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: 816–3998–10
2002 年 5 月

020321@2851

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。

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。

JFP リファレンスマニュアル 4 : ファイル形式

Intro_jfp(4)

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

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

機能説明 | atok12wordlist は、ATOK12 辞書メンテナンスで使用するテキスト形式の単語ファイルで、atok12(1) の辞書ユーティリティの機能の一部で入出力に使われます。

単語ファイルの形式は以下のように定義されています。

ファイルの先頭行

先頭行は次の記述で始まらなければなりません。

```
!ATOK12
```

この場合、"!" は半角の文字にしてください。"ATOK12" は、半角または対応する全角の文字を使うことが可能です。

コメント行

! (半角・全角どちらかの文字も使用可能) で始まる行は、最初の行を除いてコメント行として無視されます。

品詞名を使った単語指定

単語は、次のような記述方法で指定します。

```
読み,表記,品詞名
```

区切り文字として、コンマまたは読点を使用できます。表記が区切り文字を含んでいる場合、表記を二重引用符または 単一引用符で囲みます。区切り文字と引用符は、半角または全角のどちらの文字でも使用可能です。品詞名の種類は、「可能な品詞」の項を参照してください。品詞名については、日本語で記述してください。

品詞番号を使った単語指定

単語は、次のような記述方法でも指定できます。

```
読み,表記,品詞番号
```

この形式は、品詞を番号で指定することを除いて上記と同一です。品詞番号については、「可能な品詞」の項を参照してください。

可能な読み

長さ：16 文字まで

濁点や半濁点もそれ自身を 1 文字として計算します。

文字：

半角と全角の文字のうち、次の文字が使用できます。ただし、辞書に格納される際には読みの文字は 半角の対応する文字に変換され、格納されます。

- 全角のひらがな
- 次の半角と全角の文字

```
カタカナ
アルファベット
数字
濁点
半濁点
-
+
*
```

/
_
#
$
%
&
=
@
:
;
<
>読みの先頭の文字に限って、以下の文字は使用できません。

- 次の半角と全角の文字

ひらがな・カタカナの "を"
ひらがな・カタカナの "ん"
長音
ひらがな・カタカナの "ぁ", "ぃ", "ぅ", "ぇ", "ぉ", "ゃ", "ゅ", "ょ"
ひらがな・カタカナの促音 "っ"
濁点
半濁点

読みの順序: 辞書内では読みは半角の対応する文字に変換され、格納されます。読みの順序は JIS-X0201 文字セット定義のコード順です。辞書メンテナンスツールなどで一覧表示や読みの範囲指定を行う場合はこの順序が使われます。

可能な表記

長さ: 50 文字まで。半角は 1 文字を 1、全角は 1 文字を 2 と数えます。

文字： すべての半角と全角の文字

可能な品詞 全部で 33 個の品詞があります。品詞名は日本語で記述してください。

品詞番号	品詞名
1	名詞
4	固有人名
5	固有地名
6	固有組織
8	固有一般
9	名詞サ変

品詞番号	品詞名
10	名詞ザ変
11	名詞形動
13	数詞
14	副詞
15	連体詞
16	接続詞
17	感動詞
18	独立語
19	接頭語
21	接尾辞
23	カ行五段
24	ガ行五段
25	サ行五段
26	タ行五段
27	ナ行五段
32	ハ行五段
28	バ行五段
29	マ行五段
30	ラ行五段
31	ワ行五段
33	一段動詞
34	カ変動詞
35	サ変動詞
36	ザ変動詞
37	形容詞
39	形容動詞
41	単漢字

使用例

以下は単語ファイルの記述例です。

1.

```
   !ATOK12
いち、一太郎、固有一般
FD、フロッピィディスク、名詞
たろはな，"一太郎，花子"，1
```

2.

```
   !ATOK12
!====================
!company,txt
!取引先上位 200 社
!1999.04.01 現在
!====================
さん,サン・マイクロシステムズ株式会社,固有組織
XXX,XXX有限会社,固有組織
```

関連項目

atok12(1)

ATOK12 ユーザーズガイド

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)

jserverrc(4)

名前 | jserverrc – Wnn6 かな漢字変換サーバーの初期化ファイル

形式 | `/etc/lib/locale/ja/wnn/ja/jserverrc`

機能説明 | `jserverrc` は Wnn6 かな漢字変換サーバー (`jserver`) の初期化ファイルで、`jserver` が起動する時に読み込まれます。設定できるエントリを以下に示します。

`readfile` *dictionary_filename*
: *dictionary_filename* には、サーバーが立ち上がる場合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、サーバーが立ち上がる時に読み込まれ、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、各クライアントの起動時に、辞書 を読み込む時間を節約するために使用されます。また `wnnds` を使用している時に、`jserver` にファイルを読み込ませたい場合は、ファイル名の先頭にコロン (`:`) を付けます。 たとえば次のようになります。

```
readfile :iwanami/fisd
```

`max_client` *number*
: *number* には、接続できるクライアントの最大数を指定します。デフォルトは `64` です。

`max_sticky_env` *number*
: *number* には、固定化する環境の最大数を指定します。環境を固定化すると、クライアントとの接続が切れ、環境を解放できる状態になっても、 その環境は維持されます。これにより、次にその環境を使用する 時に環境設定を省くことができるため、立上りが早くなります。デフォルトは `10` です。

`jserver_dir` *path*
: *path* には、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 `uum` 環境ファイルのディレクトリパス名 (`/usr/lib/locale/ja/wnn`)。

  `@LANG` Solaris の日本語環境では ja です。

`def_param` *number_0. . .number_16*
: かな漢字変換のパラメタと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。

`max_param` *number_0. . .number_16*

かな漢字変換のパラメ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がデフォルト値より小さい場合、デ フォルトと同じ値が設定されます。上限値が 2 回以上設定された場合、最後に設定された値が有効になります。

`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がデフォルト値より大きい場合、デ フォルトと同じ値が設定されます。下限値が 2 回以上設定された場合、最後に設定された値が有効になります。

`set_giji_eisuu` *character. . .*

指定した文字を、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。 *character* は次のように表記します。

文字	表記法
`CTRL_A`	`^A`
' '(SPACE)	`0x20 \x20 32 040 \o40 040 \o40` : 8 進数

	32 : 10 進数 0x20 \x20: 16 進数
' _'	' _'

`default_wnnds_list` *wnnds_info1 wnnds_info2 wnnds_info3*

`jserver` 起動時にデフォルト用として接続する辞書引きサーバー `wnnds` に関するオプション (`-ds` 、 `+ds`) が指定されなかった場合、このエントリで指定された `wnnds` をデフォルトの `wnnds` として使用します。

`wnnds` は次の書式で指定します。

hostname	ホスト *hostname* 上で、標準のポート番号 (26208) を使用する `wnnds`
hostname:no	ホスト *hostname* 上で、(標準のポート番号 + *no*) をポート番号として使用する `wnnds`
hostname/port_no	ホスト *hostname* 上で、 *port_no* をポート番号として使用する `wnnds`

`wnnds` は最大 3 つ まで指定でき、1 番目の `wnnds` から順に接続を試みて、最初に接続に成功した `wnnds` をデフォルトの `wnnds` として使用します。

例 : `default_wnnds_list wnnds1/26209 wnnds2 wnnds3:1`

関連項目 `wnnenvutil`(1), `jserver`(1M), `wnnds`(1M)

名前 | sdtudc_map – ユーザー定義文字変換マップ

形式 | `/usr/dt/config/$LANG/sdtudc_map`

機能説明 | `sdtudc_map` は `sdtudc_convert`(1) 、 `sdtudc_extract`(1) のデフォルトの文字変換マップファイルで、Solaris 2.5.1 以前のユーザー定義文字の利用環境から Solaris 2.6 およびその互換バージョンのユーザー定義文字の利用環境へ移行する場合に参照されます。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できます。

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。

各行には変換される文字コードと変換先の文字コードを記述します。 各変換元の初めと終わりは '`,`' で、変換元と変換先は '`\t`' で区切らなければなりません。

ja ロケールの場合の初期値は以下の通りです。これは、コードセット 1 の 9 区 - 15 区のコードポイントをコードセット 1 の 85 区 - 91 区に変換する例です。

```
a9a1,a9ff f5a1
aaa1,aaff f6a1
aba1,abff f7a1
aca1,acff f8a1
ada1,adff f9a1
aea1,aeff faa1
afa1,afff fba1
```

一行目を例にとると、`sdtudc_extract`(1) では、指定フォントファイルから 0xa9a1 - 0xa9ff 間に登録されているユーザー定義 文字を抜き出し、0xf5a1 から始まる値に変換します。また、`sdtudc_convert`(1) では、テキスト中の 0xa9a1 - 0xa9ff 間のコードポイントを 0xf5a1 から始まるコードポイントに変換します。

ファイル | `/usr/dt/config/$LANG/sdtudc_map`

関連項目 | `sdtudc_convert`(1), `sdtudc_extract`(1), `sdtudctool`(1)

注意事項 | 変換先のコードポイントには、必ず、`f5a1` 以上の値を指定してください。

uumkey(4)

名前　uumkey – Wnn6 かな漢字変換操作キー割り当て定義ファイル

形式　`/usr/lib/locale/ja/wnn/ja/uumkey`

機能説明　`uumkey` は、かな漢字変換操作で使用するキーの割り当てを行います。 ユーザーごとに `uumkey` ファイルを設定することができます。

エントリの形式　エントリは次の形式で指定します。

`include`	`uumkey` ファイル名
`unset`	機能エントリ
機能エントリ	キーコード [キーコード...] `include` と `unset` は機能外エントリと呼ばれます。

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 (;) またはコロン (:) で始まる行は、コメントとして無視されます。

キーコードは、プログラミング言語 C と同じ書式の 8 進数、10 進数、16 進数で指定します。また、コントロールキーを使用した入力に関しては、「^A」のように表記できます。

各機能エントリに対して、最大 10 個のキーコードを設定できます。

同じ機能エントリを複数回記述すると、最後に記述したものが有効になります。1 つの機能エントリに複数のキーコードを指定する場合は、1 行で設定してください。

同一のかな漢字変換の操作モードで作動する異なる機能エントリに同じキーコードを設定することはできません。

キーコードとして処理できる数値は、0 以上 512 未満の整数です。キーボードで発生できないキーコード (128 以上の整数など) は、キーコード変換表(`cvt_xim_tbl/cvt_key_tbl`) または オートマトンでそのコードを発生させることができます。

機能外エントリ

`include` *uumkey_filename*	指定された `uumkey` ファイルを取り込みます。
`unset` *entry*	指定された機能エントリ *entry* に対するキーの割り当てを取り消します。

かな漢字変換の操作モード　各機能エントリは、特定の操作モードで使用できます。

操作モード	内容
0	変換後の変換結果を修正している状態、インスペクトの状態
1	変換前の文字を入力している状態
2	変換後に文節の長さを伸縮している状態
3	変換する文字列が入力されていない状態 (入力バッファが空)

操作モード	内容
4	候補選択している状態。 uum では、単語登録の品詞選択、辞書選択 などの状態も含む

機能エントリ

名前の最後に _e が付いた機能エントリは、 _e が付かない同名の機能エントリが動作する操作モードに加え、 操作モード 3 (入力バッファーが空) の状態でも動作します。

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 。全モードで適用される
send_string	012	変換行内の文字列と入力されたキーコード (send_string に割り当てられているキーコード) を連結し、アプリケーションに渡す (確定処理)
kakutei	012	変換行内の文字列をアプリケーションに送る (確定処理)
forward_char	1	1 文字後ろ右へカーソル移動
backward_char	1	1 文字前左へカーソル移動。
goto_top_of_line	1	行の先頭の文字へカーソル移動
goto_end_of_line	1	行の後端の文字へカーソル移動
delete_char_at_cursor	1	カーソル位置の文字を消去
fiwnn	012	「学習/変換/表示モード」ウィンドウを開く
fiwnn_e	0123	「学習/変換/表示モード」ウィンドウを開く。入力バッファーが空の状態でも動作する
hindo	012	学習情報を保存する
hindo_e	0123	学習情報を保存する。入力バッファーが空の状態でも動作する
kensaku	012	「単語削除・編集」ウィンドウを開く
kensaku_e	0123	「単語削除・編集」ウィンドウを開く。入力バッファーが空の状態でも動作する
keybind	012	「入力スタイル」ウィンドウを開く

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
keybind_e	0123	「入力スタイル」ウィンドウを開く。入力バッファーが空の状態でも動作する
kaijo	02	カーソル位置の文節以降の変換された文字列を、変換前の状態に戻す
henkan	1	連文節変換を行う
tan_henkan	1	小文節 1 文節として単文節変換を行う
tan_henkan_dai	1	大文節 1 文節として単文節変換を行う
nobi_henkan	2	文節の伸縮時に、反転している部分を小文節として単文節変換して、それ以降を連文節変換する
nobi_henkan_dai	2	文節の伸縮時に、反転している部分を大文節として単文節変換して、それ以降を連文節変換する
jikouho	0	小文節として次候補を表示
jikouho_dai	0	大文節として次候補を表示
zenkouho	0	小文節として前候補を表示
zenkouho_dai	0	大文節として前候補を表示
select_jikouho	0	小文節として候補一覧を表示
select_jikouho_dai	1	大文節として候補一覧を表示
kana_henkan	1	漢字かな変換を行う。ただし、逆引き形式 (登録可能形式で、漢 → かな変換が行える) 辞書のみ有効

表 2 基本機能エントリ

機能エントリ	操作モード	機能
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、kill buffer に蓄積
yank	1	kill buffer の内容を、カーソル位置に挿入

表 2 基本機能エントリ　(続き)

機能エントリ	操作モード	機能
yank_e	13	kill buffer の内容を、カーソルの位置に挿入。入力バッファーが空の状態でも動作する
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
jisho_utility	012	Wnn6 メニューを開く
jisho_utility_e	0123	Wnn6 メニューを開く。入力バッファーが空の状態でも動作する
touroku	012	「単語登録」ウィンドウを開く
touroku_e	0123	「単語登録」ウィンドウを開く。入力バッファーが空の状態でも動作する
sainyuuryoku	1	入力バッファの内容を、前に入力したかな文字列に置き換える
sainyuuryoku_e	13	入力バッファの内容を、前に入力したかな文字列に置き換える。入力バッファーが空の状態でも動作する
kuten	1	区点番号入力を行う
kuten_e	13	区点番号入力を行う。入力バッファーが空の状態でも動作する
jis	1	16 進コード入力を行う
jis_e	13	16 進コード入力を行う。入力バッファーが空の状態でも動作する
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。入力バッファーが空の状態でも動作する

表 3 基本機能エントリ

機能エントリ	操作モード	機能
previous_history	1	入力バッファの内容を、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

表 3 基本機能エントリ　(続き)

機能エントリ	操作モード	機能
previous_history_e	13	入力バッファの内容を、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。入力バッファーが空の状態でも動作する
next_history	1	入力バッファの内容を、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	入力バッファの内容を、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。入力バッファーが空の状態でも動作する
all_history	1	入力バッファの内容を、ヒストリに記憶されているすべての文字列に置き換える
all_history_e	13	入力バッファの内容を、ヒストリに記憶されているすべての文字列に置き換える。入力バッファーが空の状態でも動作する
bushu	1	部首入力を行う。uum では使用しない
bushu_e	13	部首入力を行う。uum では使用しない。入力バッファーが空の状態でも動作する
com_entry		uum で、単語と辞書にコメントを付加する
greek	13	ギリシャ文字一覧入力を行う。uum では使用しない
hankaku	012	半角文字に変換する。uum では使用しない
hiragana	012	ひらがなに変換する。uum では使用しない
jis2	13	補助漢字 16 進コード入力を行う。uum では使用しない
katakana	012	カタカナに変換する。uum では使用しない
russian	13	キリール文字 (ロシア語) 一覧入力を行う。uum では使用しない
select_ikeiji_dai	0	大文節異形字変換を行う

表 3 基本機能エントリ　(続き)

機能エントリ	操作モード	機能
tankan_henkan	01	単漢字変換を行う。uum では使用しない
kuten2	13	補助漢字区点番号入力を行う。uum では使用しない
next_page	4	次の候補群を表示する。uum では使用しない

表 4 基本機能エントリ

機能エントリ	操作モード	機能
tankan_nobi_henkan	2	単漢字変換を行う。uum では使用しない
tel_henkan	01	電話番号変換を行う。uum では使用しない
tel_nobi_henkan	2	電話番号変換を行う。uum では使用しない
zip_henkan	01	郵便番号変換を行う。uum では使用しない
zip_nobi_henkan	2	郵便番号変換を行う。uum では使用しない
one_char_kakutei	012	1 文字確定を行う。uum では使用しない
one_char_no_henkan	01	N 文字無変換を行う。uum では使用しない
previous_page	4	前の候補群を表示する。uum では使用しない
kigou	13	記号一覧入力を行う。uum では使用しない
eisuu	012	英数字に変換する。uum では使用しない
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

表 4 基本機能エントリ　(続き)

機能エントリ	操作モード	機能
quote	1	次の入力文字が henkan_on に割り当てられた文字でなければ、そのまま入力バッファーに取り込む。その場合、入力された文字はローマ字かな変換の対象とはならず、割り当てられている機能も無視される
quote_e	13	quote と同じ。入力バッファーが空の状態でも動作する
forward_select	4	次 (右) の候補に移動する。必要に応じて、次の候補群が表示される
backward_select	4	前 (左) の候補に移動する。必要に応じて、前の候補群が表示される
next_select	4	次の行 (下) の候補に移動する。必要に応じて、次の候補群が表示される
previous_select	4	前の行 (上)の候補に移動する。必要に応じて、前の候補群が表示される
linestart_select	4	左端の候補に移動する
lineend_select	4	右端の候補に移動する
select_select	4	候補を選択し、操作モード 0 に移行する

表 5 基本機能エントリ

機能エントリ	操作モード	機能
send_ascii_char	01234	入力バッファーが空の状態で ASCII 文字が入力された場合、入力バッファーに蓄積しない (ASCII 文字送りをする)
not_send_ascii_char	01234	入力バッファーが空の状態で ASCII 文字が入力された場合、入力バッファーに蓄積する (ASCII 文字送りをしない)
pop_send_ascii_char	01234	入力バッファーが空の状態で ASCII 文字が入力された場合、前の ASCII 文字送りの状態に戻す
togle_send_ascii_char	01234	uum で、入力バッファーが空の状態で ASCII 文字が入力された場合、ASCII 文字送りの状態を反転する

表 **5** 基本機能エントリ　*(続き)*

機能エントリ	操作モード	機能
quote_send_ascii_char	3	ASCII 文字送りをする状態で入力バッファーが空の場合、次に入力された ASCII 文字を入力バッファーに蓄積する
reconnect_jserver	01234	かな漢字変換サーバー (jserver) と再接続する
inspect	0	候補の情報を表示する
sakujo_kouho	0	候補をかな漢字変換辞書から削除する
forward_bunsetsu	0	一文節右へ移動する。明示的にキーが割り当てられていない場合は、forward_char に割り当てられたキーが割り当てられる。uum では使用しない
henkan_forward	2	長さの調節の対象となっている文節を単文節変換、それ以降を連文節変換し、一つ先の文節へ移動する。明示的にキーに割り当てられていない場合は、forward_char に割り当てられたキーに割り当てられる
backward_bunsetsu	0	一文節左へ移動する。明示的にキーに割り当てられていない場合は、backward_char に割り当てられたキーに割り当てられる。uum では使用しない
henkan_backward	2	長さの調節の対象となっている文節を単文節変換、それ以降を連文節変換し、一つ前の文節へ移動する。明示的にキーに割り当てられていない場合は、backward_char に割り当てられたキーに割り当てられる
top_bunsetsu	0	先頭の文節へ移動する。明示的にキーが割り当てられていない場合は、goto_top_of_line が割り当てられたキーに割り当てられる。uum では使用しない

表 **6** 基本機能エントリ

機能エントリ	操作モード	機能
end_bunsetsu	0	最後の文節へ移動する。明示的にキーが割り当てられていない場合は、goto_end_of_line に割り当てられたキーが割り当てられる。uum では使用しない
jmptijime	2	先頭の文字へ移動する。明示的にキーが割り当てられていない場合は、goto_top_of_line に割り当てられたキーが割り当てられる。uum では使用しない
c_end_nobi	2	最後の文字へ移動する。明示的にキーが割り当てられていない場合は、goto_end_of_line に割り当てられたキーが割り当てられる。uum では使用しない
forward	2	文節を一文字長くする。明示的にキーが割り当てられていない場合は、bunsetu_nobasi に割り当てられたキーが割り当てられる。uum では使用しない
chijime	2	文節を一文字短くする。明示的にキーが割り当てられていない場合は、bunsetu_chijime に割り当てられたキーが割り当てられる。uum では使用しない
rubout	1	カーソルの前の文字を消去する。このキーはローマ字かな変換処理でも使用される。キーコードは 0 から 255 の値をとる必要がある。uum では使用しない
next_ku_kuten	4	区点番号入力候補一覧ウインドウで、次の区の候補群を表示する。uum では使用しない
previous_ku_kuten	4	区点番号入力候補一覧ウインドウで、前の区の候補群を表示する。uum では使用しない

表 7 ATOK8 風入力スタイルで使用される機能エントリ

機能エントリ	操作モード	機能
bubun_kakutei	02	部分確定。先頭の一文節を確定する。後続の文節は続けて操作できる。uum では使用しない
ichi_oto_kakutei	1	一音確定。先頭のかな一文字を確定する。後続の文節は続けて操作できる。uum では使用しない
bunsetu_top_kakutei	02	一文字確定 (先頭)。選択された文節 (注目文節) の先頭の一文字を確定する。それ以外の文字はすべて破棄する。uum では使用しない
bunsetu_end_kakutei	02	一文字確定 (末尾)。選択された文節 (注目文節) の末尾の一文字を確定する。それ以外の文字はすべて破棄する。uum では使用しない
bunsetu_kakutei	012	対象文節確定。先頭の文節から選択された文節 (注目文節) まで確定する。後続の文節は続けて操作できる。uum では使用しない
atok_jikouho	4	次候補。変換行内に次候補を表示し、候補一覧選択ウィンドウ内で次候補を選択する。uum では使用しない
hankaku_space_input	0123	半角スペースを入力する。uum では使用しない
sjis	1	PC 漢字 16 進コード入力を行う。uum では使用しない
sjis_e	13	PC 漢字 16 進コード入力を行う。入力バッファーが空の状態でも動作する。uum では使用しない
atok_zenkouho	4	前候補。変換行内に前候補を表示し、候補一覧選択ウィンドウ内で前候補を選択する。uum では使用しない
atok_jikouho_gun	4	次の候補群を表示する。uum では使用しない
atok_jikouho_gun	4	前の候補群を表示する。uum では使用しない

表 **7** ATOK8 風入力スタイルで使用される機能エントリ　(続き)

機能エントリ	操作モード	機能
top_kigou_kuten	4	区点番号入力候補一覧ウインドウで、記号の先頭の区を表示する。uum では使用しない
top_gaiji_kuten	4	区点番号入力候補一覧ウインドウで、ユーザー定義文字として使用できる領域の先頭の区を表示する。uum では使用しない
atok_kill	0124	削除。未確定の文字をすべて削除する。uum では使用しない
atok_kaijo	012	解除。注目文節以降を読みに戻す。uum では使用しない
zen__kaijo	0	全変換解除 (全体) 。未確定の文字をすべて読みに戻す。uum では使用しない
atok_rubout	1	カーソルの前の文字を消去する。ただし、カーソルの前に変換後 (未確定) の文節がある場合は、その文節を読みに戻す。uum では使用しない
new_backward_char	1	カーソルを一文字前へ移動する。ただし、カーソルの前に変換後 (未確定) の文節がある場合は、その文節を読みに戻す。uum では使用しない
zenkaku_space_input	0123	操作モード 0: 次候補を表示する 操作モード 13: 全角スペースを入力する 操作モード 2: 単文節変換を行う uum では使用しない
atok_select_jikouho	0	ATOK 風の次候補一覧を表示する。uum では使用しない

表 **8** ATOK7 風入力スタイルで使用される機能エントリ

機能エントリ	操作モード	機能
atok_bubun_kakutei	01	部分確定する。uum では使用しない

表 **9** cs00 風入力スタイルで使用される機能エントリ

機能エントリ	操作モード	機能
send_string_off	0123	未確定の文字を確定し、変換をオフにする。uum では使用しない

表 10 EGBRIDGE 風入力スタイルで使用される機能エントリ

機能エントリ	操作モード	機能
eg_zenkaku_eisuu	012	全角英数字に変換する。uum では使用しない
eg_hankaku_katakana	012	半角カタカナに変換する。uum では使用しない
eg_hankaku_eisuu	012	半角英数字に変換する。uum では使用しない
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→半角英数字の順に変換する。uum では使用しない
eg_Aa_henkan_small_loop	012	ひらがな→カタカナ→半角カタカナの順に変換する。uum では使用しない
code_convert	012	コード再変換。16 進コードに変換する。uum では使用しない
undetermined_henkan	1	未確定変換。 入力した読みを変換せずに変換後モードに入り、修正モードに入る。uum では使用しない
delete_one_kanji	0	注目文節の最後の一文字を削除する。ただし注目文節が一文字の場合は削除しない。uum では使用しない

使用例

```
; include file
include /usr/lib/locale/ja/wnn/ja/uumkey
; Commands                          Codes
unset sjis_e
atok_select_jikouho               0x20 0x9E 0x118 ^W
```

- 1 行目と 3 行目はコメントです。
- 2 行目では、標準のキー割り当て定義ファイルを取り込んでいます。
- 4 行目では、機能エントリ sjis_e のキーの割り当てを解除しています。
- 5 行目では、機能エントリ atok_select_jikouho にキーを割り当てています。

関連項目

uum(1), wnnenvutil(1), xjsi(1), uumrc(4), wnn_automaton(4), wnn_cvt_key_tbl(4),wnn_cvt_xim_tbl(4),wnn_mode(4)

使用上の留意点

henkan_on にオートマトンで発生するキーコードを割り当てることはできません。

名前 | uumrc – xjsi、uum 初期化ファイル

形式 | `/usr/lib/locale/ja/wnn/ja/uumrc`

機能説明 | `uumrc` は、Wnn6 かな漢字変換の標準インタフェースを設定するファイルで、ユーザーごとに設定が可能です。

エントリは次の形式で指定します。

エントリ 設定値 ...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して無視されます。複数回指定できないエントリが 2 回以上指定された場合は、最後の指定が有効になります。

`include` *uumrc_filename*
他の `uumrc` ファイルを読み込みます。他の `uumrc` ファイルを元にして個人の設定を行う場合などに使用します。`@DEFAULT` を指定すると、以下の優先順位で `uumrc` ファイルを読み込みます。

1. `@HOME/.Wnn6/uumrc`
2. `/etc/lib/locale/ja/wnn/ja/uumrc`
3. `/usr/lib/locale/ja/wnn/ja/uumrc` このエントリは複数回指定できます。

`setconvenv` *wnnenvrc_filename*

`setconvenv` *wnnenvrc_filename* `sticky`

`setconvenv` *jserver wnnenvrc_filename*

`setconvenv` *jserver wnnenvrc_filename* `sticky`
かな漢字変換用の環境設定ファイルを指定します。

`jserver` で、接続するかな漢字変換サーバを指定できます。かな漢字変換サーバは次の形式で指定します。

hostname	ホスト *hostname* 上で、標準のポート番号 (22273) を使用するかな漢字変換サーバー
hostname:offset	ホスト *hostname* 上で、(標準のポート番号 + *offset* をポート番号として使用するかな漢字変換サーバー
hostname/port_number	ホスト *hostname* 上で、*port_number*をポート番号として使用するかな漢字変換サーバー

`sticky` を指定すると、`xjsi/uum` の終了後も環境が保持され、 次回 `xjsi/uum` を起動する時に環境の初期化が省略され、立ち上がりが早くなります。

設定が省略された場合、`wnnenvrc` のパス名は、以下の優先順位で決定されます。

1. `@HOME/.Wnn6/wnnenvrc`
2. `/etc/lib/locale/ja/wnn/ja/wnnenvrc`
3. `/usr/lib/locale/ja/wnn/ja/wnnenvrc`

`setkankanaenv` *wnnenvrc_filename*

`setkankanaenv` *wnnenvrc_filename* `sticky`

`setkankanaenv` *server_hostname wnnenvrc_filename*

`setkankanaenv` *server_hostname wnnenvrc_filename* `sticky`
漢字かな変換用の環境設定ファイルを指定します。標準の設定では、漢字かな変換はできません。また、Wnn6 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。

`jserver` で、接続するかな漢字変換サーバーを指定できます。かな漢字変換サーバーは、`setconvenv` と同じ形式で指定します。

`sticky` を指定すると、現在の環境が `xjsi/uum` の終了後にも記憶され、次に `xjsi/uum` を起動した時に環境の初期化を行う必要がなく、立ち上がりが早くなります。

`setmaxchg` *number*
最大変換可能文字数を指定します。 0 以下の値が指定された場合は省略時値が使用されます。省略時値は 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。 0 以下の値が指定された場合は省略時値が使用されます。省略時値は 80 です。

`setmaxichirankosu` *number*
uum の候補一覧の最大表示次候補数を指定します。 0 以下の値が指定された場合は、画面の幅に応じて表示個数が決定されます。省略時値は 36 です。

`setmaxhistory` *number*
かな漢字変換の履歴を最大何個まで記憶するかを指定します。 0 以下の値が指定された場合は省略時値が使用されます。省略時値は 11 です。

`excellent_delete`
オートマトン (ローマ字かな変換) で文字を消去する場合、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 (省略時設定)。

`simple_delete`
オートマトン (ローマ字かな変換) で文字を消去する場合、 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`
uum で、tty のフロー制御をオンに設定します (省略時設定)。

`flow_control_off`
uum で、tty のフロー制御を オフに設定します。

`convkey_not_always_on`
uum で、変換オフのとき、キーコード変換を機能させません (省略時設定)。

`convkey_always_on`
uum で、変換オフのとき、キーコード変換を機能させます。

`not_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す (省略時設定)。

`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`
uum を変換モードオンで立ち上げます (省略時設定)。

`waking_up_no_henkan_mode`
uum を変換モードオフで立ち上げます。

`henkan_off_kuten`
句点 (。) を入力しても、かな漢字変換を開始しません (省略時設定)。

`henkan_on_kuten`
句点 (。) を入力すると、かな漢字変換を開始します。

`setuumkey` *uumkey_filename*
キーの割り当ての設定ファイルを指定します。

設定が省略された場合、`uumkey` のパス名は以下の優先順位で決定されます。

1. `/etc/lib/locale/ja/wnn/ja/uumkey`
2. `/usr/lib/locale/ja/wnn/ja/uumkey`

`setrkfile` *roman_character-Kana_conversion_file_name*
ローマ字かな変換で使用するモード定義表のファイル名を指定します。また、モード定義表ファイル (`mode`) が存在するディレクトリ名で指定することもできます。省略時値は `/usr/lib/locale/ja/wnn/ja/rk/mode` です。

`setconvkey` *convert–key_filename*

`setconvkey` *termtype convert–key_filename*
uum で機種 (端末) によるキー割り当ての違いを吸収するために使用するキーコード変換表のファイル名を指定します。省略時値は `/usr/lib/locale/ja/wnn/cvt_key_tbl` です。*termtype* が指定されたエントリは、その *termtype* が環境変数 `TERM` の値と一致した場合に有効になります。*termtype* の指定には、ワイルドカード (*) が使用できます。このエントリは複数回指定できます。

`setjishopath` *pathname*
uum の辞書追加機能で使用される辞書ファイル名入力領域の初期値を指定します。省略時値は空文字列です。

`sethindopath` *pathname*
uum の辞書追加機能で使用される頻度ファイル名入力領域の初期値を指定します。省略時値は空文字列です。

`setfuzokugopath` *pathname*
uum の辞書追加機能で使用される付属語ファイル名入力領域の初期値を指定します。省略時値は空文字列です。

touroku_comment
  uum で、単語登録時にコメントの入力を行います。

touroku_no_comment
  uum で、単語登録時にコメントの入力を行いません (省略時設定)。

ファイル名の指定では、以下の記法を使用できます。

~	環境変数 HOME の値
~user	*user* のホームディレクトリ
@HOME	環境変数 HOME の値
@LIBDIR	/usr/lib/locale/ja/wnn
@USR	ユーザー名
@LANG	ja

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 uum(1), wnnenvutil(1), xjsi(1), jserver(1M), uumkey(4), wnn_automaton(4), wnn_cvt_key_tbl(4), wnn_mode(4), wnnenvrc(4)

名前 | wnnenvrc – Wnn6 かな漢字変換辞書/変換パラメタ設定ファイル

形式 | /usr/lib/locale/ja/wnn/ja/wnnenvrc

機能説明 | wnnenvrc は、かな漢字変換に使用する辞書、付属語ファイル、変換パラメタの設定などを行います。

エントリは次の形式で指定します。

エントリ 設定値 ...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して無視されます。同じエントリ (setdic を除く) が 2 回以上指定された場合は、最後の指定が有効になります。辞書を設定するエントリ (setdic、set_fi_system_dic、set_fi_user_dic、muhenkan_gakusyuu、bunsetsugiri_gakusyuu) は、合計 30 個まで設定できます。

- include *wnnenvrc_filename*

  他の wnnenvrc ファイルを読み込みます。他の wnnenvrc ファイルを元にして個人の設定を行う場合などに使用します。

  ファイル名の指定では、以下の記法を使用できます。

~	環境変数 HOME の値
~user	*user* のホームディレクトリ
@HOME	環境変数 HOME の値
@LIBDIR	/usr/lib/locale/ja/wnn
@USR	ユーザー名

- setdic <辞書ファイル> <頻度ファイル> <辞書優先度> <辞書の属性> <頻度の属性> <辞書ファイルのパスワードが記述してあるファイル> <頻度ファイルのパスワードが記述してあるファイル> <変換>

  辞書について指定します。

  <辞書ファイル> <頻度ファイル>
  辞書ファイルと、対応する頻度ファイルを指定します。<頻度ファイル> にハイフン (-) を指定すると、<辞書ファイル> 内の頻度を使用します。<辞書ファイル> と <頻度ファイル> のファイル名の先頭でファイルの場所を指定できます。

!	クライアント側
:	jserver 側
hostname:	*hostname* の wnnds 側

  ファイルの場所が指定されていない場合、jserver の設定で接続する辞書引きサーバー (wnnds) が指定されていれば、指定された wnnds 側のファイルを使用します。wnnds が指定されていなければ、jserver 側のファイルを使用します。

  <辞書優先度>
  辞書優先度を 10 進数で指定します。

<辞書の属性>
辞書の属性を指定します。

1　読み取り専用。辞書ファイルを更新しません

2　テンポラリ学習。一時的に学習しますが、辞書ファイルを更新しません

3　グループ辞書。1 つの辞書を複数ユーザーで同時に使用します。辞書ファイルを更新します

4　マージ辞書。1 つの辞書を複数ユーザーで同時に使用します。辞書ファイルを更新しません

0　上記以外

<頻度の属性>
頻度の属性を指定します。

1　読み取り専用。頻度ファイルを更新しません

2　テンポラリ学習。一時的に学習しますが、頻度ファイルを更新しません

0　上記以外

<辞書ファイルのパスワードが記述してあるファイル>
辞書ファイルのパスワードが記述してあるファイル

<頻度ファイルのパスワードが記述してあるファイル>
頻度ファイルのパスワードが記述してあるファイル

<変換>
かな漢字変換の場合は 0 を、漢字かな変換の場合は 1 を指定します。

<頻度ファイル> 以降の設定値の指定を省略すると、次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> <FI 関係頻度ファイル> <FI 関係頻度の属性> <FI 関係頻度ファイルのパスワードが記述してあるファイル>

  FI 関係システム辞書について指定します。

  <FI 関係システム辞書ファイル> <FI 関係頻度ファイル>
  FI 関係システム辞書ファイルと FI 関係頻度ファイルを指定します。<FI 関係システム辞書> にハイフン (-) を指定すると、FI 関係頻度は更新されません。<FI 関係システム辞書> と <FI 関係頻度ファイル> のファイル名の先頭でファイルの場所を指定できます。

  !　クライアント側

  :　jserver 側

  hostname:　*hostname* の wnnds 側

ファイルの場所が指定されていない場合、jserver の設定で接続する辞書引きサーバー (wnnds) が指定されていれば、指定された wnnds 側のファイルを使用します。wnnds が指定されていなければ、jserver 側のファイルを使用します。

<FI 関係頻度の属性>
FI 関係頻度の属性を指定します。

1 読み取り専用。FI 関係頻度を更新しません

2 テンポラリ学習。一時的に FI 関係頻度を学習しますが、FI 関係頻度ファイルを更新しません

3 グループ辞書。1 つの FI 関係頻度ファイルを複数ユーザーで同時に使用します。FI 関係頻度ファイルを更新します

0 上記以外

<FI 関係頻度ファイルのパスワードが記述してあるファイル>
FI 関係頻度ファイルのパスワードを記述したファイルを指定します。

- set_fi_user_dic <FI 関係ユーザー辞書ファイル> <FI 関係ユーザー辞書の属性> <FI 関係ユーザー辞書ファイルのパスワードが記述してあるファイル>

  FI 関係ユーザー辞書について指定します。

  <FI 関係ユーザー辞書ファイル>
  FI 関係ユーザー辞書ファイルを指定します。ファイル名の先頭でファイルの場所を指定できます。

  ! クライアント側

  : jserver 側

  hostname: *hostname* の wnnds 側 ファイルの場所が指定されていない場合、jserver の設定で接続する辞書引きサーバー (wnnds) が指定されていれば、指定された wnnds 側のファイルを使用します。wnnds が指定されていなければ、jserver 側のファイルを使用します。

  <FI 関係ユーザー辞書の属性>
  FI 関係ユーザー辞書の属性を指定します。

  1 読み取り専用。FI 関係ユーザー辞書ファイルを更新しません

  2 テンポラリ学習。一時的に学習しますが、FI 関係ユーザー辞書ファイルを更新しません

  3 グループ辞書。1 つの辞書を FI 関係複数ユーザーで同時に使用します。FI 関係ユーザー辞書ファイルを更新します

  0 上記以外

  <FI 関係ユーザー辞書ファイルのパスワードが記述してあるファイル>
  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を指定します。

- muhenkan_gakusyuu <無変換学習辞書ファイル> <無変換学習辞書の属性> <無変換学習辞書優先度> <無変換学習辞書ファイルのパスワードが記述してあるファイル> <変換>

無変換学習について設定します。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時、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>
無変換学習辞書ファイルを指定します。ファイル名の先頭でファイルの場所を指定できます。

`!` クライアント側

`:` `jserver` 側

hostname`:` *hostname* の `wnnds` 側 ファイルの場所が指定されていない場合、`jserver` の設定で接続する辞書引きサーバー (`wnnds`) が指定されていれば、指定された `wnnds` 側のファイルを使用します。`wnnds` が指定されていなければ、`jserver` 側のファイルを使用します。

<無変換学習辞書の属性>
無変換学習辞書の属性を指定します。

`1` 読み取り専用。無変換学習辞書ファイルを更新しません

`2` テンポラリ学習。一時的に学習しますが、無変換学習辞書ファイルを更新しません

`3` グループ辞書。1 つの無変換学習辞書を複数ユーザーで同時に使用します。無変換学習辞書ファイルを更新します

`0` 上記以外

<無変換学習辞書優先度>
無変換学習辞書の優先度を 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が記述してあるファイル>
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を指定します。

<変換>
かな漢字変換の場合は `0` を、漢字かな変換の場合は `1` を指定します。

- `bunsetsugiri_gakusyuu` <文節切り学習辞書ファイル> <文節切り学習辞書の属性> <文節切り学習辞書優先度> <文節切り学習辞書ファイルのパスワードが記述してあるファイル> <変換>

文節切り学習について設定します。文節切り学習とは、確定時に文節を切り直した時、切り直した箇所の前後の 2 文節を 1 つの単語として、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>
文節切り学習辞書ファイルを指定します。ファイル名の先頭でファイルの場所を指定できます。

`!` クライアント側

`:` `jserver` 側

hostname`:` *hostname* の `wnnds` 側 ファイルの場所が指定されていない場合、`jserver` の設定で接続する辞書引きサーバー (`wnnds`) が指定されていれば、

指定された wnnds 側のファイルを使用します。wnnds が指定されていなければ、jserver 側のファイルを使用します。

<文節切り学習辞書の属性>
文節切り学習辞書の属性を指定します。

1	読み取り専用。文節切り学習辞書ファイルを更新しません
2	テンポラリ学習。一時的に学習しますが、文節切り学習辞書ファイルを更新しません
3	グループ辞書。1 つの文節切り学習辞書を複数ユーザーで同時に使用します。文節切り学習辞書ファイルを更新します
0	上記以外

<文節切り学習辞書優先度>
文節切り学習辞書の優先度を 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が記述してあるファイル>
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を指定します。

<変換>
かな漢字変換の場合は 0 を、漢字かな変換の場合は 1 を指定します。

- setfuzokugo *auxiliary_word_file*

  setgrammar *auxiliary_word_file*

  付属語ファイルを指定します。

- setparam param0. . . param9 hindo0. . . hindo6

  変換パラメタと、疑似品詞の頻度を指定します。( ) 内は初期設定値です。

param0	N (大文字) 文節解析の N (5)
param1	大文節中の小文節の最大数 (10)
param2	自立語の頻度に対する係数 (2)
param3	小文節長に対する係数 (45)
param4	自立語長に対する係数 (0)
param5	直前に使ったことを示すビットに対する係数 (80)
param6	辞書に対する係数 (5)
param7	小文節の評価値に対する係数 (1)
param8	大文節長に対する係数 (20)
param9	小文節数に対する係数 (0)
hindo0	疑似品詞「数字」の頻度 (400)
hindo1	疑似品詞「カナ」の頻度 (–100)
hindo2	疑似品詞「英数」の頻度 (400)

hindo3 疑似品詞「記号」の頻度 (80)

hindo4 疑似品詞「閉括弧」の頻度 (200)

hindo5 疑似品詞「付属語」の頻度 (2)

hindo6 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ファイルまたは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成するかどうかの確認をします。

- `confirm1`

  このエントリ以降で指定された辞書ファイルまたは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成するかどうかの確認を 1 度だけ行い、それ以降はその指定に従います。

- `create_without_confirm`

  このエントリ以降で指定された辞書ファイルまたは頻度ファイルが存在しない場合、無条件に新しく作成します (初期設定)。

- `no_create`

  このエントリ以降で指定された辞書ファイルまたは頻度ファイルが存在しない場合、新しく作成しません。

- `saisyu_siyou TRUE|FALSE`

  最終使用最優先処理を行うかどうかを指定します。最終使用最優先処理は、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 TRUE|FALSE`

  複合語優先変換を行うかどうかを指定します。複合語優先変換は付属語を含まない候補を優先する (たとえば、「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 REGULATION|YES|NO`

  送り基準処理を指定します。送り基準処理は、変換の候補が送りがなだけ異なる (たとえば、「行う」と「行なう」) 場合に、指定された送り基準に一致する候補を優先する処理です。本則候補を優先する場合は `REGULATION`、送りがなが長い候補を優先する場合は `YES`、送りがなが短い候補を優先する場合は `NO` を指定します。

- `settou_kouho HIRAGANA|KANJI`

  接頭語初期候補を指定します。ひらがなの候補を出す場合は `HIRAGANA`、漢字の候補を出す場合は `KANJI` を指定します。

- `rendaku TRUE|FALSE`

  連濁処理を行うかどうかを指定します。連濁処理は、連濁する候補 (たとえば、会社「がいしゃ」) を最初に変換に使用しない処理です。連濁処理を行う場合は `TRUE`、行わない場合は `FALSE` を指定します。

- yuragi TRUE|FALSE

  長音・ゆらぎ処理を行うかどうかを指定します。長音・ゆらぎ処理は、長音・ゆらぎを含む読み (たとえば、「はなじ」を「鼻血」、「 ほーほー」を「方法」、「こうり」を「氷」) 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 TRUE|FALSE

  送り基準学習を行うかどうかを指定します。送り基準学習は、直前に確定した送り基準を学習し、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 TRUE|FALSE

  接頭語学習を行うかどうかを指定します。 接頭語学習は、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 TRUE|FALSE

  接尾語学習を行うかどうかを指定します。接尾語学習は、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 TRUE|FALSE

  一般語学習を行うかどうかを指定します。一般語学習は、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 NOT|LOW|NORMAL|HIGH|ALWAYS

  頻度学習の度合いを指定します。学習しない場合は NOT、じわじわ学習にする場合は LOW、普通学習にする場合は NORMAL、すぐ学習にする場合は HIGH、必ず学習する場合は ALWAYS を指定します。

- fi_hindo_kakuritu NOT|LOW|NORMAL|HIGH|ALWAYS

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

- use_hinsi *hinsi. . .*

  変換に使用する品詞の一覧を指定します。

- unuse_hinsi *hinsi. . .*

  変換に使用しない品詞の一覧を指定します。

- giji_number HAN|ZEN|HANCAN|ZENCAN|KAN|KANSUUJI|KANOLD

  数字の変換初期候補を指定します。コンマなしの半角数字にする場合は HAN、コンマなしの全角数字にする場合は ZEN、コンマありの半角数字にする場合は HANCAN、コンマありの全角数字にする場合は ZENCAN、位なしの漢数字 (一、二、三) にする場合は KAN、位ありの漢数字 (一、二、三) にする場合は KANSUUJI、位ありの旧漢数字 (壱、弐、参) にする場合は KANOLD を指定しま

す。

- giji_eisuu HAN|ZEN

  アルファベットの変換初期候補を指定します。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 HAN|JIS|ASC

  記号の変換初期候補を指定します。半角にする場合は HAN、全角にする場合は JIS、ASCII 候補にする場合は ASC を指定します。

- kutouten TRUE|FALSE

  句読点入力モードを指定します。「.」「,」をそれぞれ「。」「、」に変換する場合は TRUE、「. 」「, 」 に変換する場合は FALSE を指定します。

- kakko TRUE|FALSE

  括弧入力モードを指定します。[ ] を「 」に変換する場合は TRUE、[ ] に変換する場合は FALSE を指定します。

- kigou TRUE|FALSE

  斜線入力モードを指定します。「/」を「・」に変換する場合は TRUE、「／」に変換する場合は FALSE を指定します。

- autosave *number*

  学習情報の自動保存を行う確定操作の回数を指定します。指定された回確定操作を行うと、学習情報は自動的に保存されます。省略時値は 50 です。

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

```
confirm1
setfuzokugo iwanami/kougo.fzk
set_fi_system_dic iwanami/fisd    usr/@USR/fisd.h            0 -
set_fi_user_dic            usr/@USR/fiud            0 -
setdic iwanami/kihon.dic        usr/@USR/kihon.h    6 1 0 - - 0
setdic iwanami/symbol.dic    usr/@USR/symbol.h    1 1 0 - - 0
setdic iwanami/tankan.dic        -        1 1 1 - - 0
setdic iwanami/tankan2.dic        -        1 1 1 - - 0
setdic iwanami/tankan3.dic        -        1 1 1 - - 0
setdic iwanami/tel.dic            -        1 1 1 - - 0
setdic iwanami/zip.dic            -        1 1 1 - - 0
setdic iwanami/ikeiji.dic            -        1 1 0 - - 0
setdic usr/@USR/ud            -        15 0 0 - - 0
muhenkan_gakusyuu        usr/@USR/muhenkan    0 15 - 0
bunsetsugiri_gakusyuu        usr/@USR/bunsetsu    0 15 - 0
okuri_kijun REGULATION
settou_kouho    KANJI
setubi_gakusyu    TRUE
hanyou_gakusyu    TRUE
hindo_kakuritu    NORMAL
giji_number HAN
unuse_hinsi 単漢字  郵便番号    電話番号
;;      N nsho hindo len jiri flag jisho sbn dbn_len sbn_cnt
```

例 1　（続き）

```
suuji kana eisuu kigou toji_kakko fuzokogo kaikakko
setparam 5 10   2   45  0    80   5     1   20    0    400
-100   400   80    200    2    200
```

関連項目　uum(1), xjsi(1), jserver(1M), uumrc(4)

名前 wnnhosts – 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

形式 /etc/lib/locale/ja/wnn/wnnhosts

機能説明 wnnhosts は Wnn6 かな漢字変換サーバー (jserver) を利用できるユーザー、および Wnn6 辞書引きサーバー (wnnds) に接続できるかな漢字変換サーバーを指定します。

アクセス制御ファイルの書式は次のとおりです。"{" の前には空白を入れます。

```
jserver        ja     <かな漢字変換サーバー>     {
<アクセス許可データー>
    :
}
wnnds        ja     <辞書引きサーバー>     {
<アクセスデーター>
    :
}
```

<かな漢字変換サーバー> は次の書式で指定します。

hostname	ホスト *hostname* 上で、標準のポート番号 (22273) を使用するかな漢字変換サーバー
hostname/port_no	ホスト *hostname* 上で、*port_no* をポート番号として使用するかな漢字変換サーバー

<辞書引きサーバー> は次の書式で指定します。

hostname	ホスト *hostname* 上で、標準のポート番号 (26208) を使用する辞書引きサーバー
hostname/port_no	ホスト *hostname* 上で、*port_no* をポート番号として使用する辞書引きサーバー

jserver に対する <アクセス制御データー> は次の書式で指定します。

<ホスト名>	このホストの全ユーザーが利用可能となります。
<ホスト名>:<ユーザー名リスト>	<ユーザー名リスト> は <ユーザー名> を並べ、"," で区切ったものです。このホストの、リストで指定されたユーザーが利用可能となります。
@<ユーザー名>	このユーザーはすべてのホストから利用可能となります。

wnnds に対する <アクセス制御データー> は次の書式で指定します。

<ホスト名>	このホスト上の jserver が接続できます。

jserver および wnnds は、ホスト名とポート番号が一致するアクセス制御情報を使用します。

";" で始まる行は、コメントとして扱われます。

使用例

例 1

```
jserver ja_JP hostA {
;hostC:usr1,usr2,usr3
hostA:usr1,usr4
hostB
hostC:usr5
@usrA
;usrB
}
wnnds ja_JP hostA {
hostA
hostD
}

jserver ja_JP hostA/22273 {
hostB
hostE
@usrA
}

wnnds ja_JP hostA/22385 {
hostA
hostD
}
```

関連項目

jserver(1M), wnnaccess(1M), wnnds(1M)

名前 wnn_2A_CTRL – 入力変換モード切り替え定義表

形式 /usr/lib/locale/ja/wnn/ja/rk/2A_CTRL

機能説明 2A_CTRL は xjsi(1) とuum(1) の入力変換モードを切り替えるキーを設定します。

2A_CTRL 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。

設定例 '\x81' (switch katakana) ;PF1 key

'\x82' (switch zenkaku) ;PF2 key

'\x83' (switch romkan) ;PF3 key

関連項目 uum(1), xjsi(1), uumkey(4), wnn_automaton(4), wnn_cvt_key_tbl(4), wnn_cvt_xim_tbl(4),wnn_mode(4)

注意事項 2A_CTRL は cvt_xim_tbl または cvt_key_tbl で処理されたコードに対して適用されます。

wnn_2B_ROMKANA(4)

名前 | wnn_2B_ROMKANA – ローマ字かな変換定義表

形式 | /usr/lib/locale/ja/wnn/ja/rk/2B_ROMKANA

機能説明 | 2B_ROMKANA は xjsi(1) と uum(1) のローマ字かな変換の規則を設定します。

2B_ROMKANA 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 | uum(1), xjsi(1), wnn_automaton(4), wnn_mode(4)

名前 | wnn_automaton – オートマトン

機能説明 | オートマトンは、xjsi やuum のローマ字かな変換を実現している機能です。変換内容を設定した表 (変換表) を変えることで 、任意の変換に変更することができます。

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 つの処理では、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。

変換表はテキストファイルであるため、容易に任意の変換表に変更できます。また、1 つの変換を行なったあと次の変換を行うまでは、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。

xjsi 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

    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。

xjsi は、モード定義表を以下の順番で探します。

1. xjsi 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/locale/ja/wnn/ja/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、モード表示文字列を記述します。

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. 特殊パス表記

特殊パス表記としては、以下に示すものがあります。

@HOME	環境変数 HOME
@MODEDIR	モード定義表の存在するディレクトリ
@LIBDIR	/usr/lib/locale/ja/wnn/
~user	*user* がユーザー名であれば、そのユーザーのホームディレクトリ名
~	自分のホームディレクトリ名

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* [on|off] [r|nr]	*mode_name* は、英数字を指定します。[on|off] は初期状態を表します。デフォルトは off です。[r|nr] はローマ字かな変換モードをオートマトンに認識させるためのフラグです。r が設定されているモードは、F6 キーなどでひらがな変換を行う際に、その時のモードに関わらず使用されます。デフォルトは nr 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、指定された *directory* を検索します。*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...* 対応表を検索するディレクトリパスを無効にし、引数に指定された *directory* を検索するように設定します。 *directory* は、空白で区切って複数指定できます。 `path` 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。

xjsi では、この文字列がモード表示に使用されます。

(2)、(3) は、条件によって選択する対応表を変えるときに使用します。(2) の if 文 は、その条件式の結果が真であれば、if 文内の指定を参照し、if 文の次の指定は参照しません。条件式が偽であれば、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であれば、when 文内の指定を参照し、偽であれば参照しません。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) があり、双方のモードが on のとき、(and kana romajikana) は真になります。

ここで、条件式を○◎●で表し、変換表の名前を ABC... で表すという規則において、たとえば次のように書かれていたとします。

(when ○ A (if ◎ B) C) (if ● D) E

この条件式○◎●が、すべて成り立っているとします。この並びを最初から見ていくと、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の 「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、xjsi が使用するモード定義表を例に挙げます。

このモード定義表では、3 のモードが定義されています。そのあとの 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 文は参照しません。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、条件式が偽になります。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になっています。そのため、前処理、後処理の対応表には次の制限があります。

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) のうちのいずれか、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。

出力コードを省略した場合やバッファー残りを省略した場合は、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、「評価すると文字になる式」または「評価すると文字列になる式」とは、その式によって文字あるいは文字列に置き換わる式を指します。

「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　次のような文字表記になります (「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 の文字です。「^?」は DEL コードを表します

'\文字'　文字は、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文字を表します。「\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... ...'　　指定された 8 進コードに相当する文字を表します。

'\o8 進コード... ...'　　指定された 8 進コードに相当する文字を表します。

'\d10進コード... ...'　　指定された 10 進コードに相当する文字を表します。

'\x16進コード... ...'　　指定された 16 進コードに相当する文字を表します。

(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 英小文字であれば大文字に変える 例: (toupper a) → A
tolower	引数が ASCII 英大文字であれば小文字に変える 例: (tolower A) → a
toupdown	引数が ASCII 英小 (大) 文字であれば大 (小) 文字に変える 例: (toupdown a) → A (toupdown A) → a
tozenalpha	引数が ASCII 文字であれ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であればひらがなに変える 例: (tohira ア) → あ
tokata	引数がひらがなであれ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であれ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であれ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 '\d256') → 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。

以下に、評価すると文字列になる式を示します。

(1) "文字表記. . ."　次のような文字表記になります (「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. . . [;]　指定された 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 (;) を指定します。

\o8 進コード. . . [;]　指定された 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 (;) を指定します。

\d10 進コード. . . [;]　指定された 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 (;) を指定します。

\x16進コード. . . [;]　指定された 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 (;) を指定します。 " " 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であれ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 の場合、(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 の場合、(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

関数名	機能
alloff	すべてのモードを off にする

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))
```

機能

次の 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場合、エラーとみなします。

(restart) 前回のモード定義表をあらた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。

(defvar 変数表記 (between 文字表記 1 文字表記 2)) between は、コード順に並べた時に、文字表記 1 と文字表記 2 の間に含まれる文字 (両端を含む) 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を定義します。変数は、それを使用する表の中で宣言します。

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。

変数の定義は、その表全体で有効になります。1 つの表の中で、同名の変数を 2 度 defvar で宣言することはできません。

1 つの表で a1 という変数を定義し、別の表でまた a1という変数を別仕様で定義することができます。この 2 つの a1 という変数は別々に処理されます。

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

変数宣言で行われた変域の文字について処理されます。

(between A E) は (list A B C D E) と同じです。

変数に関する注意事項

使用する変数はその表の中において変数宣言で定義してください。

1 つの表で a1 という変数を定義し、別の表で 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。変数の定義はその表の中全体で有効です。1 つの表の中で同名の変数を 2 度変数宣言することはできません。

対応表の 1 行の中では、同名の変数は常に同じ値をもちます。

(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。

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た場合、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。

この場合、次のように変更すれば期待する動作を得ることができます。

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

「aA」「bB」を入力すると「3」に変換します。

(defvar a1 (list A B)) (a1)(toupper (a1)) 3

「Aa」「Bb」を入力すると「 3」に変換します。

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た (つまり、入力コードとの 照合で何らかの値を代入された) 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誤設定になります。

対応表による変換方式

前処理　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。

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で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当たるコードが出力されます。

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　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、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 コード部分に、先頭から何文字かは一致している状態) があれば、変換未確定となり、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。

その他、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すると (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ます。バッファー残りの部分があれば、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。合致したけれども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、常に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が再度読み込まれます。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、オートマトンを終了せずに読み込み直すことができます。

後処理　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。

以下の例では、「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 `cvt_key_tbl` は、`terminfo` のエントリとキーコードの変換表を定義します。uum(1) は `terminfo` と `cvt_key_tbl` を使用して、入力された文字列をキーコードに変換します。`terminfo` で定義されたエスケープシーケンスの各文字が一秒以上間を開けて入力された場合、uum はそれぞれを独立した文字として処理します。

書式 *terminfo_entry* *code*

terminfo_entry と、*code* の間には空白文字が必要です。セミコロン (;) で始まる行はコメントになります。

terminto_entry

変換の対象となる `terminfo` のエントリは次の通りです。

kf0	kf1	kf2	kf3	kf4	kf5	kf6
kf7	kf8	kf9	kf10	kf11	kf12	kf13
kf14	kf15	kf16	kf17	kf18	kf19	kf20
kf21	kf22	kf23	kf24	kf25	kf26	kf27
kf28	kf29	kf30	kf31	kbs	ktbc	kclr
kctab	kdch1	kdl1	kcud1	krmir	kel	ked
khome	kich1	kil1	kcub1	kll	knp	kpp
kcuf1	kind	kri	khts	kcuu1		

code

空白文字、バックスラッシュ (\)、キャレット (^) を除く 1 文字　文字そのものを表します。

^*character*　*character* は、`@`、`A (a)`、`B (b)`、`C (c)`、`D (d)`、`E (e)`、`F (f)`、`. . .`、`Z (z)`、`[`、`\`、`]`、`^`、`_` であり、^@ はコントロール + スペース (0x00)、^A はコントロール + A (0x01)、. . .、^_ はコントロール + _ (0x1f) を表します。

\8進数、\o8進数、\d10進数、\x16進数	文字コードを直接記述します。
\n、\t、\b、\r、\f、\e、\E	\n は改行 (NEWLINE) 、\t はタブ (TAB) 、\b はバックスペース (BACKSPACE) 、\r はリターン (RETURN)、\f はフォームフィード (FORMFEED) 、\e はエスケープ (ESC) 、\E はエスケープ (ESC) を表します。
character	*character* は、0 から 7 までの数字、o、d、x、n、t、b、r、f、e、E 以外の文字。\ 自身を表す場合は、「\\」 とします。

使用例

```
kf1         \x81
kf2         \x82
kf3         \x83
kf4         \x84
kcud1         \x92
kcub1         \x91
kcuf1         \x90
kcuu1         \x93
```

関連項目 uum(1), uumkey(4), wnn_2A_CTRL(4)

注意事項 この表によって変換されたコードは、次にローマ字かな変換オートマトンの表 2A_CTRL (デフォルト) によって評価され、さらに uumkey によって評価されます。

名前 | wnn_cvt_xim_tbl – xjsi 用キー変換テーブル

形式 | `/usr/lib/locale/ja/wnn/cvt_xim_tbl`

機能説明 | `cvt_xim_tbl` はキーボードからの入力とキーコードの変換表を定義します。`xjsi`(1) は `cvt_xim_tbl` を使用してキーボードからの入力 (*KeySym*) をキーコード (*Wnn code*) へ変換します。

書式 | *State-or-KeySym*
Wnn_code

State-or-KeySym
= [*States*
]*KeySym-name*

States = *State-or-KeySym* と *Wnn_code* は「空白文字」または「タブ文字」で区切り *State-name* ます。セミコロン (`;`) で始まる行はコメント行です。
| [*States*]

コードの記述法 | 8 進数　`0??`

10 進数　`??`

16 進数　`0x??` または `0X??`

使用例 |
```
Meta|Left                    0x9A
Meta|Up                    0x99
Meta|F11                    0x95
Meta|minus                0x81
Meta|asciicircum    0x82
Kanji                        0x81
F1                            0x91
F2                            0x90
Meta|Shift|F1            0x91
```

関連項目 | `xjsi`(1), `uumkey`(4), `wnn_2A_CTRL`(4)

注意事項 | `cvt_xim_tbl` により変換されたコードは、次にオートマトンの表である `2A_CTRL` (デフォルト) によって評価され、そのあと `uumkey` によって評価されます。

名前 | wnn_hinsi.data – Wnn6 品詞管理ファイル

形式 | /usr/lib/locale/ja/wnn/ja/hinsi.data

機能説明 | hinsi.dataは、幹語品詞に関する情報を管理するためのファイルです。

hinsi.data に定義されている順番で、品詞、複合品詞に番号が割り当てられます。番号は、辞書、品詞ファイルを作成する場合、サーバ -、クライアントで品詞名を検索する場合、複合品詞に対してそれを構成する品詞の集合を検索する場合に用いられます。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行える操作は、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、@ だけからなる行を品詞、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、品詞名は定めないが、あらかじめ領域を確保し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
```

複合品詞の定義で現れる品詞は、それよりも前に品詞として定義する必要があります。また、同じ名前を持つ品詞、または複合品詞が存在してはいけません。

セミコロン (;) から行末までは、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検索する、または、複合品詞に対してその構成要素を検索する) は、ライブラリが提供するので、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            ;これは名詞です。
一段
一段名$一段:名詞      ;これは複合品詞
@
@
```

注意事項 | 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ければならないため、このファイルを編集しないでください。変更を行なった場合、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のでご注意ください。

名前 | wnn_mode – モード定義表

形式 | /usr/lib/locale/ja/wnn/ja/rk/mode

機能説明 | mode は xjsi(1) と uum(1) の入力変換モードを定義し、各入力変換モードで使用する対応表の組み合わせとその適用規則を設定します。mode はオートマトンの変換表の「目次」に相当します。

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 | uum(1), uumrc(4), xjsi(1), 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 | `serverdefs` は、クライアントと Wnn6 かな漢字変換サーバーの接続環境を設定します。Wnn6 のライブラリは、このファイルを参照して変換サーバーと接続します。セミコロン (`;`) で始まる行はコメントです。

書式 | <言語名> <サーバーのホスト名> <UNIX ドメインのソケット名> <サービス名> <ポート番号> <環境変数名>

<言語名> 言語を指定します。`xjsi` と `uum` は "ja" の行を参照します。

<サーバーのホスト名> 変換サーバーのホスト名を指定します。

<UNIX ドメインのソケット名> UNIX ドメインのソケットで接続する場合のソケットの終端を指定します。

<サービス名> inet ドメインのソケットで接続する場合の TCP のサービス名を指定します。

<ポート番号> inet ドメインのソケットで接続する場合の TCP のポート番号を指定します。サービス名からポート番号を検索できなかった場合にこのポート番号が使用されます。このポート番号は省略可能です。

<環境変数名> 変換サーバーのホスト名が設定される環境変数名を指定します。

使用例 | **例 1** wnn_serverdefs の例

```
ja  jserver /tmp/jd_sockV6   wnn6     22273    JSERVER
```

関連項目 | `uum`(1), `xjsi`(1), `jserver`(1M)

名前 wnn_ximrc – xjsi 用環境設定ファイル

形式 /usr/lib/locale/ja/wnn/ximrc

機能説明 ximrc ファイルは、xjsi の環境を設定するファイルです。ユーザーごとに異なる設定が可能です。エントリは次の形式で指定します。

<エントリ> <設定値> ...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して無視されます。同じエントリが 2 回以上指定された場合は、最後の指定が有効になります。設定値が指定されない場合は、省略時値が使用されます。

setuumrc *language uumrc*	言語ごとに、xjsi が参照する uumrc ファイルを指定します。Solaris の日本語環境では、 *language* には ja を指定します。uumrc のデフォルトは @LIBDIR/@LANG/uumrc です。
preloadrkfile *language_name*	起動時にオートマトンテーブルをロードする言語を指定します。xjsi はこのエントリを無視します。
setbackspacechar *backspace_char*	バックスペース文字を指定します。省略時値は 0x7f です。
setposition *location*	候補一覧ウィンドウを表示する位置を指定します。 over 入力操作を行うウィンドウの下 spot 入力した文字が挿入される位置 center 画面の中央 デフォルトは over です。
setlayout *format*	候補一覧ウィンドウの形式を指定します。 multi 縦横に候補を表示 vert 縦一列で候補を表示 horiz 横一列で候補を表示 ATOK 風入力スタイルの省略時値は horiz です。ATOK 風入力スタイル以外の入力スタイルの省略時値は multi です。ATOK 風入力スタイルで multi を指定すると、horiz になります。デフォルトは multi です。

使用例 **例 1**

```
;;    sample    ximrc
setposition            spot
setlayout            vert
```

関連項目 | `xjsi`(1)