ移動する。未定義の場合は backward_char と同じキーコードにする
henkan_backward	2	反転している部分を単文節変換にして、それ以降を連文節変換して、1 つ左の文節に移動する。未定義の場合は backward_char と同じキーコードにする
top_bunsetsu	0	行の先頭の文節へ移動する。未定義の場合は goto_top_of_line と同じキーコードにする
end_bunsetsu	0	行の後端の文節へ移動する。未定義の場合は goto_end_of_line と同じキーコードにする
jmptijime	2	行の先頭の文字へ移動する。未定義の場合は goto_top_of_line と同じキーコードにする
c_end_nobi	2	行の後端の文字へ移動する。未定義の場合は goto_end_of_line と同じキーコードにする
forward	2	文節の長さを 1 文字分長くする。未定義の場合は bunsetu_nobasi と同じキーコードにする
tijime	2	文節の長さを 1 文字分短くする。未定義の場合は bunsetu_chijime と同じキーコードにする
rubout	1	rubout キー。カーソルの左位置の文字消去。rubout は romkan にも使用されるので、キーコードの指定は 1-byte の範囲で動的変更できないに制限される

表 2 キーバインド細分化　　(続き)

機能エントリ	操作モード	機能
touroku	012	単語登録画面を表示する
touroku_e	0123	単語登録画面を表示する
bushu_e	0	部首入力画面を表示する
tankan_henkan	01	単漢字変換を行う
select_ikeiji_dai	01	異形字変換を行う
zip_henkan	012	郵便番号変換を行う
tel_henkan	012	電話番号変換を行う
select_assoc_dai	0	連想変換を行う。類義語、同義語を表示する
hankaku	012	入力文字を半角に変換する
eisuu	012	入力文字を英数字に変換する
hiragana	012	入力文字をひらがなに変換する
katakana	012	入力文字をカタカナに変換する
eg_Aa_henkan_big_loop	012	ひらがな → カタカナ → 半角カタカナ → 全角英数字 → 半角英数字の変換を繰り返す
eg_Aa_henkan_small_loop	012	ひらがな → カタカナ → 半角カタカナの変換を繰り返す
reverse_conv	13	逆引き変換画面を表示する
codelist	13	コード一覧画面を表示する
unicode_codelist	13	Unicode コード一覧画面を表示する
greek	13	ギリシャ語一覧を表示する
russian	13	ロシア語一覧を表示する
kigou	13	記号一覧を表示する
yosoku_modein	15	楽々入力候補を選択する

使用例

```
; Commands              Codes
forward_char            ^F 0x90
jikouho                 ^N 0x92 ^W 0x9E
;yank                   ^Y
yank_e                  ^Y
select_select           32 ^J ^M 0x9E 0x9F
```

使用上の注意事項

コードとして取ることのできる数は、0 以上 512 未満の整数で、これには、キーボードで発生できるものと、そうでないものとがありますが、発生できるものはそのままコードを書くことができます。発生できないもの (128 以上の整数等) を使用する場合は、キーコード変換 (`cvt_xim_tbl`)、あるいはオートマトンでそのコードを発生させる必要があります。

変換モード OFF とは一時的に Wnn8LE を切り離したのと同じ効果を上げるもので、ローマ字かな変換オートマトンも同時に切り離されます。`henkan_on` エントリに設定するコードには、キーコード変換の出力コードを指定できません。

関連項目

`uumrc(4)`, `wnn_automaton(4)`

名前　uumrc – Wnn8LE 初期化ファイル

形式　`/usr/lib/wnn/ja_JP/uumrc`

機能説明　uumrc は、かな漢字変換の標準インタフェースを使用する時の環境を設定するもので、各ユーザーごとの設定が可能です。

エントリが重複した場合は上書きされます。

セミコロン (;) で始まる行はコメントになります。

このファイルで設定できるものは、以下のとおりです。

`include` *uumrc_file_name*
他の uumrc ファイルを読み込みます。デフォルトの uumrc ファイルに個人の設定を 付け加える場合などに使用します。`@DEFAULT_RC_NAME` と指定すれば、以下の順番でデフォルトの uumrc ファイルを検索し、設定の追加を行うことができます。

1. `@HOME/.Wnn8/uumrc`
2. `/usr/lib/wnn/ja_JP/uumrc`

`setconvenv` *wnnenvrc_file_name*

`setconvenv` *wnnenvrc_file_name* `sticky`

`setconvenv` *server_host_name wnnenvrc_file_name*

`setconvenv` *server_host_name wnnenvrc_file_name* `sticky`
かな漢字変換用の環境設定ファイルを指定します。省略時は、以下の順番でデフォルトの wnnenvrc ファイルを検索し、設定を行います。

1. `@HOME/.Wnn8/wnnenvrc`
2. `/etc/wnn/ja_JP/wnnenvrc`
3. `/usr/lib/wnn/ja_JP/wnnenvrc`

サーバーのホスト名が指定されているものは、そのサーバーに接続されます。sticky を指定すると現在の環境は、Wnn8LE の終了後にも記憶され、次回の Wnn8LE 起動で環境の初期化を行なう必要がなく、立上りが早くなります。

`setkankanaenv` *wnnenvrc_file_name*

`setkankanaenv` *wnnenvrc_file_name* `sticky`

`setkankanaenv` *server_host_name wnnenvrc_file_name*

`setkankanaenv` *server_host_name wnnenvrc_file_name* `sticky`
漢字かな変換用の環境設定ファイルを指定します。省略時は、漢字かな変換ができません。また、 Wnn8 のシステム辞書を用いて漢字かな変換を行うことはできません。サーバーのホスト名を指定すると、そのサーバーに接続されます。

sticky を指定すると現在の環境は、Wnn8LE の終了後にも記憶され、次回の Wnn8LE 起動で環境の初期化を行なう必要がなく、立上りが早くなります。

setuumkey *uumkey_file_name*
キーバインド定義ファイルを指定します。省略時は、/usr/lib/wnn/ja_JP/uumkey です。

setrkfile roman_character-Kana_conversion_file_name
ローマ字かな変換ファイルを指定します。ディレクトリ名の場合、その下の mode ファイルが対象になります。省略時は、/usr/lib/wnn/ja_JP/rk/mode です。

not_send_ascii_char
かな漢字変換バッファー (変換行) が空の時、ASCII 文字をかな漢字変換バッファーに取り込みます (デフォルト)。

send_ascii_char
かな漢字変換バッファー (変換行) が空のとき、ASCII 文字をかな漢字変換バッファーに取り込みません。

waking_up_in_henkan_mode
変換モード ON で立ち上げます (デフォルト)。

waking_up_no_henkan_mode
変換モード OFF で立ち上げます。

setjishopath *path_name*
辞書追加のときの辞書名入力バッファーの初期値を指定します。デフォルトは空文字列です。

sethindopath *path_name*
辞書追加のときの頻度ファイルの入力バッファーのパス名の初期値を指定します。デフォルトは空文字列です。

setfuzokugopath *path_name*
辞書追加のときの付属語ファイルのディレクトリパス名の初期値を指定します。デフォルトは空文字列です。

setmaxchg *number*
最大変換可能文字数を指定します。数字に 0 以下の値 (0 を含む) を指定すると、デフォルト値となります。デフォルトは、100 です。

setmaxbunsetsu *number*
最大変換可能文節数を指定します。上限は、400 です。数字に 0 以下の値 (0 を含む) を指定すると、デフォルト値となります。デフォルトは、80 です。

setmaxichirankosu *number*
次候補一覧の時の最大表示次候補数を指定します。省略時は、画面の幅に応じて表示個数を選択します。数字に 0 以下の値 (0 を含む) を指定すると、デフォルト値となります。デフォルトは、36 です。

setmaxhistory *number*
ヒストリーを最大何個まで記憶するかを指定します。数字に 0 以下の値 (0 を含む) を指定すると、デフォルト値となります。デフォルトは、11 です。

excellent_delete
オートマトン (ローマ字かな変換) で文字を消去する時、確定された文字を入力時の個々のアルファベットの単位で消去します (デフォルト)。

simple_delete
オートマトン (ローマ字かな変換) で文字を消去する時、確定された文字を個々の日本語文字の単位で消去します。

flow_control_on
tty のフローコントロールを on に設定します (デフォルト)。

flow_control_off
tty のフローコントロールを off に設定します。

convkey_not_always_on
変換 off の時、キーコード変換を機能させません (デフォルト)。

convkey_always_on
変換 off の時、キーコード変換を機能させます。

remove_cs
/etc/termcap から cs を削除します。

not_remove_cs
/etc/termcap から cs を削除しません。

touroku_comment
単語登録時にコメントの入力を行います。

touroku_no_comment
単語登録時にコメントの入力を行いません。

henkan_off_kuten
句点 (。) 変換を行いません (デフォルト)。

henkan_on_kuten
句点 (。) 変換を行います。

autotune_off
パラメタ自動チューニングを行いません (デフォルト)。

autotune_on
パラメタ自動チューニングを行います。

include、setuumkey、setrkfile、setconvkey、setconvenv、setjishopath、sethindopath、setfuzokugopath、setkankanaenv、setkanaromenv の引数の先頭に ~、~username、@HOME、@LIBDIR、@LANG の記法が使用できます。

~	環境変数 HOME の値
~username	/etc/passwd に登録されているログネーム username のホームディレクトリ
@HOME	環境変数 HOME の値
@LIBDIR	標準の Wnn8LE 環境ファイルのディレクトリパス名 (/usr/lib/wnn)
@LANG	環境変数 LANG の値の最初の 5 文字

setdic、setjishopath、sethindopath、setdic、setjishopath、sethindopath の引数内では、最初の @USR を起動ユーザー名に展開します。

関連項目

jserver(1M), uumkey(4), wnn_cvt_key_tbl(4), wnn_automaton(4)

名前　wnn_2A_CTRL – コントロールコマンド定義表

形式　/usr/lib/wnn/ja_JP/rk/2A_CTRL

機能説明　2A_CTRL は WnnLE 起動時に解釈され、変換用のキーの設定を行います。文字入力モードの設定のために使用します。これで変換されたコードは、uumkey の設定の中で使用されます。

設定例　'\x81' (switch katakana) ;PF1 key

'\x82' (switch zenkaku) ;PF2 key

'\x83' (switch romkan) ;PF3 key

注意事項　PF1 から PF4 のキーおよびカーソルキーについては、次のような注意が必要です。つまり、端末が 2 バイト以上のコードを発生する場合、この表による変換が行われる前に、WnnLE によってキーボードから発生する文字列のデータが 1 バイトのコードに変換されます (cvt_xim_tbl(4) のマニュアルページを参照してください)。

以下に、PF1 から PF4 のキーとカーソルキーが入力された場合に、この表で受け取るコードを示します。ただし、これは、WnnLE によってこの表が使用されている場合の対応で、ローマ字変換ライブラリを使用した場合は、これらのコードは、当てはまりません。

PF1	0x81	→	0x90
PF2	0x82	←	0x91
PF3	0x83	↓	0x94
PF4	0x84	↑	0x93

関連項目　wnn_mode(4), uumkey(4), wnn_cvt_xim_tbl(4), wnn_automaton(4)

名前　wnn_2B_ROMKANA – ローマ字かな変換定義表

形式　/usr/lib/wnn/ja_JP/rk/2B_ROMKANA

機能説明　ローマ字変換用の設定を行います。

設定例

M(q1) ん (q1) (defvar q1 (list B M P))	M の次に B M P のいずれかがきた場合、M を「ん」に確定することを定義しています
(aa)(aa) っ (aa) (defvar aa (list K S T H Y R W G Z D B P C F J V))	「K S T H Y R W G Z D B P C F J V」の何れかが連続して入力された場合、1 文字目の入力を「っ」に確定することを定義しています
TCH っ CH	TCH と入力された場合、T を「っ」に確定することを定義しています
A あ I い U う E え O お . . . (中略) - ー , 、 . 。	A と入力された場合、「あ」に確定することを定義しています
(alpha) (error) (defvar alpha (between A Z))	A から Z までの文字が単独で現れた場合、エラーとします

ローマ字変換デフォルト設定 (括弧内のものは、カタカナモードの時に限り適用される変換を示します。)

A	あ	KA	か	SA	さ	TA	た
I	い	KI	き	SI	し	TI	ち
U	う	KU	く	SU	す	TU	つ
E	え	KE	け	SE	せ	TE	て

O	お	KO	こ	SO	そ	TO	と
NA	な	HA	は	MA	ま	YA	や
NI	に	HI	ひ	MI	み	YI	い
NU	ぬ	HU	ふ	MU	む	YU	ゆ
NE	ね	HE	へ	ME	め	YE	いぇ
NO	の	HO	ほ	MO	も	YO	よ
RA	ら	WA	わ	GA	が	ZA	ざ
RI	り	WI	ゐ	GI	ぎ	ZI	じ
RU	る	WU	う	GU	ぐ	ZU	ず
RE	れ	WE	ゑ	GE	げ	ZE	ぜ
RO	ろ	WO	を	GO	ご	ZO	ぞ
DA	だ	BA	ば	PA	ぱ	GYA	ぎゃ
DI	ぢ	BI	び	PI	ぴ		
DU	づ	BU	ぶ	PU	ぷ	GYU	ぎゅ
DE	で	BE	べ	PE	ぺ	GYE	ぎぇ
DO	ど	BO	ぼ	PO	ぽ	GYO	ぎょ
ZYA	じゃ	JA	じゃ	DYA	ぢゃ	BYA	びゃ
		JI	じ	DYI	でぃ		
ZYU	じゅ	JU	じゅ	DYU	ぢゅ	BYU	びゅ
ZYE	じぇ	JE	じぇ	DYE	ぢぇ	BYE	びぇ
ZYO	じょ	JO	じょ	DYO	ぢょ	BYO	びょ
PYA	ぴゃ	VA	ぶぁ (ヴぁ)	KYA	きゃ		
		VI	ぶぃ(ヴぃ)				
PYU	ぴゅ	VU	ぶ(ヴ)	KYU	きゅ		

PYE	ぴぇ	VE	ぶぇ (ヴぇ)	KYE	きぇ		
PYO	ぴょ	VO	ぶぉ (ヴぉ)	KYO	きょ		
SYA	しゃ	SHA	しゃ	TYA	ちゃ		
		SHI	し	TYI	てぃ		
SYU	しゅ	SHU	しゅ	TYU	ちゅ		
SYE	しぇ	SHE	しぇ	TYE	ちぇ		
SYO	しょ	SHO	しょ	TYO	ちょ		
CHA	ちゃ	NYA	にゃ	HYA	ひゃ	FA	ふぁ
CHI	ち					FI	ふぃ
CHU	ちゅ	NYU	にゅ	HYU	ひゅ	FU	ふ
CHE	ちぇ	NYE	にぇ	HYE	ひぇ	FE	ふぇ
CHO	ちょ	NYO	にょ	HYO	ひょ	FO	ふぉ
MYA	みゃ	LA	ら	RYA	りゃ	LYA	りゃ
		LI	り				
MYU	みゅ	LU	る	RYU	りゅ	LYU	りゅ
MYE	みぇ	LE	れ	RYE	りぇ	LYE	りぇ
MYO	みょ	LO	ろ	RYO	りょ	LYO	りょ
KWA	くゎ	GWA	ぐゎ	TSA	つぁ	QA	くぁ
KWI	くぃ	GWI	ぐぃ	TSI	つぃ	QI	くぃ
KWU	く	GWU	ぐ	TSU	つ	QU	く
KWE	くぇ	GWE	ぐぇ	TSE	つぇ	QE	くぇ
KWO	くぉ	GWO	ぐぉ	TSO	つぉ	QO	くぉ
N	ん						
N'	ん						

XA	ぁ	\A	ぁ	XYA	ゃ	\YA	ゃ
XI	ぃ	\I	ぃ				
XU	ぅ	\U	ぅ	XYU	ゅ	\YU	ゅ
XE	ぇ	\E	ぇ				
XO	ぉ	\O	ぉ	XYO	ょ	\YO	ょ
XTU	っ	\TU	っ	XTI	てぃ	XWI	うぃ
XTSU	っ	\TSU	っ	XDI	でぃ	XWE	うぇ
XWA	ゎ	\WA	ゎ	XDU	どぅ	XWO	うぉ
XKA	(ヵ)	\KA	(ヵ)	XDE	でぇ		
XKE	(ヶ)	\KE	(ヶ)	XDO	どぉ		
-	ー	/	・	Z.	…		
,	、	[	「	Z-	〜		
.	。	]	」				

母音の次に長音がきた場合、次のように確定します。

A^	ああ	A~	ああ
I^	いい	I~	いい
U^	うう	U~	うう
E^	えい	E~	えい
O^	おう	O~	おう

Ｍの次にＢＭＰのいずれかがきた場合、Ｍを"ん"に確定します。

ＫＳＴＨＹＲＷＧＺＤＢＰＣＦＪＶの内いずれかが、連続して入力された場合、初めの文字を、"っ"に確定します。

関連項目　wnn_mode(4)

名前　wnn_automaton – オートマトン

機能説明　オートマトンは、IIIMF ランゲージエンジン Wnn8LE など (Wnn8LE) のローマ字かな変換を実現している機能で、変換内容を設定した表 (変換表) を変えることで任意の変換にすることができます。また、このシステムには、これと同一の機能を持ったオートマトンライブラリ (日本語入力ライブラリ (libwnn) の romkan_ で始まる関数) があり、多種多様な変換プログラムに応用できます。

オートマトンは、変換表に従って直列に結ばれた 3 つの変換 (順に、前処理、本処理、後処理) を行い、その最終結果を出力します。3 つの処理操作は、それぞれの変換表に基づいて変換を行います。また、オートマトンは、モード機能を持っています。モードを切り替えることにより、3 つの処理で使用する表の組み合わせを動的に変更できます。このモードの設定、切り替えコードの設定も変換表で行います。

変換表が、テキストファイルであるため、表は容易に変更でき、しかも任意の変換表に変更できます。また、1 つの変換を行った後、次の変換を行うまでは、 BS (バックスペース) により変換の 1 つ前の状態に戻すことができます。

Wnn8LE のローマ字かな変換では、本処理で「英大文字からひらがな」の変換しか行えませんが、前処理で「英大文字から英大文字」、「英小文字から英大文字」の切り替え、また後処理では「ひらがなからひらがな」、「ひらがなからカタカナ」、「ひらがなから半角カタカナ」の切り替えを行うことにより、ローマ字かな変換の入力と出力のさまざまな状況に対処しています。

オートマトンは次のように処理されます。

1. 入力。英 (半角) の大文字と小文字
2. 前処理。小文字は大文字に変換
3. 本処理。英大文字からひらがなへの変換表に基づいて変換
4. 後処理。ひらがなを必要に応じてカタカナや半角カナに変換
5. 出力。

変換表　オートマトンは、変換表として次の表を使用します。

- モード定義表

  モード宣言や使用する対応表を指定する表。ファイル名は mode
- 対応表
  - 前処理表

    前処理で使用する対応表。1 で始まるファイル名
  - 本処理表

    本処理で使用する対応表。2 で始まるファイル名
  - 後処理表

    後処理で使用する対応表。3 で始まるファイル名

モード定義表には、モードの宣言、各モードで使用する対応表の組み合わせとその決定規則を記述します。

対応表には、入力コードと出力コードとの対応が記述されます。対応表は、前処理、本処理、後処理の 3 つに分かれ、それぞれ、任意の数の表を使用できます。

Wnn8LE は、モード定義表を以下の順番で探します。

1. Wnn8LE 初期化ファイル `uumrc` の `setrkfile` エントリによる指定
2. ファイル名 `/usr/lib/wnn/ja_JP/rk/mode`

以下のモード定義表と対応表の説明では、次の表記法を使用しています。

- . . . は、0 回以上の繰り返し
- . . . . . は、1 回以上の繰り返し
- [ ] は、省略可能

モード定義表

モード定義表には、モード宣言、各モードで使用する対応表の組み合わせとその決定規則、モード表示文字列モード表示文字列が記述さます。

モード定義表は、次の 1、2、3、4 で構成されます。ただし、行の先頭または空白文字 (タブを含む) に続き、しかもエスケープされていないセミコロン (;) から行末までの文は、注釈文として扱われます。

1. 特殊文字

特殊文字としては、以下に示すものがあります。

`@HOME` 環境変数 `HOME` を表す

`@MODEDIR` モード定義表の存在するディレクトリを表す

`@LIBDIR` 標準設定がしてある変換表が存在するディレクトリ (`/usr/lib/wnn/`)

~user *user* がユーザー名ならば、そのユーザーのログインディレクトリ名を表す。*user* の指定がなければ、自分のログインディレクトリ名を表す。

2. モード宣言

モード宣言の書式は、次のとおりです。

`defmode` *mode_name* `[initial_status]`
mode_name は、英数字からなる文字です。`[initial_status]` は、`on` または `off` を指定します。省略時は、`off` です。

モード宣言は、そのモードを使用する前に行います。

3. 対応表の検索指定

対応表の検索指定の書式は、次のとおりです。

`search` *directory* . . . . . .
モード定義表で指定された対応表がモード定義表と同じディレクトリにない場合、探しにいく *directory* を指定しま

す。*directory* は、空白で区切って複数指定できます。search は、対応表の指定より前に指定してください。

path *directory*. . . . . .
対応表を検索するために既に格納されているディレクトリ名を削除し、引数に指定された *directory* を格納します。*directory* は、空白で区切って複数指定できます。search は、対応表の指定より前に指定してください。

4. 対応表とモード表示文字列の指定

指定方法は、次の 3 つがあります。

(1) 対応表のファイル名またはモード表示文字列. . . . . .

(2) (if 条件式 対応表の指定またはモード表示文字列. . .)

(3) (when 条件式 対応表の指定またはモード表示文字列 . . .)

対応表は、(1)、(2)、(3) のいずれかで始まるファイル名を指定します。パスによる指定もできます。モード表示文字列は、そのときのモードを表す二重引用符 (“) で囲んだ文字列です。

(a) “*string*“

変換が ON のときのモード表示文字列を表します。

(b) (on_dispmode “*string*“)

変換が ON のときのモード表示文字列を表します。

(c) (off_dispmode “*string*“)

変換が OFF のときのモード表示文字列を表します。

(d) (on_unchg)

変換が ON のときのモード表示文字列をモード変換する前と同じモード表示文字列を表します。

(e) (off_unchg)

変換が OFF のときのモード表示文字列をモード変換する前と同じモード表示文字列を表します。

Wnn8LE では、この文字列がモード表示に使用されます。

オートマトンライブラリは romkan_dispmode() によって、この文字列を取り出すことができます。ただし、モード定義表にあるモード時のモード表示文字列が複数あるように記述されている場合、最後のものだけが有効です。

(2)、(3) は、条件によって選択する対応表を変えたいときに使用します。(2) の if 文は、その条件式の結果が真ならば、その if 文内の指定を参照し、if 文の次の指定は参照しません。条件式が偽ならば、すぐ if 文を抜け出して if 文の次の指定を参照します。

(3) の when 文は、その条件式の結果が真ならば、その文内の指定を参照し、偽ならば参照しません。しかし、if 文と異なり、条件式の真偽にかかわらず when 文の次の指定を参照します。

なお、(2)、(3) で対応表を指定する場合には、(2) または (3) を再帰的に定義できます。

条件式には次のうち 1 つを記述します。

モード名	モードの状態が on のとき真
(and 条件式 条件式)	2 つの条件式が真のとき真
(or 条件式 条件式)	どちらか条件式が真のとき真
(not 条件式)	条件式が偽のとき真
(false)	常に偽
(true)	常に真

たとえば、モード定義に (defmode kana) と (defmode romajikana) があるとき、(and kana romajikana) は、双方のモードが on のとき真になります。

またたとえば、(ここでは、条件式を○◎●で表し、変換表の名前を ABC... で表すという規則のもとに)

(when ○ A (if ◎ B) C) (if ● D) E

と書かれていたとします。そして、条件式○◎●共に成り立っているとします。この並びを最初から見ていきます。まず、(when ○ A (if ◎ B) C) とあります。ここでは○は成り立つので、「A (if ◎ B) C」という並びを見ます。はじめに、表 A を選択します。

次に、(if◎ B) がきて、しかも◎が成り立つので、表 B を選択します。この文は if 文で、しかも条件式が成り立っているので、現在注目している並び「A (if◎ B) C」のうち残りの部分は見ません。これで、「A(if◎ B) C」という並びを見終えたことになりますが、この並びを含んでいたものは、when 文なので、さらに「(when○ A (if◎ B) C) (if● D) E」という並びの、残りの部分を見ます。

次に書かれているのは、(if● D) です。●が成り立つので、表 D を選択しますが、if 文なので、「(when○ A (if◎ B) C) (if● D) E」という並びのうち、残りの部分は見ません。こうして、表 A、B、D が選択されます。

次に、Wnn8LE が使用するモード定義表を例として示します。

このモード定義表では、3 のモードが定義されています。その後の 2A_CTRL から最後までが使用する対応表とモード表示文字列の指定です。モードが変わるごとに、この表を見て、上記のようにして使用する表の選択を行います。

```
(defmode romkan)
(defmode katakana)
(defmode zenkaku)
2A_CTRL
(if romkan
         1B_TOUPPER
         2B_ROMKANA 2B_JIS
          (if (not katakana) "[あr]")
          (if zenkaku 3B_KATAKANA "[アr]")
          3B_HANKATA "[アイr]")    ;「ア」と「イ」は半角のカタカナを示す
2B_DAKUTEN
(if (not katakana)
         1B_ZENHIRA
         (if zenkaku 3B_ZENKAKU "[あ ]")
         "[Aあ]")
(if zenkaku
         1B_ZENKATA
         3B_ZENKAKU
         "[ア ]")
         "[アイA]"    ;「ア」と「イ」は半角のカタカナを示す
```

初期状態は、romkan、katakana、zenkaku のモードすべてが off です。このとき表は、始めに 2A_CTRL を選択します。romkan が off なので、次の if 文は参照ません。そして、2B_DAKUTEN を選択します。次の if 文の条件式 (not katakana) は、katakana が off なので真となります。そこで if 文内を参照し、1B_ZENHIRA を選択します。次に if 文内の if 文を参照します。この if 文では zenkaku が off になっているため条件式が偽となります。したがって、この if 文は参照しません。

次に、モード表示文字列「[Aあ]」を選択します。残りの変換表並びは見ません。

対応表

対応表には、前処理、本処理、後処理のそれぞれが行う変換データ (入力コードと出力コードとの対応) が記述されます。

前処理、後処理は、本処理の補助という位置付けがなされています。このことから、前処理、後処理の対応表には次の制限が課せられます。

前処理表　　下記の (2) の記述ができません。また、(1) の入力コードと出力コードには、それぞれ評価すると文字になる式が 1 つだけ書けます。バッファー残りには書けません。

後処理表　　下記の (2) の記述ができません。また、(1) の入力コードには、評価すると文字になる式が 1 つだけ書けます。バッファー残りには書けません。

対応表の行のうちのある行は、次の (1) から (3) のうち 1 つか、または空行です。この繰り返しで対応表が構成されます。

(1)　入力コード [出力コード [バッファー残り]]

(2)　入力コード 機能

(3)　変数宣言

これらは、2 行にわたって記述することはできません。改行または空白文字 (タブを含む) に続き、かつエスケープされていないセミコロン (;) から行末までの文は、注釈文として扱われます。

出力コードの省略、およびバッファー残りの省略は NULL 文字列の文として扱われます。入力コード、出力コード、およびバッファー残りに記述できるのは、「評価すると文字になる式」と「評価すると文字列になる式」を空白なしに並べたものです。

ここで、評価すると文字式または文字列になる式とは、その式によって文字あるいは文字列に置き換わる、そのような式のことです。

「評価すると文字になる式」には、次のものがあります。

(1) 文字表記　　文字表記を下記に示します (「評価すると文字列になる式」の文字表記とは異なります)。

文字　　「(」、「)」、「'」、「"」、「\」、「;」、「 」(空白文字) を除く文字。

'文字'　　「'」、「\」、「^」を除く文字。

'^文字'　　コントロール文字を表します。文字は ASCII コードの 32 から 126 の文字または英小文字です。「^?」は DEL コードを表します。

'\文字'　　「\」の次の文字を表します。この式でいう文字からは、数字と「o」「d」「x」は除きます。「\n」、「\t」、「\b」、「\r」、「\f」は C 言語のエスケープ

符号列と同じです。「\e」、「\E」はESC文字を表します。他の文字は、その文字自身を表します。

'\8進コード...' その8進コードを持つ文字を表します。

'\o8進コード...' その8進コードを持つ文字を表します。

'\d10進コード...' その10進コードを持つ文字を表します。

'\x16進コード...' その16進コードを持つ文字を表します。

(2) (関数名 評価すると文字になる式)

関数名	機能
toupper	引数がASCII英小文字ならば大文字に変える 例: (toupper a) → A
tolower	引数がASCII英大文字ならば小文字に変える 例: (tolower A) → a
toupdown	引数がASCII英小(大)文字ならば大(小)文字に変える 例: (toupdown a) → A (toupdown A) → a
tozenalpha	引数がASCII文字ならば対応する全角文字に変える 例: (tozenalpha A) → A
tohira	引数が全角カタカナならばひらがなに変える 例: (tohira ア) → あ
tokata	引数がひらがななならば全角カタカナに変える 例: (tokata あ) → ア
tozenhira	引数が半角カタカナならばひらがなに変える 例: (tozenhira ア) → あ ;「ア」は半角カタカナ
tozenkata	引数が半角カタカナならば全角カタカナに変える 例: (tozenkata ア) → ア ;「ア」は半角カタカナ
value	引数の文字コードを実際の数値にする 例: value 0 → '\x0' value A → '\xA' value F → '\xf'

(3) (関数名 評価すると文字になる式 評価すると文字になる式)

関数名	機能
+	引数の和を値にする 例: (+ あ '256') → ア (+ 0 (value 3)) → 3
–	引数の差を値にする
*	引数の積を値にする
/	引数の商を値にする

(4) (変数名) 関数名、機能、宣言名 (defvar) といずれも 一致しない英字で始まる英数字からなる文字列 (変数の項参照)。ただし、'_' も英字とみなします。

評価すると文字列になる式には、次のものがあります。

(1) "文字表記..." 文字表記を下記に示します (「評価すると文字になる式」 の文字表記とは異なります)。

文字	「"」、「^」、「\」を除く文字
^文字	コントロール文字を表します。文字は ASCIIコードの 32 から 126 の文字または英小文字です。^? は、DELコードを表します。
\文字	数字と「o」「d」「x」を除く文字。「\n」、「\t」、「\b」、「\r」、「\f」 はC 言語のエスケープ符号列と同じです。
\8 進コード... ...[;]	その 8 進コードに相当する文字を表します。後に数字が続く場合はセミコロン (;) が必要です。
\o8 進コード... ...[;]	その 8 進コードに相当する文字を表します。後に数字が続く場合はセミコロン (;) が必要です。
\d10 進コード... ...[;]	その 10 進コードに相当する文字を表します。後に数字が続く場合はセミコロン (;) が必要です。

\x16進コード. . . . . .[;]　その 16 進コードに相当する文字を表します。後に数字が続く場合はセミコロン (;) が必要です。

" " は空文字列を表します。

(2) (関数名 評価すると文字列になる式)

関数名	機能
tohankata	引数が全角ひらがなまたは全角カタカナならば半角カタカナに変える 例: tohankata が → ガ (半角)
last=	最後に照合した文字列の最後の文字と関数の引数 (評価すると文字になる式) が一致するかどうかを検査し、一致する場合 (last= 評価すると文字になる式) は空文字列になる。 例: last= A → あ ただし、last= は入力コードにだけ記述できる
todigit	第 1 引数で与えられたコードを第 2 引数のコードの進法の数に変換する
dakuadd	引数の後ろに濁点を付ける
handakuadd	引数の後ろに半濁点を付ける

(3) (関数名 モード名)　モード名はモード定義表で宣言された名前にしてください。

関数名	機能
if	引数であるモードが on なら、(if *mode_name*) は空文字列になる 例: (if katakana)VU ヴ
unless	引数であるモードが off なら、(unless *mode_name*) は空文字列になる 例: (unless katakana)VU ぶ
on	引数であるモードを on にする 例: (on katakana)
off	引数であるモードを off にする 例: (off katakana)
switch	引数であるモードが on なら off に、off なら on にする 例: (switch katakana)

ただし、if、unless は、入力コードにだけ記述できます。また、on、off、switch は、本処理表の出力コードにだけ記述できます。

(4) (関数名)　本処理表の出力コードにだけ記述できます。

関数名	機能
allon	すべてのモードを on にする
alloff	すべてのモードを off にする

機能　機能には下記の 2 つがあり、それぞれ単独で用いられます。

(error)　入力コードの部分のコードが入力されたら、エラーとみなします。

(restart)　前回のモード定義表を改めて読み込んで変換の再設定を行います。新しい変換表にエラーがあればエラーメッセージを表示したあと、元の変換表の設定に戻します。

変数宣言

(defvar 変数表記 (list 文字表記... ...))
　list は、その引数の文字を変域とします。

(defvar 変数表記 (all))
　all は、すべての文字を変域とします。

「評価すると文字になる式」として使用する変数とその変域の定義を行います。宣言は、上記の方法で行います。変数表記は、変数名または (変数名... ...) です。文字表記は、「評価すると文字になる式」の文字表記と同じです。

関数の使用上の注意　関数は「評価すると文字になる式」または「評価すると文字列になる式」であるため、(toupper (tolower Y)) と書けます。しかし、次のように評価すると文字列になる関数は他の関数の引数にできません。

(toupper (tohankata か))

変数

変数は同一パターンの変換が何通りもある場合に効果を発揮します。次に例を示します。

```
(defvar    a1    (list  K S T H Y R W G Z D B P))
(a1)(a1)    っ    (a1)
```

この 2 行は次の記述と同じ変換を行います (これはローマ字かな変換における促音処理です)。

KK	っ	K
SS	っ	S
TT	っ	T
...		
(中略)		
PP	っ	P

変数は、変数宣言 (defvar) で行った変域の文字を値とします。

変数使用上の注意事項

1. 使用する変数はその表の中において defvar で定義してください。
2. 変数の定義はその表の中全体で有効です。1 つの表で a1 という変数を定義し、別な表で a1 という変数を別様に定義することができます。この 2 つの a1 は別の変数として扱われます。そのとき、逆に 1 つの表の中で同名の変数を 2 度 defvar することはできません。
3. 対応表の 1 行の中では、同名の変数は常に同じ値を持ちます。

   ```
   (defvar a1 (list  A  B))
   (a1)(tolower (a1)) 3
   ```

   この場合、「Aa」、「Bb」という文字列が「3」に変換されます。「Ab」、「Ba」は変換されません。
4. 入力コードと表の入力コード部との照合は左から行われます。このため、表の入力コード部を左から見ていったとき、ある変数が特定の文字に一致する前に関数の引数として現れることはできません。

   ```
   (defvar a1 (list a b))
   (toupper (a1))(a1) 3
   ```

   この場合、「Aa」と入力しても、「3」には変換されません。これは、値が特定されていない状態で a1 が (toupper (a1)) に引数として現れるからです。このような設定は、表読み込み時にチェックされます。

5. 変数を出力コード部やバッファー残り部に書く場合も、入力コード部に現れる (つまり、入力コードとの照合で何らかの値を代入される) 変数でなければなりません。

```
(defvar  a1  (list  K  S))
(defvar  a2  (list  a))
(a1)(a1)   (a2)    (a1)
```

この場合、入力コード部で照合が行われない変数 a2 が出力コード部に現れているので規則 違反になります。

変換の方式

まず、入力されるコードを文字単位にまとめます (2 バイトコードの文字も 1 文字として扱います)。これを入力コードといいます。これは、前処理、本処理、後処理の 3 処理を経て最終出力となります。前処理では、入力コード 1 つに対して出力コード 1 つが対応付けられます。この出力コードが本処理コードの入力コードになります。

前処理表のうち現在使用されている表の入力コードの部分を上から順に見ていきます。最初に実際の入力コードと合致したところで、その行の出力コードの部分に書かれてあるものが出力されます。ただし、表が複数選択されているときは、モード定義表で先に参照されたものから見ていきます。また、実際の入力コードと合致するものが表にないとき (表が 1 つも選択されない時も含む) は、入力コードがそのまま出力されます。これは、本処理、後処理においても同様です。

本処理では、入力コードが表の入力コードの部分と合致する可能性がある (入力コードの部分の先頭から何文字かは一致している状態) 限り、入力コードは、バッファーに追加されます。バッファーに入力コードが追加されるごとに、本処理表の入力コードの部分を上から順に見ていってバッファーと比較されます。ここでバッファーの内容が表の入力コードの部分の最長のものと合致する可能性 (入力コードに部分の先頭から何文字かは一致している状態) があれば、変換未確定という判断のもとに、変換を行わずに次の入力コードを待ちます。

ただし、画面処理などの都合上、バッファー内のコードは未確定文字として出力されます。他に入力エラー、モード変更などを知らせるコードも出力されます。

これらの出力コードは本来の出力コードとは区別され、後処理は行われません。

バッファーの内容が表の入力コードの部分の最長のものと合致したら (同じ長さのものがあれば先に見つかったものと合致します)、その出力コードが出力されます。バッファー残りの部分がなければバッファー内の合致した部分が消され、あれば、バッファー内の合致した部分と入れ替わり、以上の操作が繰り返されます。

表に合致する可能性のあるものが見つからなかった場合、バッファーの先頭の 1 文字がそのまま出力されます。合致したものの出力コードの部分がモードの状態を変える関数 (on、off、switch など) である場合、使用される対応表がモード定義表の記述にしたがって変更されます。これらのモードを変える関数は、モードの状態 (on、off) に関係なく、絶えず使用される表に記述してください。また、機能の

(restart) の入力コードの部分に合致した場合は、モード定義表そのものの再読み込みが行われます。ただし、そのモード定義表は前回と同じファイルです。この機能を使うことで、オートマトンの使用中に修正した変換表 (モード定義表を含む) を、オートマトンを終えずにその変換内容を変えることができます。

後処理では、1 つの入力コードに対して 1 つ以上の出力コードが最終出力として出力されます。出力コードが 1 つ以上出力されることを除いて前処理と同じです。

下記の例では、「Ls」または「LS」が入力されたとき「ls –la (改行)」が出力されます。

前処理表

```
(defvar a1 (list s)) (a1) (toupper (a1))
```

本処理表

```
LS "LS –la\n"
```

後処理表

```
(defvar a1 (all)) (a1) (tolower (a1))
```

名前　　wnn_cvt_key_tbl – かな漢字変換フロントエンドプロセッサ (uum) キーコード変換表ファイル

形式　　`/usr/lib/locale/ja/wnn/cvt_key_tbl`

機能説明　　uum コマンドが本リリースから提供されなくなりましたので、 本ファイルも提供されなくなりました。

名前　wnn_cvt_xim_tbl – ファンクションキーおよびメタキー変換テーブル

形式　`/usr/lib/wnn/cvt_xim_tbl`

機能説明　ここで設定されたデータにしたがって、*KeySym* を 1 バイトの文字に変換します。変換されたキーコードは、`2A_CTRL` ファイル (初期設定) によって評価され、次に `uumkey` ファイルによって評価されます。

書式　*Keysym code*　「*Keysym*」と「*code*」の間は、半角スペースまたはタブで区切ります。セミコロン (;) ではじまる行はコメントとなります。

コードの記述法　8 進数、10 進数、16 進数で表記できます。? には整数値が入ります。

8 進数　`0??`

10 進数　`??`

16 進数　`0x??` または `0X??`

関連項目　`wnn_2A_CTRL(4)`, `uumkey(4)`

名前　wnnenvrc – かな漢字変換環境設定ファイル

形式　/usr/lib/wnn/ja_JP/wnnenvrc

機能説明　wnnenvrc は、かな漢字変換の環境を設定するものです。エントリが重複した場合、setdic (辞書ファイルの指定) 以外は最後に指定したものが採用されます。setdic は、128 個まで設定できます。セミコロン (;) で始まる行はコメントになります。

このファイルで設定できるものは、以下のとおりです。

- setdic <辞書ファイル名> <頻度ファイル名> <辞書プライオリティー> <辞書リードオンリーか否か> <頻度リードオンリーか否か> <辞書ファイルのパスワードの入っているファイル名> <頻度ファイルのパスワードの入っているファイル名> <仮名漢字変換 / 漢字仮名変換>

  <頻度ファイル名>
  指定した辞書の頻度ファイル名を指定します。ハイフン (–) を指定すると辞書ファイル内の頻度を使用します。ファイル名の先頭で次のような指定ができます。

!	クライアント側のファイルを使用
:	jserver 側のファイルを使用
指定しない	jserver 側のファイルを使用

  <辞書プライオリティー>
  辞書の優先度を 10 進数で指定します。

  <辞書リードオンリーか否か>
  次のように指定します。

1	リードオンリー
2	テンポラリ学習
3	グループ辞書
4	マージ辞書
0	上記以外

  リードオンリーを指定した場合、辞書ファイルの更新を行いません。

  テンポラリ学習を指定した場合、ファイルセーブ時に、ディスクに情報を書き込みません。

  1 つの辞書を複数ユーザーで同時に使用するときは、グループ辞書かマージ辞書で指定しておきます。

  <頻度リードオンリーか否か>
  次のように指定します。

1　　リードオンリー

2　　テンポラリ学習

0　　上記以外

リードオンリーを指定した場合、頻度ファイルの更新を行いません。

テンポラリ学習を指定した場合、ファイルセーブ時に、ディスクに情報を書き込みません。

<辞書ファイルのパスワードの入っているファイル名>
辞書ファイルのパスワードを記述したファイル名を指定します。

<頻度ファイルのパスワードの入っているファイル名>
頻度ファイルのパスワードを記述したファイル名を指定します。

<仮名漢字変換 / 漢字仮名変換>
仮名漢字変換の場合は 0 を、漢字仮名変換の場合は 1 を指定します。

引数の数が少ないとき、第 2 引数以降はデフォルトの値が使用されます。

```
- 5 0 0 - - 0
```

- set_fi_system_dic <FI 関係システム辞書ファイル名> <FI 関係頻度ファイル名> <FI 関係頻度リードオンリーか否か> <FI 関係頻度ファイルのパスワードの入っているファイル名>

  <FI 関係頻度ファイル名>
  指定した FI 関係システム辞書の頻度ファイル名を指定します。ハイフン (-) を指定すると FI 関係頻度更新は行いません。ファイル名の先頭で次のような指定ができます。

  !　　クライアント側のファイルを使用

  :　　jserver 側のファイルを使用

  指定しない　　jserver 側のファイルを使用

  <FI 関係頻度リードオンリーか否か>
  次のように指定します。

  1　　リードオンリー

  2　　テンポラリ学習

  3　　グループ辞書

  0　　上記以外

  リードオンリーを指定した場合、FI 関係頻度の更新を行いません。

テンポラリ学習を指定した場合、ファイルセーブ時に、ディスクに情報を書き込みません。

1 つの辞書または頻度を複数ユーザーで同時に使用するときは、グループ辞書で指定しておきます。

<FI 関係頻度ファイルのパスワードの入っているファイル名>
FI 関係頻度ファイルのパスワードを記述したファイル名を指定します。

- set_fi_user_dic <FI 関係ユーザー辞書ファイル名> <FI 関係ユーザー辞書リードオンリーか否か> <FI 関係ユーザー辞書ファイルのパスワードの入っているファイル名>

  <FI 関係ユーザー辞書ファイル名>
  FI 関係ユーザー辞書のファイル名の先頭で次のような指定ができます。

!	クライアント側のファイルを使用
:	jserver 側のファイルを使用
指定しない	jserver 側のファイルを使用

  <FI 関係ユーザー辞書リードオンリーか否か>
  次のように指定します。

1	リードオンリー
2	テンポラリ学習
3	グループ辞書
0	上記以外

  リードオンリーを指定した場合、FI 関係ユーザー辞書の更新を行いません。

  テンポラリ学習を指定した場合、ファイルセーブ時に、ディスクに情報を書き込みません。

  1 つの辞書を複数ユーザーで同時に使用するときは、グループ辞書で指定しておきます。

  <FI 関係ユーザー辞書ファイルのパスワードの入っているファイル名>
  FI 関係ユーザー辞書ファイルのパスワードを記述したファイル名を指定します。

- muhenkan_gakusyuu <無変換学習辞書ファイル名> <無変換学習辞書リードオンリーか否か> <無変換学習辞書プライオリティー> <無変換学習辞書ファイルのパスワードの入っているファイル名> <仮名漢字変換 / 漢字仮名変換>

  無変換学習とは、ひらがな、カタカナ、ローマ字で候補を確定した場合、その候補を無変換学習辞書に自動的に登録することです。

<無変換学習辞書ファイル名>
無変換学習辞書ファイル名の先頭で次のような指定ができます。

!	クライアント側のファイルを使用
:	jserver 側のファイルを使用
指定しない	jserver 側のファイルを使用

<無変換学習辞書リードオンリーか否か>
次のように指定します。

1	リードオンリー
2	テンポラリ学習
3	グループ辞書
0	上記以外

リードオンリーを指定した場合、無変換学習辞書の更新を行いません。

テンポラリ学習を指定した場合、ファイルセーブ時に、ディスクに情報を書き込みません。

1 つの辞書を複数ユーザーで同時に使用するときは、グループ辞書で指定しておきます。

<無変換学習辞書プライオリティー>
辞書の優先度を、10 進数で指定します。

<無変換学習辞書ファイルのパスワードの入っているファイル名>
無変換学習辞書ファイルのパスワードを記述したファイル名を指定します。

<仮名漢字変換 / 漢字仮名変換>
仮名漢字変換の場合は 0 を、漢字仮名変換の場合は 1 を指定します。

- bunsetsugiri_gakusyuu <文節切り学習辞書ファイル名> <文節切り学習辞書リードオンリーか否か> <文節切り学習辞書プライオリティー> <文節切り学習辞書ファイルのパスワードの入っているファイル名> <仮名漢字変換 / 漢字仮名変換>

  文節切り学習とは、確定時に文節を切り直したときに、切り直された箇所の前後の 2 文節を 1 つの候補として、文節切り学習辞書に登録することです。

  <文節切り学習辞書ファイル名>
  文節切り学習辞書ファイル名の先頭で次のような指定ができます。

!	クライアント側のファイルを使用
:	jserver 側のファイルを使用
指定しない	jserver 側のファイルを使用

<文節切り学習辞書リードオンリーか否か>
次のように指定します。

`1` リードオンリー

`2` テンポラリ学習

`3` グループ辞書

`0` 上記以外

リードオンリーを指定した場合、文節切り学習辞書の更新を行いません。

テンポラリ学習を指定した場合、ファイルセーブ時に、ディスクに情報を書き込みません。

1 つの辞書を複数ユーザーで同時に使用するときは、グループ辞書で指定しておきます。

<文節切り学習辞書プライオリティー>
辞書の優先度を、10 進数で指定します。

<文節切り学習辞書ファイルのパスワードの入っているファイル名>
文節切り学習辞書ファイルのパスワードを記述したファイル名を指定します。

<仮名漢字変換 / 漢字仮名変換>
仮名漢字変換の場合は `0` を、漢字仮名変換の場合は `1` を指定します。

- `setfuzokugo` *auxiliary_word_file_name*

  `setgrammar` *auxiliary_word_file_name*

  付属語ファイル名を指定します。

- `setparam` *number 0. . .number 16*

  10 個の変換パラメタ、および、疑似品詞の頻度を整数値で指定します。( ) 内は初期設定値です。

number 0	N (大) 文節解析のN (5)
number 1	大文節中の小文節の最大数 (10)
number 2	自立語の頻度に対する係数 (2)
number 3	小文節長に対する係数 (45)
number 4	自立語長に対する係数 (0)
number 5	使用ビットに対する係数 (80)
number 6	辞書に対する係数 (5)
number 7	小文節の評価値に対する係数 (1)

number 8	大文節長に対する係数 (20)
number 9	小文節数に対する係数 (0)
number 10	疑似品詞「数字」の頻度 (400)
number 11	疑似品詞「カナ」の頻度 (–100)
number 12	疑似品詞「英数」の頻度 (400)
number 13	疑似品詞「記号」の頻度 (80)
number 14	疑似品詞「閉括弧」の頻度 (200)
number 15	疑似品詞「付属語」の頻度 (2)
number 16	疑似品詞「開括弧」の頻度 (200)

- `confirm`

  このエントリ以降で指定された辞書、頻度ファイルが存在しない場合、ユーザーに対して新しく作るか否かの確認を行います。

- `confirm1`

  このエントリ以降で指定された辞書、頻度ファイルが存在しない場合、ユーザーに対して、新しく作るか否かの確認を 1 度だけ行い、それ以降はその値に従います。

- `create_without_confirm`

  このエントリ以降で指定された辞書、頻度ファイルが存在しない場合、無条件に新しく作ります。

- `no_create`

  このエントリ以降で指定された辞書、頻度ファイルが存在しない場合、新しく作りません。

- `saisyu_siyou` *last-useage_top-priority_processing_specification*

  最終使用最優先処理とは、同音異義語の中で直前に確定した候補を最初に出し、次候補リストには確定順に 6 つまで並べる処理です。最終使用最優先処理を行う場合は `TRUE`、行わない場合は `FALSE` を指定します。

- `fukugou_yuusen` *composite-word_priority_conversion_specification*

  複合語優先変換とは、付属語を含まない候補を優先する(「東京 – と」より「東京都」を優先する) 変換です。複合語優先変換を行う場合は `TRUE`、行わない場合は `FALSE` を指定します。

- `okuri_kijun` *Okurigana_processing_rule*

  送り基準処理とは、送り仮名のみ異なる候補 (「行う」と「行なう」) の変換を行うとき、指定された法則 (本則、送る、送らない) に一致する候補を最初に出す処理です。送り基準処理で本則候補を出す場合は `REGULATION`、 送る候補を出す場合は `YES`、 送らない候補を出す場合は `NO` を指定します。

- settou_kouho *initial_prefix_candidate*

  接頭語初期候補でひらがな候補を出す場合は HIRAGANA、漢字候補を出す場合は KANJI を指定します。

- rendaku *euphonic_change_processing_specification*

  連濁処理とは、連濁する候補 (会社「がいしゃ」) を最初に変換に使用しない処理です。連濁処理をを行う場合は TRUE、 行わない場合は FALSE を指定します。

- yuragi *long_vowel/alternate_spelling_processing_specification*

  長音・ゆらぎ処理とは、長音・ゆらぎを含む読み (「はなじ」を「鼻血」、「ほーほー」を「方法」) でも変換を行えるようにする処理です。長音・ゆらぎ処理を行う場合は TRUE、行わない場合は FALSE を指定します。

- okuri_gakusyu *Okurigana_process_learning*

  送り基準学習とは、直前に確定した送り基準法則を学習し、次回の変換に反映する処理です。送り基準学習を行う場合は TRUE、行わない場合は FALSE を指定します。

- settou_gakusyu *prefix_learning_specification*

  接頭語学習とは、直前に確定した接頭語候補情報（ひらがな、漢字）を学習し、次回の変換に反映する処理です。接頭語学習を行う場合は TRUE、 行わない場合は FALSE を指定します。

- setubi_gakusyu *suffix_learning_specification*

  接尾語学習とは、直前に確定した接尾語候補情報 (送る、送らない) を学習し、次回の変換に反映する処理です。接尾語学習を行う場合は TRUE、行わない場合は FALSE を指定します。

- hanyou_gakusyu *general_learning_specification*

  一般語学習とは、直前に確定した一般語候補情報 (ひらがな、カタカナ、漢字) を学習し、次回の変換に反映する処理です。一般語学習を行う場合は TRUE、行わない場合は FALSE を指定します。(一般語の例：子供、子ども、こども)

- hindo_kakuritu *Wnn8_frequency_increase_coefficient*

  Wnn8 頻度学習方法で、すぐ学習にする場合は HIGH、 普通学習にする場合は NORMAL、じわじわ学習にする場合は LOW、必ず学習する場合は ALWAYS、学習しない場合は NOT を指定します。

- fi_hindo_kakuritu *FI-related frequency_increase_coefficient*

  FI 関係頻度学習方法で、すぐ学習にする場合は HIGH、普通学習にする場合は NORMAL、じわじわ学習にする場合は LOW、必ず学習する場合は ALWAYS、学習しない場合は NOT を指定します。

- use_hinsi *part_of_speech_for_conversion_1 part_of_speech_for_conversion_2. . .*

  変換に使用する品詞名のリストを、空白文字またはタブで区切って指定します。

- unuse_hinsi *part_of_speech_not_used_1 part_of_speech_not_used_2...*
  変換に使用しない品詞名のリストを、空白文字またはタブで区切って指定します。
- giji_number *pseudo-number_initial_conversion_candidate*
  疑似数字の変換初期候補を指定します。

KANSUUJI	単位付きの漢数字にします
KANOLD	単位付きの旧漢数字にします
HANCAN	コンマ付きの半角数字にします
ZENCAN	コンマ付きの全角数字にします
HAN	コンマなしの半角数字にします
ZEN	コンマなしの全角数字にします
KAN	単位なしの漢数字にします

- giji_eisuu *pseudo_alphabet_initial_converson_candidate*
  疑似アルファベットの変換初期候補を、半角にする場合は HAN、全角にする場合は ZEN を指定します。
- giji_kigou *pseudo_symbol_initial_converson_candidate*
  疑似記号の変換初期候補を、半角にする場合は HAN、JIS候補にする場合は JIS、ASCII 候補にする場合は ASC を指定します。
- kutouten *period_input_mode*
  句読点入力モードを「、」「。」にする場合は TRUE、「，」「．」にする場合は FALSE を指定します。
- kakko *parenthesis_input_mode*
  括弧入力モードを「」にする場合は TRUE、［］にする場合は FALSE を指定します。
- kigou *symbol_input_mode*
  記号入力モードを「・」にする場合は TRUE、「／」にする場合は FALSE を指定します。
- yosoku_use *easy_input_mode_setting*
  楽々入力を使用する場合は TRUE、使用しない場合は FALSE を指定します。
- yosoku_max_disp *number_of_easy_input_candidate_lists*
  楽々入力候補一覧の表示数を、1 から 10 の範囲で指定します。
- yosoku_realtime *easy_input_candidate_auto*
  楽々入力候補を自動で表示させる場合は TRUE、表示させない場合は FALSE を指定します。

- boin_kabusoku *when_vowel_excess_and_deficiency_input_correct_setting*

  母音過不足時の入力補正を行なう場合は TRUE、行なわない場合は FALSE を指定します。

- shiin_choka *when_consonant_exceed_input_correct_setting*

  子音超過時の補正を行なう場合は TRUE、行なわない場合は FALSE を指定します。

- n_choka *when_N_exceed_input_correct_setting*

  N (ん) 超過時の補正を行なう場合は TRUE、行なわない場合は FALSE を指定します。

- nihongo_kousei *Japanese_proofread_setting*

  日本語校正を行なう場合は TRUE、行なわない場合は FALSE を指定します。

setdic、setjishopath、sethindopath の引数内では、最初の @USR を起動ユーザー名に展開します。

使用例

例 1 指定例

```
setdic usr/@USR/ud          -               15 0 0 - - 0
setdic system/kihon.dic  usr/@USR/kihon.h   6 1 0 - - 0
set_fi_user_dic          usr/@USR/fiud          0 -
```

注意事項

辞書、頻度のファイルは、Wnn8LE が接続する jserver が起動しているマシン上のファイル名を指定します。指定したユーザー辞書、頻度ファイルが存在しない場合、jserver 起動マシン上に作成されます。

関連項目

jserver(1M), uumrc(4)

名前　　wnn_hinsi.data – 品詞管理ファイル

形式　　/usr/lib/wnn/ja_JP/hinsi.data

機能説明　　このファイルは、幹語品詞に関する情報を管理するためのものです。幹語品詞に関する情報は、すべての辞書と接続情報ファイルの間で共通でないといけないので、このファイルを勝手にエディットすることは許されません (もし、勝手に変更すると、古い品詞管理ファイルを使用して作成した辞書、および接続情報ファイルの品詞番号の意味が、異なったものになってしまいます)。このファイルに定義されている順番に、品詞、複合品詞に番号が割り当てられ、その番号は、辞書、品詞ファイルを作る時、及びサーバー、クライアントで番号から品詞名を引くため、および、複合品詞に対しそれを構成する品詞の集合を引くために用いられます。

この割り当てられる番号は 0 からの昇順になっています。このファイルに対して許される操作は、新しい品詞および複合品詞をファイルの最後に付け加えることと、@ のみからなる行を品詞、複合品詞の定義に置き換えることだけです。エントリの削除は決してしないでください。@ は、そこの品詞名は定めないが、取り敢えず領域を取っておくことを意味します。

hinsi.data の各行の品詞の書式は次のとおりです。

```
品詞
複合品詞$品詞:品詞: . . . :品詞
```

ただし、複合品詞の定義で現れる品詞は、それより前に品詞として定義されていなくてはなりません。また、同じ名前を持つ品詞、あるいは複合品詞が存在していてはいけません。セミコロン (;) に出合うと、行末までコメントとして無視されます。

このファイルに関する情報 (品詞番号から品詞名を引く、あるいは、複合品詞に対しその構成要素を引く) は、ライブラリによって提供されますので、クライアントプロセスから参照できます。

使用例

```
先頭          ;文節先頭
名詞          ;これは、名詞が 8 番であることを意味しています
一段
一段名$一段:名詞      ;これは複合品詞
@
@
```

名前　wnnhosts – アクセス制御設定

機能説明　wnnhosts は、変換サーバーに接続できるホストやユーザーを制限することができます。初期状態では wnnhosts は用意されていないため、アクセス制御を行なう場合に新規に作成する必要があります。wnnsysenv_server から設定を行うと、wnnhosts は自動的に作成されます。wnnhosts 変更後は、jserver の再起動が必要となります。

参照順位
1. jserver の起動オプション -A での指定
2. /etc/wnn/wnnhosts (システム設定)

書式

```
[Conversion_Server_Name]
  [Language_of_Conversion_Server]
  [Server_Host_Name:No] [Server_Host_Name/No]  {
    [Access_Data]
    [Access_Data]
      :
}
```

[*Conversion_Server_Name*] と [*Language_of_Conversion_Server*] は、Wnn8 ではそれぞれ jserver と ja_JP が固定値となります。

[*Server_Host_Name*] または [*Server_Host_Name*:*No*] または [*Server_Host_Name*/*No*] と { の間は、必ず半角空白文字で区切ります。

ポート番号の省略時は、jserver の標準ポート (22273) が適用されます。

[:*No*] とコロンで区切ると相対ポート番号となり、jserver の標準ポート番号 (22273) に設定値を加えた値をポート番号とします。例えば:1 とした場合は 22274 が使用されます。[/*No*] とスラッシュで区切ると絶対ポート番号となり、設定値をそのままポート番号として使用します。

セミコロン (;) ではじまる行はコメントとなります。

Access_Data は、次の書式で指定します。

Host_Name	このホストからはすべてのユーザーがアクセス可能になります。
Host_Name:*User_name*, *User_name*, . . .	このホストからは、ユーザー名で指定したユーザーのみがアクセス可能となります。複数のユーザーを指定する場合はコンマ (,) で区切ります。
@User_name	このユーザーはすべてのホストからアクセス可能になります。

名前　wnnlerc – Wnn8LE 起動時オプション設定ファイル

形式　`/usr/lib/wnn/ja_JP/wnnlerc`

機能説明　`wnnlerc` ファイルは、Wnn8LE の環境を設定するファイルです。ユーザごとに異なった設定が可能です。同じエントリがある場合は、後のものが優先されます。セミコロン (;) で始まる行は、コメント行です。

Wnn8LE は、以下の順序でこのファイルを検索します。

1. `$HOME/.Wnn8/wnnlerc` (ユーザ設定)
2. `/etc/wnn/ja_JP/wnnlerc` (システム設定)
3. `/usr/lib/wnn/ja_JP/wnnlerc` (初期設定)

エントリは次の形式で指定します。

<エントリ> <設定値> ...

設定値には言語名やファイル名などが入ります。エントリと設定値は空白文字またはタブ文字で区切ります。

この設定ファイルで記述できるものは、以下のとおりです。それらが設定されない場合は、デフォルトが使用されます。

`set_jserver` <host> | <host>:No| <host>/No
変換サーバー `jserver` を起動しているホスト名を host で、ポート番号を No で指定します。:No と指定すると相対ポート番号となり、`jserver` の標準ポート番号 (22273) に No を加えた値をポート番号とします。例えば :1 とした場合は 22274 が使用されます。/No と指定すると絶対ポート番号となり、No の値をそのままポート番号として使用します。それぞれ省略時は `localhost` の標準ポート (22273) が適用されます。`jserver` の省略時には、以下の優先順位で設定されます。

1. 環境変数 (`JSERVER`) の設定
2. `/etc/hosts` ファイルにおける設定 (`jserver`)
3. `Unix-Domain`

`set_geometry` <X>+<Y>
Wnn8LE のツールパレットのオープン位置を指定します。原点は画面左上として、X は原点からの X 座標、Y は原点からの Y 座標を指定します。

`henkanon_map` <ツールパレットを変換 ON 時のみ表示するか否か>
Wnn8LE のツールパレットを、変換 ON 時のみ表示／常時表示するかを指定します。

`TRUE`	変換 ON 時のみ表示
`FALSE`	常時表示

設定例

例 1 指定例

```
set_jserver     localhost
set_geometry    +20-50
henkan_on_map    TRUE
```

名前　wnn_mode – モード定義表

形式　/usr/lib/wnn/ja_JP/rk/mode

機能説明　モード定義表は、モードの宣言、各モードで使用する対応表の組み合わせとその決定規則を記述しておくファイルであり、Wnn8LE 起動時に解釈されます。このモード定義表は、オートマトンの変換表の「目次」に相当するものです。

Wnn8LE は、モード定義表を以下の順番で検索します。

1. Wnn8LE 初期化ファイル uumrc の setrkfile エントリによる指定
2. ファイル名 /usr/lib/wnn/ja_JP/rk/mode

wnn_mode の記述法については、wnn_automaton(4) のマニュアルページを参照してください。

関連項目　uumrc(4), wnn_automaton(4)

名前　wnn_serverdefs – Wnn6 かな漢字変換サーバー接続パラメタ設定ファイル

形式　`/etc/lib/locale/ja/wnn/serverdefs`

機能説明　wnn_serverdefs ファイルは本リリースから提供されなくなりました。

名前　wnn_ximrc – 入力クライアント環境設定ファイル

形式　/usr/lib/wnn/ximrc

機能説明

ximrc ファイルは、Wnn8LE の環境を設定するファイルです。ユーザーごとに異なった設定が可能です。同じエントリがある場合は、後のものが優先されます。セミコロン (;) で始まる行は、コメント行です。

Wnn8LE は、以下の順序でこのファイルを検索します。

1. $HOME/.Wnn8/ximrc (ユーザー設定)
2. /etc/wnn/ximrc (システム設定)
3. /usr/lib/wnn/ximrc (初期設定)

エントリは次の形式で指定します。

<エントリ> <設定値> ...

設定値には言語名やファイル名などが入ります。エントリと設定値は空白文字またはタブ文字で区切ります。セミコロン (;) で始まる行はコメントとなります。

この設定ファイルで記述できるものは、以下のとおりです。それらが設定されない場合は、デフォルトが使用されます。

setuumrc *language uumrc*
: 使用する言語に対応する uumrc ファイルを指定します。省略時には uumrc の参照順位が適用されます。

preloadrkfile *language_name*
: ローマ字変換の言語名を指定します。その言語に対応する uumrc で指定されたローマ字定義ファイルを読み込みます。

setlayout *format*
: 次候補一覧のレイアウト形式を指定します。指定方法は以下のとおりです。

  multi　マルチカラム

  vert　縦方向

  horiz　横方向

  デフォルトは multi です。

使用例

例 1 指定例

```
;setuumrc  ja_JP  /usr/lib/wnn/ja_JP/uumrc.xim
;preloadrkfile  ja_JP
setlayout  multi
setposition  spot
```