ロ

ローマ字カタカナ入力モード 13
ローマ字・かな対応表 12
ローマ字かな変換 15, 37, 39
ローマ字ひらがな入力モード 13

フ

フック 13, 30, 36

ブ

文体に依存しないかな漢字変換 2

プ

プログラム実行変換 25

ヘ

変換 20
変換開始点 16
変換モード 15

マ

マージ 57

メ

メールによる入手方法 81
メイリングリスト 79
メニューによる入力 23

モ

モード 13
モードライン 13
文字コードを知る方法 30

ユ

ユーザ辞書 42
ユーザ辞書ファイル 13, 43, 2, 9
ユーザ辞書と共用辞書によるかな漢字変換 2
ユーティリティプログラム 8, 53

ヨ

読みのコンプリーション 24

全英文字の入力 19
全角英字入力モード 13

ソ

その他 (辞書サーバについて) 51
その他の入手方法 82

タ

単語の削除 18

ダ

第 2 水準の漢字 8

チ

チュートリアル 12

テ

テスト方法 51

ト

登録 27

ニ

入力モード 13

ネ

ネットワーク 47

バ

バッファ・ローカル 30

パ

パッケージの確認 5

ヒ

日付の入力 25, 43
平仮名確定入力モード 15

ゲ

限界・改良の余地 52

コ

コードによる入力 23
コードまたはメニューによる入力 23
候補 16, 25

サ

サーチ・パターン 29
サーバ 47
再帰的辞書登録 3
差分 58

シ

終了 12
将棋 27

ジ

辞書 9, 16, 20, 26, 50, 53, 56
辞書サーバ 2, 10, 47
辞書登録 17, 22, 25
辞書バッファ 24, 54
辞書バッファのセーブ 56
辞書ユーティリティプログラム 56
自動右揃え 13, 28
準備 9

ス

スーパーユーザの権限を持たない人のインストール方法 49
スーパーユーザの権限を持つ人のインストール方法 47
数をパラメータとする表現の入力 26

ゼ

全英モード 36

イ

インクリメンタル・サーチ 29
インストール 7

エ

英字変換 19
エコーバック 40

オ

送り仮名 19, 21, 42
送り仮名のある辞書 13
送り仮名のある単語の入力 19
送り仮名の自動処理 21
送り仮名のない辞書 13

カ

確定 17, 18, 20, 42
カスタム化 13, 33
カタカナの入力 19
かなの小文字 14
かなモード 33

キ

キー 34
起動 12
起動と終了 12
棋譜 27
旧バージョンとの相違点 63
共用辞書 2, 9

ク

クライアント 47, 52
訓令式 37

SKK5.30 との相違点 65
skkfep 83, 87
skkserv 47
skkserv が動作する環境 51
skkserv とは 47
skkserv のインストール方法 47
skkserv の起動 49
SKK 形式の辞書 50
SKK コマンド一覧 67
SKK 辞書 53
SKK 辞書の形式 53
SKK チュートリアル 12
SKK とは 1
SKK のインストールの方法 5
SKK の使用法 9
SKK の設計方針 1
SKK の特徴 2
SKK メイリングリスト 79
SKK メッセージ一覧 73
SKK モード 13
SKK 関連ソフトウェア 83

V

VIP 41

▽

▽モード 15

▼

▼モード 16

ア

アスキー入力モード 13

M

Mule 1

N

Nemacs 1, 56
Nemacs との高い親和性 2

O

overwrite モード 13

P

Perl によるユーティリティプログラム 58
picture モード 13

S

skk-check-jisyo 57
skk-diff 58
skk-echo 40
skk-input-vector 33
skk-jisyo-code 43
skk-kakutei-early 42
skk-keep-record 40
skk-merge 57
skk-prefix-list 37
skk-process-okuri-early 41
skk-record 12
skk-rom-kana-rule-list 38
skk-roma-kana-[aiueo] 37
skk-romkan.tex 12
skk-use-vip 41
skk-zenkaku-vector 33
SKK2.25 との相違点 63
SKK3.47 との相違点 63
SKK4.33 との相違点 64

付録 G

事項索引

.

.emacs 10, 33

.emacs の設定 10

.skk 11, 27, 33, 35, 39

.skk-jisyo 13, 54

.skk-record 12, 40

.skk.elc 33

.skkl 30

.vip 41

A

Anonymous Ftp による入手方法 82

ari 83

D

Demacs 1, 10, 11

E

egg 42

Emacs 10, 13, 56

Emacs Lisp ユーティリティプログラム 56

K

kakasi 83, 86

skk-search-prog-list 45, 54
skk-serv 11, 44
skk-server-host 11, 44
skk-server-version 71
skk-tut-file 12
skk-tutorial 11
skk-use-vip 41, 63
skk-version 71
skk-zenei-mode-string 43
skk-zenkaku-vector 36

skk-date-ad 25, 43
skk-diff 11, 58
skk-echo 40
skk-egg-like-newline 42
skk-file 12
skk-hirakana-mode-string 43
skk-initial-search-jisyo 46, 54
skk-input-vector 34
skk-isearch-backward 11
skk-isearch-forward 11
skk-isearch-start-mode 29
skk-isearch-use-previous-mode 29
skk-jisyo 42, 54
skk-kakutei-early 64
skk-kakutei-jisyo 45, 54
skk-kakutei-key 45
skk-katakana-mode-string 43
skk-keep-record 40
skk-kill-emacs-without-saving-jisyo 56, 70
skk-large-jisyo 11, 46, 54
skk-merge 8, 11, 56, 58
skk-mode 11, 14, 28, 33, 71
skk-mode-hook 33
skk-mode-string 43
skk-num-type-list 44
skk-number-style 25, 43
skk-numeric-convsersion 44
skk-prefix-list 37
skk-process-okuri-early 42
skk-record-file 43
skk-rom-kana-rule-list 38
skk-roma-kana-a 37
skk-save-jisyo 56, 71

j-mode-off 69
j-mode-off-in-minibuff 69
j-newline 69
j-okuri-search 55
j-previous-candidate 69
j-purge-from-jisyo 69
j-search-jisyo-file 55
j-search-kakutei-jisyo-file 55
j-search-server 55
j-self-insert 69
j-set-henkan-point 69
j-set-henkan-point-subr 69
j-start-henkan 70
j-today 70
j-toggle-kana 70
j-try-completion 70
j-zenkaku-eiji 70
j-zenkaku-henkan 70
j-zenkaku-region 70
j-zenkaku-self-insert 70

K

kanji-fileio-code 43, 64

R

register-to-point 10

S

skk-auto-fill-mode 11, 28, 70
skk-auto-okuri-process 21, 42
skk-aux-large-jisyo 11, 46, 54
skk-backup-jisyo 45
skk-check-jisyo 11, 57
skk-dabbrev-like-completion 45

付録 F

変数および関数索引

J

j-abbrev-comma 67
j-abbrev-input 67
j-abbrev-period 67
j-ascii-ignore-exp 30
j-delete-bakward-char 67
j-display-code-for-char-at-point 67
j-hiragana-henkan 68
j-hiragana-region 68
j-ignore-exp 30
j-input-by-code-or-menu 68
j-insert-a 68
j-insert-comma 68
j-insert-e 68
j-insert-i 68
j-insert-period 68
j-insert-u 68
j-kakutei 68
j-kana-input 68
j-katakana-henkan 68
j-katakana-region 69
j-keyboard-quit 69

[4] 最新バージョン, release の日付

version 0.60 March, 1993

[5] 開発者

伊藤 彰則，渡辺 雅俊，金子 裕

[6] 配布条件

なし

[7] 入手方法

goku.ecip.tohoku.ac.jp から FTP で最新版が入手できる． fj.sources にも流れている.

[8] bug report, 質問等のためのアドレス

aito@ecip.tohoku.ac.jp

[9] その他

本物の SKK に近づくようにがんばっています.

[6] 配布条件

SKK と同じように、附属辞書を含めて GNU Public Licence にしたがいます。

[7] 入手方法

fj.sources をアーカイブしているサイトから入手できます。また、 anonymous uucp では tinygw (+81-298-58-1364 ID: anon) から入手できます。

[8] bug report, 質問等のためのアドレス

```
Domestic: takahasi@tiny.or.jp
Foreign:  takahasi@sanyo-tsukuba.rwcp.or.jp
```

[9] その他

SKK の配布を追って、 1993 年版を出す予定です。

E.3 skkfep

[1] 正式名称

skkfep

[2] どのようなソフトウェアか

SKK と似た入力方式の，漢字入力フロントエンドプロセッサです.

[3] 使用するために要求される環境

UNIX; 現在サポートされているのは,

```
AIX/ESA, DOMAIN-IX SR9.x, HP-UX 8.x,
SEIUX 3.x, EWS-UX/V Rel4.0, NEWS-OS 4.x, NEWS-OS 3.3,
SunOS 4.x (AIX3.x に移植中)
```

[9] その他

発表以降 bug report が全然来ないのですが、これは安定しているからなのか、それともあんまり使われていないからなのか、どっちなんでしょう？

一年以上 update してませんが、もし行なうとすれば、今後のバージョンアップは version 1.00 に対して行なう予定ですので、skksrch.c の version 1.00 はとっておいて下さい。

E.2 kakasi

[1] 正式名称, 略称, 名前の由来

正式名称: kakasi

Kanji Kana Simple Inverter の略としていますが、skk をさかさにしたわけですね。

[2] どのようなソフトウェアかを簡単に

漢字を含む文章を変換して、ひらがな, かたかな、ローマ字等に変換します。フィルターとして機能します。また文字種の変換やコーディング方式の変換等も同時に可能です。基本辞書は SKK 辞書ですが、ほとんどのパブリックな「かな漢字」用の辞書が利用できます。

[3] 使用するために要求される環境

UNIX で C 言語がコンパイルできれば使用できます。また MSDOS では、漢字変換の部分だけに特化したバージョン karasu が利用できます。

[4] 最新バージョン, release の日付

```
version: 2.2.3
最終日付: 1992 年 8 月 6 日
```

[5] 開発者

高橋 裕信 (Hironobu TAKAHASHI)

のみ version 1.00a、その他のドキュメント、ソースは version 1.00 が最新です。なお README1.00a は、 May 21 1992 改定となってます。

[5] 開発者

ベースとなった ae の著者は Anthony Howe 氏 (ae は「Anthony's Editor」でもあります)、その他の拡張部分は、二村 祐地です。

[6] 配布条件

Free Software Foundation が公表した GNU General Public License （一般公有使用許諾） version 1 あるいはそれ以降の各バージョンの中からいずれかを選択し、そのバージョンが定める条項に従って本プログラムを再頒布または変更することができます。

[7] 入手方法

(1) USENET の comp.os.minix、 JUNET の fj.os.minix、 fj.sources の保存記事を漁る。

(2) 日経 MIX の MINIX 会議、 NIFTY の UNIX フォーラムなどから down load する。

```
日経 MIX … minix/2fixes/16,122,123,140
NIFTY   … FUNIX DL5(Free software MINIX source) 92,84,72,38,12
```

(3) 書籍「MINIX ver. 1.5 フリーソフトウェア集」(ASCII 出版) 付属のアーカイブディスクからコピーする。

※同ディスク内にはコピーフリーでないソフトウェアも入っていますので、その扱いはご注意下さい。

(4) 奇特な anonymous ftp/uucp サイトでは、保存されているかも知れません。:-)

[8] bug report, 質問等のためのアドレス

nimura@isl.melco.co.jp または Yuji.Nimura@ecs.isl.melco.co.jp

[3] 使用するために要求される環境

(1) OS

UNIX version 7 程度のファイル入出力および端末制御、プロセス制御機能がある OS。MINIX をターゲットとして開発しましたが、大抵の UNIX 風 OS なら動作するでしょう。bsd 系ばかりでなく sysV 系もサポートしています。また、サポートはしてませんが、MS-DOS など他の OS への移植性もある程度考慮しています。

(2) コンソール / 端末

JIS コード、EUC、SHIFT-JIS コードのいずれかのコードで日本語が表示できること。またデフォルトでは以下のコードで端末制御ができることが要求されます。MINIX のコンソール (minix) など、vt100 系 (ANSI?) なら問題はないはずです。

カーソルを指定した行、カラムへ … `cm=\E[%i%d;%dH`
カーソルを左上にして画面クリア … `cl=\E[H\E0J`
カーソルから行の最後までクリア … `ce=\E[K`

(3) 仮名漢字変換辞書

SKK に含まれる辞書ファイル、または同じフォーマットの辞書ファイルを別途入手してください。辞書ファイルがなくとも、仮名入力とコードによる漢字入力 (JIS コードまたは EUC) だけは可能です。

(4) インストール

ソースを展開するためのツールと、C コンパイラ、そして make コマンドが必要です。ソース・ドキュメントの総量は 200KB 程度、ソースの展開とのコンパイルには 400 〜 500KB ぐらいの作業領域があれば十分です (MINIX 1.5 for IBM/PC の場合)。

[4] 最新バージョン, release の日付

最新版は、1.00 版 (Apr. 12 1992) に、version 1.00a 緊急パッチ (May 11 1992) を当てたものになります。具体的には、README1.00a と skksrch.c

付録 E

SKK 関連ソフトウェア

SKK 関連のソフトウェアとしてよく知られている ari, kakasi, skkfep を簡単に紹介します. 以下は, これらのソフトウェアの作成者自身による紹介です.

E.1 ari

[1] 正式名称, 略称, 名前の由来

正式名称 … あり？

略称 … ari

名前の由来 … ベースとなった ae が「Ant's Editor」だったから、そして vi 風であり SKK 風でもあくまで「まがいもの」だから。

[2] どのようなソフトウェアか

64KB の壁がある MINIX でも日本語入力が可能なスクリーンエディタ。ae をベースとした Japanese fat ae with mock SKK です。

(1) vi 風の操作と、SKK 風の日本語入力インタフェース。ちょっと「たまご (on Nemacs)」の影響も受けてます。

(2) ファイル入出力と表示に、JIS コード、EUC、SHIFT-JIS コードが使えます。ファイル入出力コードと表示コードは分けて設定可能。

(3) 仮名漢字変換辞書は SKK 辞書フォーマットのファイルを利用。ユーザ辞書へは漢字登録可能。

とすれば part1 が入手できるので, それにより全体の個数を知ってから,

```
Subject:  all-2-10
```

のようにして全体の一部 (上の場合は part2 から part10) を入手することができる. またメイルの取りこぼしがあったときは, たとえば,

```
Subject:  all-22,31
```

とすると part22, part31 が入手できる.

D.2 Anonymous Ftp による入手方法

IP 接続のされているサイトからは anonymous ftp により SKK を入手することもできる.

東北大学 (skk.tohoku.ac.jp [130.34.200.30]) では,
'/pub/skk/6.32'に SKK6.32 が置いてある.

大阪大学 (ftp.ics.osaka-u.ac.jp [133.1.12.30]) では,
'/elisp/library/for-Nemacs/Kanji/SKK/6.32'の下に SKK6.32 がある.

スタンフォード大学 (sail.stanford.edu[36.28.0.130]) では,
'/pub/skk/6.32'に SKK6.32 が置いてある.

また, これらのサイトでは公式リリースのものの他に最新の SKK も置いてある.

D.3 その他の入手方法

SKK は 1988 年 4 月に第 1 版を JUNET に投稿して以来, 毎年 4 月に JUNET のニュースグループ fj.sources に新しいバージョンを投稿している. この方式もひきつづき続けていきたいと考えている.

SKK はまた Nemacs および Mule の contributed software でもあるので, これらの一部としても入手することができる.

付録 D

SKK の入手方法

D.1　メールによる入手方法

SKK の最新版をメールで入手するためには,

```
skk-source@sato.riec.tohoku.ac.jp
```

あてに, 欲しいファイルの名前を Subject: に書いたメールを送ればよい. たとえば, 'skk.el' がほしい場合は,

```
Subject:  skk.el
```

をヘッダに含むメールを送ればよい.

skk-source で入手できるファイルを知るためには,

```
Subject:  index
```

というメールを送ればよい.

'index'にはファイルサイズと最終更新日がはいっている. ファイルのバージョン等を知るには,

```
Subject:  VERSIONS
```

というメールを送ればよい.

SKK の最新版全体を入手するには,

```
Subject:  all
```

とすればよい. こうすると SKK のパッケージがいくつか (1993 年 4 月現在では 44) のメイルに分割されて送られてくる. 一度に多数のメイルを受け取ることができない場合には, まず,

```
Subject:  all-1
```

という行のはいったメイルを送付すればよい．この行は本文の 1 行目でなければならない．

メイリングリストから抜けるためには，

```
skk-leave@sato.riec.tohoku.ac.jp
```

へメールを送るとリストから削除される．

付録 C

SKK メイリングリスト

SKK に関する意見交換の場として SKK メイリングリストが開設されている. これまでメイリングリストで話題になったこととしては, SKK の虫の報告 / 修正, 質問 / 回答, SKK 改良のための提案等がある. また, 最新の SKK もメイリングリストを通して配布し, 参加者の協力を得て, テスト, 改良を重ねている.

メイリングリストへの参加は,

`skk-join@sato.riec.tohoku.ac.jp`

へメイルを送れば, 自動的に, 発信者のメイルアドレスが登録される.

メイリングリストへの記事の投稿は,

`skk@sato.riec.tohoku.ac.jp`

へ送れば自動的に全ての登録者に送付される.

メイリングリストに登録したアドレスを変更したい時は, 新しいメイルアドレスから,

`skk-change@sato.riec.tohoku.ac.jp`

あてに,

`Old:` *your-old-name*`@`*your-old-address*

Too big num!

数をパラメータとする変換により 9999999999999999999 より大きい数字を漢数字に変換しようとするときにこのメッセージがでる.

You are not using SKK SERVER.

辞書サーバを使用していないユーザがコマンド `skk-server-version`を実行するとこのメッセージがでる.

file **is not well formed!**

辞書ファイル *file* をバッファに読み込むときに, ファイルが SKK 辞書の形式に合っていないときにでるエラーメッセージ. 古い辞書の場合, 必要な註釈行がないことがあるのでチェックする必要がある.

'*char*' **is not valid here!**

メニューの選択で有効でない文字 *char* をタイプしたときにこのメッセージがでる.

SKK SERVER *3.7.3* **running on HOST** *venus*.

コマンド skk-server-versionにより出されるメッセージ．辞書サーバのバージョン番号とサーバが動いているホスト名が表示される．

SKK SERVER is active now.

辞書サーバの起動に成功したとき, このメッセージがでる．

SKK SERVER is not active.

コマンド skk-server-versionを実行したときに辞書サーバが動いていないとき, このメッセージがでる．

SKK SERVER is not active, I will activate it.....

SKK を最初に起動したとき，ユーザが skk-server-hostを指定していると, SKK は辞書サーバとの TCP 接続を試みる．これに失敗したとき, SKK は辞書サーバを起動しようとしてこのメッセージをだす．

SKK version *6.32* **of** *April 24, 1993*

コマンド skk-versionにより出されるメッセージ． SKK のバージョン番号とリリースの日付が表示される．

Saving Jisyo...

辞書バッファをユーザ辞書にセーブしているとき，このメッセージがでる．

Saving Jisyo...done.

辞書バッファをユーザ辞書にセーブしおわったとき，このメッセージがでる．

Shall I register this as okuri-ari entry: わた s / 渡 / ? (y or n)

変数 skk-auto-okuri-processの値が tのときに，辞書登録をしようとして, 送り仮名ありとして登録すべきか送り仮名なしとして登録すべきかをシステムが判定できないときにでるメッセージ．『y』をタイプすれば, 送り仮名あり，『n』をタイプすれば送り仮名なしとして登録される．

Inserting contents of *file*...**done**

辞書ファイルの読み込みが終了したときにこのメッセージがでる.

It seems that you have deleted ▼.

▼モードで確定するときに【▼】がユーザにより消されていると, このメッセージがでる. また, ▼モードから『x』で▽モードに戻ろうとして【▼】が消えているときにもこのメッセージがでる.

It seems that you have deleted ▽.

▽モードで確定するときに【▽】がユーザにより消されていると, このメッセージがでる. また, ▽モードから▼モードに移ろううとして【▽】が消えているときにもこのメッセージがでる.

JIS or EUC code (00nn or CR for Jump Menu):

『\』キーによるコードまたはメニューによる文字の入力のときにでるメッセージ.

No completions for "*string***"**

文字列 *string* についての読みのコンプリーションをしようとして候補が無いときにこのメッセージがでる.

No more completions for "*string***"**

文字列 *string* についての読みのコンプリーションをしようとして候補がこれ以上無いときにこのメッセージがでる.

No more previous completions.

文字列 *string* についての読みのコンプリーションをしようとしてひとつ前の候補が無いときにこのメッセージがでる.

No need to save Jisyo.

辞書バッファをユーザ辞書にセーブするとき, 前回セーブしてからバッファに変化がないときにこのメッセージがでる. このとき, バッファの内容は辞書にセーブされない.

が壊れている可能性があるのでバッファ「 *.skk-jisyo*」の内容をチェックする必要がある.

Header line for okuri-nasi entries is missing!

SKK は終了時にユーザ辞書ファイルにユーザ辞書バッファの内容をセーブする. このとき, バッファに本来存在していないといけない註釈行「;; okuri-nasi entries.」が存在しないとき, このメッセージがでる. 辞書が壊れている可能性があるのでバッファ「 *.skk-jisyo*」の内容をチェックする必要がある.

Henkan end point must be after henkan start point.

変換を開始しようとして, 変換終了点が変換開始点より前にあると, このメッセージがでる.

Henkan key may not contain a new line character.

変換を開始しようとして, 変換開始点と変換終了点の間に改行コードがあると, このメッセージがでる.

I don't understand this character.

カーソルのある位置の文字のコードを表示しようとして, その文字が JIS character でも ASCII character でもないとき, このメッセージがでる.

I have created an SKK record file for you.

SKK を最初に起動したとき, '.skk-record'が存在しないとき, SKK は空の '.skk-record'を作成し, このメッセージをだす.

I have created an empty Jisyo file for you.

SKK を最初に起動したとき, '.skk-jisyo'が存在しないとき, SKK は空の '.skk-jisyo'を作成し, このメッセージをだす.

Inserting contents of *file*...

辞書ファイル *file* を最初にバッファに読み込むときにこのメッセージがでる.

付録 B

SKK メッセージ一覧

SKK がエコーエリアに表示するメッセージとその意味を列挙する.

Cannot complete an empty string!

読みのコンプリーションを空の文字列に対して行なおうとするときにでるメッセージ.

Could not activate SKK SERVER.

SKK を最初に起動したとき, ユーザが `skk-server-host`を指定していると, SKK は辞書サーバとの TCP 接続を試みる. これに失敗したとき, SKK は辞書サーバの起動を試みる. 辞書サーバの起動ができなかったときこのメッセージがでる.

Cursor is at the end of the buffer.

カーソルのある位置の文字のコードを表示しようとして, カーソルがバッファの最後の位置にあるときにこのメッセージがでる.

Deleted prefix "*prefix*".

変数 `skk-echo`が `nil`のときローマ字かな変換の途中でローマ字のプレフィクスを `DEL`で消去するとこのメッセージがでる.

Header line for okuri-ari entries is missing!

SKK は終了時にユーザ辞書ファイルにユーザ辞書バッファの内容をセーブする. このとき, バッファに本来存在していないといけない註釈行「`;; okuri-ari entries.`」が存在しないとき, このメッセージがでる. 辞書

skk-mode[C-X C-J]

かなモードから SKK を起動する前の状態に戻る．SKK が起動されていない状態のときは SKK を起動する．

skk-save-jisyo

現在の辞書バッファの内容をユーザ辞書にセーブする．

skk-server-version

辞書サーバのバージョンとサーバが動いているホストの名前を知らせる．

skk-version

SKK のバージョンを知らせる．

j-start-henkan[SPC]

平仮名確定入力モードでは【SPC】を挿入する. ▽モードでは見出し語を確定し▼モードに移行して変換を開始する. ▼モードでは次候補を表示し, 次候補がないときは辞書登録モードに移行する.

j-today[@]

今日の日付をカーソルの位置に挿入する.

j-toggle-kana[q]

かなモードからカナモードに移行する. ▽モードでは見出し語をカナに変換する.

j-try-completion[TAB]

▽モードで読みのコンプリーションを開始する.

j-zenkaku-eiji[L]

全英モードに移行する.

j-zenkaku-henkan

▽モードで見出し語を全角文字に変換する.

j-zenkaku-insert

全英モードで skk-zenkaku-vectorで指定した全角文字を挿入する.

j-zenkaku-region

region 中の ascii 文字を対応する全角文字に変換する.

j-zenkaku-self-insert

タイプした ascii 文字を対応する全角文字に変換して挿入する.

skk-auto-fill-mode[C-X j]

自動右揃えモードに移行する.

skk-kill-emacs-without-saving-jisyo

辞書バッファの内容をユーザ辞書にセーブしないで Nemacs から抜ける.

j-katakana-region

region 中の平仮名をカタカナに変換する.

j-keyboard-quit[C-G]

▼モードでは, 変換の候補の表示を中止して直前の▽モードに戻る. ▽モードでは, 見出し語と「▽」を削除して, 平仮名確定入力モードに戻る. その他の場合は通常の C-Gと同じ動作をする.

j-mode-off[l]

かなモードからアスキー入力モードに移行する.

j-mode-off-in-minibuff[l]

ミニバッファでかなモードからアスキー入力モードに移行する.

j-newline[C-M]

改行コードを挿入する.

j-previous-candidate[x]

▼モードでは, 変換中の候補についてひとつ前の候補に戻る. 平仮名確定入力モードでは小文字の入力を開始する.

j-purge-from-jisyo[X]

▼モードで現在の候補を辞書バッファから消去する.

j-self-insert[! " # % ' ( ) + , - . 0 1 2 3 4 5 6 7 8 9]
[: ; < = > ? [ ] ^ _ ` { | } ~]

タイプした文字をそのまま挿入する.

j-set-henkan-point[A B C D E F G H I J K M N O P R S T]
[U V W Y Z]

平仮名確定入力モードでは▽モードに移行し見出し語の入力を開始する.

j-set-henkan-point-subr[Q]

カーソルの位置を見出し語の先頭とする▽モードに移行する.

j-hiragana-henkan

▽モードで見出し語中のカタカナを平仮名に変換する.

j-hiragana-region

region 中のカタカナを平仮名に変換する.

j-input-by-code-or-menu[\]

コードまたはメニューによる文字の入力.

j-insert-a[a]

ローマ字表現が『a』で終るかなの入力.

j-insert-comma[,]

『,』の入力

j-insert-e[e]

ローマ字表現が『e』で終るかなの入力.

j-insert-i[i]

ローマ字表現が『i』で終るかなの入力.

j-insert-period[.]

『.』の入力

j-insert-u[u]

ローマ字表現が『u』で終るかなの入力.

j-kakutei[C-J]

▼モードでは変換中の候補を確定し, 平仮名確定入力モードになる. ▽モードでは入力中の見出し語はそのままにして平仮名確定入力モードになる.

j-kana-input[b c d f g h j k m n o p r s t v w y z]

ローマ字かな変換による入力を開始する.

j-katakana-henkan

▽モードで見出し語中の平仮名をカタカナに変換する.

付録 A

SKK コマンド一覧

SKK のコマンドを以下に列挙する．かなモードでキーにバインドされているコマンドについては，コマンド名の後にそのキーを示す．キーにバインドされていないコマンドは

`M-x` *command*

で実行できる．

j-abbrev-comma

英字変換中に『,』をタイプすると通常はコンマが挿入されるが，読みの補完を行っているときには，ひとつ前の補完の候補に戻る．

j-abbrev-input[/]

英字変換のための見出し語の入力を開始する．見出し語はアスキー文字として入力される．

j-abbrev-period

英字変換中に『.』をタイプすると通常はピリオドが挿入されるが，読みの補完を行っているときには，次の補完の候補を表示する．

j-delete-backward-char[DEL]

カーソルの直前の文字を prefix argument で与えた個数だけ消去する．通常は 1 文字だけ消去する．

j-display-code-for-char-at-point[$]

カーソルのある位置の文字のコードを表示する．

6. region をひらがな / かたかな / 全角に変換する関数が使えるようになった. (→ 3.4, 3.6)

7. インクリメンタル・サーチでのサーチパターンの入力モードの制御のカスタム化ができるようになった. (→ 3.15)

8. ▼モードで RETを egg と同様確定だけに用いることもできるようになった. (→ 4.9)

7.4 SKK5.30 との相違点

SKK5.30 との主な相違点は以下の通りである.

1. Mule および Demacs でも同一のソースファイルで SKK が利用できるようになった. (→ 1, 3.1)

2. 共用辞書をバッファに取り込んで利用できるようになった. (→ 3.1, 4.26, 4.29, 6.1)

3. 見出し語に対する変換の候補を得るときに, 利用する辞書, 検索の方法, 順番を指定できるようになった. (→ 4.26, 6.1)

4. skk-jisyo にある見出し語については, 同じ見出し語について複数の候補があるときには, 確定により, 確定した候補が次回の変換では最初の候補になる. これに対して, 確定しても候補の順序を変えてほしくないものを集めた辞書が利用できるようになった. (→ 4.26, 4.28)

5. 見出し語に対する候補が見つかればただちに確定してほしい項目を集めた辞書が利用できるようになった. (→ 4.27)

6. 英語版のチュートリアルが利用できるようになった. (→ 3.1.5)

7. 読みの補完のときの動作を dabbrev と同様にするようことができるようになった. (→ 4.24)

8. 確定を実行するコマンドを任意のキーに割り当てることができるようになった. (→ 4.25)

2. カーソルの位置にある文字のコードを知ることができるようになった. (→ 3.11)

3. インクリメンタル・サーチで探索する日本語の文字列が改行で分断されていても見つかるようにした. (→ 3.15)

4. 辞書バッファをユーザの辞書ファイルにセーブするときの `kanji-fileio-code`を設定できるようにした. (→ 4.13)

5. 日付の入力の形式を選択できるようになった. (→ 3.11, → 4.14)

6. ローマ字かな変換にオートマトンの状態遷移表も使うことができるようになった. (→ 4.3)

7. 全角スペースの入力が容易になった. (→ 4.1)

8. 変数 `skk-kakutei-early`を導入し, 変換中の候補を確定するタイミングをより早くできるようにした. (→ 4.10)

9. SKK サーバの動いているホスト名とサーバのバージョンを知ることができるようになった. (→ A, → B)

7.3 SKK4.33 との相違点

SKK4.33 との主な相違点は以下の通りである.

1. 送り仮名の自動処理の機能を追加した. (→ 3.8)

2. 送り仮名の自動処理に関連して, 辞書を送り仮名のある見出し語からなる部分と送り仮名のない見出し語からなる部分に分離した. (→ 3.2)

3. プログラム実行変換の機能を追加した. (→ 3.12)

4. 数をパラメータとする表現の入力ができるようになった. (→ 3.13)

5. overwrite モードと picture モードが SKK のもとで使えるようになった. (→ 3.2)

第 7 章

旧バージョンとの相違点

7.1 SKK2.25 との相違点

SKK2.25 との主な相違点は以下の通りである.

1. サーバを用いた共用辞書の利用が可能になった. (→ 3, → 7)
2. 入力中のローマ字のエコーバックができるようになった. (→ 4.4)
3. 送り仮名のある単語の変換の方法が変った. (→ 3.7)
4. 自動的に右揃えをする機能を公開した. (→ 3.14)
5. VIP と SKK を併用するユーザのための定数 `skk-use-vip` を導入した. (→ 4.6)
6. ユーザがカスタム化できる変数はすべて, skk で始まるようにした. (→ 4)

7.2 SKK3.47 との相違点

SKK3.47 との主な相違点は以下の通りである.

1. コードや読み方のわからない文字をメニューを用いて入力できるようになった. (→ 3.9)

ujis2jis

EUC → JIS 変換コマンド

roma2kana

ローマ字 → かな (EUC) 変換コマンド

kana2roma

かな (EUC) → ローマ字変換コマンド

```
% sub 旧リスト 新リスト
```

とすればよい. 比べられるふたつのファイルはあらかじめソートされていなければならない.

skkconv は SKK サーバを使用してコマンドラインでかな漢字変換するためのコマンドである. 例えば「かんじ」という読みをもつ漢字を以下のように検索できる.

```
% skkconv kanji
漢字
幹事
感じ
......
%
```

この他に以下の perl ライブラリとコマンドがある.

codeconv.pl

JIS, EUC, SJIS 相互変換ライブラリ

roma2kana.pl

ローマ字 → 平仮名変換ライブラリ

kana2roma.pl

平仮名 → ローマ字変換ライブラリ

jis2sjis

JIS → SJIS 変換コマンド

sjis2jis

SJIS → JIS 変換コマンド

jis2ujis

JIS → EUC 変換コマンド

skktools には以下のプログラムがある.

skk2list

SKK 辞書をリスト形式に変換する.

pubdic2list

pubdic 辞書の名詞エントリをリスト形式に変換する.

list2skk

リスト形式を SKK 辞書に変換する.

adddummy

SKK 辞書ソートのためにダミー文字を加える.

removedummy

加えたダミー辞書を取り除く.

sub

ふたつのリスト形式の辞書の差分を計算する.

skkconv

漢字検索コマンド.

例えば既存の SKK 辞書と pubdic 辞書をマージして新しい SKK 辞書を作成するには次のように行なう.

```
% (skk2list skk-jisyo ; pubdic2list kihon.u) \ ; リスト形式を連結
        | adddummy \                              ; ダミー文字追加
        | sort -u \                               ; ソート
        | removedummy \                           ; ダミー文字削除
        | list2skk \                              ; SKK 辞書に変換
        > skk-jisyo.new
```

これらのツールでは EUC の辞書のみ取り扱い可能である. それ以外の辞書を使うときは後述の jis2ujis などのコマンドを加える必要がある.

ふたつのリスト形式の辞書の差分を計算するには,

[1] 作業用のディレクトリを用意し, そこに cd する.

[2] マージしたい辞書を *dic1*, *dic2* にコピーする.

[3] `M-x cd` 等により, emacs の default directory を [1] の作業用のディレクトリと一致させる.

[4] `M-x skk-merge` を実行する.

[5] 「辞書ファイル 1」と「辞書ファイル 2」を聞いてくるので *dic1*, *dic2* を答える. 計算が終了すると結果が `NEW-SKK-JISYO` という名前のファイルに書かれる. 結果の辞書はソートされている.

6.3.1.3 skk-diff

ふたつの SKK 形式の辞書の差分をつくるプログラムである.

[1] 作業用のディレクトリを用意し, そこに cd する.

[2] 差分をとりたい 2 つの辞書を *dic1*, *dic2* にコピーする.

[3] M-x cd 等により, emacs の default directory を [1] の作業用のディレクトリと一致させる.

[4] `M-x skk-diff`を実行する. 「辞書ファイル 1」と「辞書ファイル 2」を聞いてくるので, それぞれ, *dic1* と *dic2* を答える. 計算が終了すると, *dic1* と *dic2* の差分が'`SKK-JISYO-DIFF`'という名前のファイルに書かれる.

6.3.2 Perl によるユーティリティプログラム

perl でかかれたユーティリティプログラムは skktools と総称され, SKK の distribution ではディレクトリ'`skktools`'の下にある.

skktools は SKK 辞書をマージしたり pubdic 辞書を SKK 辞書に変換したりするためのツール群である. SKK 辞書は読みに対応する複数の漢字が『/』で区切られた構造をしているが, このツールでは読みと漢字が 1 対 1 に対応した形式（これをリスト形式と呼ぶことにします）のファイルを中間形式として取り扱う.

6.3.1.1 skk-check-jisyo

SKK は辞書バッファでは SKK 形式の辞書を扱い, その内容を更新している. バッファの内容は通常 SKK 辞書の形式を保ったまま更新されるが, SKK の古いバージョンにあった虫等により, ユーザの辞書に異常が生じている場合がある. このことをチェックし, また間違いではないが, 問題があるかもしれないような辞書の項目を見つけるためには以下のようにすればよい.

[1] `M-x skk-check-jisyo`を実行する. 「辞書ファイル: 」と聞いてくるので, チェックしたい辞書ファイルの名前をタイプする.

[2] 辞書をチェックした結果がバッファ「*jisyo check*」に表示される. このうち,

> 単語がひとつもない.
> 空単語.
> 行の最後が「/」でない.
> 読みの直後が「 /」でない.
> 読みと単語の区切りがない.

というメッセージのある行については, 辞書に問題があるので, 適当に修正する. これ以外のメッセージは, ユーザへの注意なので無視してもよい. 以上の修正を必要ならば行い, 結果を再びもとの辞書ファイルにセーブする.

6.3.1.2 skk-merge

このプログラムと次のプログラムでは辞書の見出し語のソートを shell script を用いて行う. これは最近の SKK 辞書のサイズが大きくなったために Emacs Lisp のソート関数を用いると emacs が core dump することがあるためである. この shell script のファイル '`emacs-sort`' をディフォールトの位置以外に置いてある場合は '`.emacs`' または '`.skk`' の中で変数 `skk-emacs-sort`の値に正しい full path name を設定する必要がある.

ふたつの SKK 形式の辞書をマージするコマンドである.

関数 j-okuri-search は送り仮名の自動処理のための関数である.

6.2 辞書バッファのセーブ

コマンド skk-save-jisyoを実行すれば, その時点での辞書バッファがユーザ辞書ファイルにセーブされる. コマンドの実行は

```
M-x skk-save-jisyo
```

を入力すればよい. Emacs を終了するときに自動的にセーブされるので, system down 等の心配が無ければ, 意識してセーブする必要は無い. また, 同時にその前の Emacs のセッションでセーブした辞書は '.skk-jisyo.BAK'にコピーされる.

また, なにかの都合で, 辞書バッファをセーブしないで Emacs を終了したいときは,

```
M-x skk-kill-emacs-without-saving-jisyo
```

とすればよい.

6.3 辞書ユーティリティプログラム

SKK 辞書に関するユーティリティプログラムには, emacs lisp でかかれたものと perl でかかれたものとがある.

6.3.1 Emacs Lisp ユーティリティプログラム

Nemacs の中でコマンド skk-merge, skk-diffを実行すれば, SKK 形式のふたつの辞書をマージしたり差分を計算することができる.

注意: 誤って貴重な辞書を破壊しないように, 以下の作業はすべて辞書をコピーしてから, コピーについて行うほうがよい.

```
(j-search-jisyo-file skk-initial-search-jisyo 10000 t)
(j-search-jisyo-file skk-jisyo 0 t)
(j-okuri-search)
(j-search-jisyo-file skk-large-jisyo 10000)
(j-search-server skk-aux-large-jisyo 10000))
```

である．このように skk-search-prog-list は任意個数のプログラムのリストである．それぞれのプログラムは, 変換の候補のベクトルを返すようなプログラムであれば何でもよい．SKK は, これらのプログラムを, 必要に応じて, リストにかかれている順に実行し変換の候補ベクトルをつくり, 候補として表示する．

上の例にでてくる関数 j-search-jisyo-file は 3 つの引数をとる関数である．最初の引数は nil またはファイル名であり, ファイル名を与えたときには, そのファイルをバッファに取り込んで見出し語を検索する．nil のときは空のベクトルを候補のベクトルとして返す．第 2 引数には 0 以上の整数を与える．0 のときはバッファの最初から見出し語を順に検索する．引数の値 n が正のときは, バッファが見出し語の順にソートされていると仮定し, 二分法で検索の範囲が n になるまで範囲を狭めていき, その先は線形に検索する．第 3 引数が nil のときは, 第 1 引数で指定したファイルをバッファに最初に取り込むときにメッセージを表示し, t のときは表示しない．

関数 j-serach-kakutei-jisyo-file の動作は j-search-jisyo-file と同様である．ただし, 候補が見つかったときには, その候補で自動的に確定をする．したがって, この関数で用いる辞書としては, その見出し語に対する候補がひとつしかないことがあらかじめわかっているものを集めてつくっておく必要がある．

関数 j-search-serverは 3 つの引数をとる関数であり, 辞書サーバを利用するユーザにだけ意味のある関数である．この関数は辞書サーバが正常に動作しているときには, サーバに変換を依頼し候補ベクトルを返すが, 辞書サーバにアクセスできなくなると自動的に第 1 引数で指定したファイル (ディフォールトでは 'skk-aux-large-jisyo') をバッファに読み込み, このファイルを用いて検索をする．残りの変数の意味は関数 j-search-server の引数の意味と同じである．

ない．さらに，送り仮名のつく見出し語をもつ行は，見出し語について逆順にソートされていないといけない．

送り仮名のない見出し語については，見出し語の順にソートされている辞書とそうでない辞書がある．SKK-JISYO.L のような大規模な辞書は，見出し語の順にソートされており，そのためにバッファに読み込んだ場合でも高速な検索が可能になっている．

一方，ユーザのユーザ辞書ファイル (‘.skk-jisyo’) は，SKK が起動されるとバッファに読み込まれる．変換はこの辞書バッファを用いて行われ，確定するたびに辞書バッファは更新される．例えば，今，この文を書いている時点で送り仮名のない見出し語をもつ行の最初の方は以下のようになっている．

```
いか /以下/
ほう /方/法/豊/訪/縫/泡/砲/放/報/鵬/鳳/
さいしょ /最初/
ぎょう /行/
みだしご /見出し語/
おくりがな /送り仮名/
じてん /時点/辞典/
ぶん /文/分/聞/
いま /今/
こうしん /更新/交信/恒真/
buffer /バッファ/
じしょ /辞書/
```

これで分るように，送り仮名なしの辞書バッファの項目は，つねに，最近確定したものが前に来るようなっている．このために，送り仮名のない見出し語については読みの補完が可能になっている．

SKK の基本的な変換はユーザ辞書と共用辞書を用いて行われるが，これらの辞書を含めて，ユーザはどの辞書を，どのような方法で，どのような順番でアクセスするかを指定することができる．この指定は変数 skk-search-prog-list により行われる．この変数のディフォールトの値は

```
((j-search-kakutei-jisyo-file skk-kakutei-jisyo 10000 t)
```

第 6 章

SKK 辞書とユーティリティプログラム

6.1 SKK 辞書の形式

SKK 辞書は以下のいずれかの形式の行からなる.

midashi /*kouho*/.../*kouho*/

または

midashi /*kouho*/.../*kouho*/[*okuri*/*kouho*/.../*kouho*/]/.../

最初の形式の行は送り仮名のつかない見出し語をもつ項目に対応するもので, たとえば, 次のような行がそうである.

いこう /以降/移行/意向/

ふたつめの形式の行は送り仮名のつく見出し語をもつ項目に対応するもので, たとえば, 次のような行がそうである.

おおk /大/多/[く/多/]/[か/多/]/[き/大/]/[け/多/]/

この他 SKK 辞書は, 先頭が「;;」である行を註釈行として許している. とくに SKK 辞書には次のふたつの註釈行がこの順で含まれていないといけない.

```
;; okuri-ari entries.
;; okuri-nasi entries.
```

また, 「;; okuri-ari entries.」の行と「;; okuri-nasi entries.」の行の間はすべて送り仮名のつく見出し語をもつ行でないといけなく, 「;; okuri-nasi entries.」の行の後はすべて送り仮名のつかない見出し語をもつ行でないといけ

```
(process-send-string "skkservd" "1 かんじ ")
;; server を disconnect する.
(process-send-string "skkservd" "0")
```

5.5.3 限界・改良の余地

skkserv がディバッグモードでない時に, 合法的に殺す手段はない. kill する.

ひとつの計算機の上で使用できる同じ番号の TCP ソケットは, ひとつしか存在できない. 従って, 標準ポート番号を使う skkserv も, ひとつしか動かない. ディバッグやテストの際に, skkserv を殺してすぐに起動すると, ポートが使われている, という趣旨のエラーが起きる. この場合は, 少し待ってから, 再度起動する.

ひとつのサーバが受けつけられるクライアントの数の上限は, 関数 `getdtablesize` により決まる. もし, 上限を越えた数のクライアントがコネクト要求をしてきた時はサーバは異常終了する.

本来の SKK 辞書には, 見出し語の長さの上限はないが, skkserv では 512 バイトに限っている (‘`skkserv.h`’の `BUFSIZE`の値). なお, 変換文字列の長さに制限はない.

辞書検索は, 「読み」の第 1 文字目による場合分けを行ったあとは, 線型探索をしている. 現在の状況では, これで十分であるが, 場合によっては, より高速な探索方法を検討する必要があるかもしれない.

5.5 その他 (辞書サーバについて)

5.5.1 skkserv が動作する環境

skkserv が動作するためには, BSD 系 UNIX のソケットが必要である.
skkserv のテストは,

- SunOS 4.0.3, 4.1.1
- NEWS OS 4.1.1
- Ultrix V4.1
- System V release 4.0

の上で行った. また, ユーザのレポートによると, HP-UX, MassComp の上でもインストール時に適切な設定をすると, 動作することがわかっている.

5.5.2 テスト方法

skkserv がうまく起動できない時は, Nemacs で, 以下を順次評価する. ここで, "tororin" のところには, skkserv を動かしているホスト名を書いておく.

```
;; network stream を open
(open-network-stream
    "skkservd" (current-buffer) "tororin" "skkserv")
;; 初期設定
(if (fboundp 'set-process-kanji-code)
    (set-process-kanji-code (get-process "skkservd") 0))
(setq kanji-fileio-code 0)
;; 変換すべき文字列として『かんじ』を送信する.
;; 成功すると, バッファの最後に,
;; 【1/ 漢字 / 幹事 / 完治 / 完爾 / 寛治 / 感じ /】
;; が挿入される.
```

`-d`	ディバッグ・モード
`-p` *nnnn*	通信用のポート番号として *nnnn* を使う.
~/jisho	*~/jisho* を辞書として利用.

オプションなしで起動すると, 通常モード（ディバッグ・モードでない）で, サーバのコンパイル時に指定した辞書とポートを利用する. skkserv は, サーバとなるべき子プロセスを作ったあと, ただちに終了する. エラーがあっても何のメッセージも出さない.

ディバッグ・モードで走らせると, skkserv はそのまま foreground で走り, メッセージを出力する. キーボードから割りこみをかけることもできる.

skkserv の起動はスーパーユーザが行う必要はない.

5.4 SKK 形式の辞書

SKK 形式の辞書の各行は, 見出し語, 1 文字の空白, 変換文字列からなる文字列である.

辞書の中で, 各行は, 見出し語を unsigned-char と見なして, ソートされていると仮定する. この順序は Emacs が文字列を比較するときの順序であり, Emacs のコマンド `sort-lines`を用いればファイルをこの順序でソートすることができる. UNIX の `sort` コマンドでの標準の順序とは異なるので注意が必要である.

見出し語は, 空白と改行を含まない任意の文字列である. 変換文字列は, 「/」 (スラッシュ) と改行を含まない任意の文字列を, 「/」 で区切って並べたものである. (最初と最後にも 「/」 がある.) いずれも漢字コードは EUC である.

skkserv は以下の形式の新形式辞書もサポートしている.

;; okuri-ari entries. パート 1 ;; okuri-nasi entries. パート 2

新形式辞書は, 各行の中身は変わらないが, 行の順序が旧形式と異なる. パート 1 は, 送り仮名のある見出語が逆順にソートされている. パート 2 は, 送り仮名のない見出し語が正順にソートされている.

また, 「;; 」ではじまる, 上の 2 種類の形でない行は, 注釈として無視される.

5.2.2 スーパーユーザの権限を持たない人のインストール方法

1. 'Makefile' を編集する.

   スーパーユーザの権限を持つ場合と同じ変更を加え, さらに,

   ```
   PRIVATE = -DPRIVATE
   ```

   とする. ETCDIR (skkserv の置き場所) は好きなところで良い.

2. skkserv を作り, インストールする.

   ```
   make
   make install
   ```

3. '.skk'に

   ```
   (setq skk-portnum 1178)
   ```

   を追加する.

4. SKK を使う前に, skkserv を動かす計算機の上で, 自分で,

   ```
   skkserv
   ```

   として, 起動する. (SKK よびだし時に, skkserv は自動的に起動されるようになっているが, 必ずしも成功しない.)

5.3 skkserv の起動

skkserv の起動時のオプションで, ディフォルト辞書名の変更, ディバッグ・モードの指定などができる.

`skkserv [-d] [-p` *nnnn*`] [`*jisho*`]`

自分の計算機の状況に応じて, 以下の変数の値を変更する.

COMPAT オペレーティングシステムに応じて選択する. BSD UNIX 系統であれば変更しない. 詳細は, 'skkserv/Makefile' を参照.

ETCDIR skkserv と辞書がインストールされるディレクトリであり, 通常, '$NEMACS/etc'にする.

JISYO SKK 辞書の名前であり, 通常 'SKK-JISYO.L'とする.

2. skkserv をコンパイルする.

```
make
```

3. スーパーユーザになり, skkserv をインストールする.

```
make install
```

このとき, 対話的 shell script が走り, '/etc/services' と '/etc/rc.local' の 2 つのファイルに若干の行を追加するかどうか質問する. yes と答えると, バックアップを取ったあと, これらのファイルに若干の行を追加する. no と答えた場合, 手作業で追加すべき内容が表示される.

4. NIS(旧名 YP) を使っている場合は, さらに, NIS のマスタサーバとなっている計算機でスーパーユーザになり, '/etc/services'に

```
skkserv 1178/tcp skk-jisyo-server
```

という行を加えた後,

```
cd /var/yp または cd /etc/yp
make services
```

とする.

第 5 章

辞書サーバ (skkserv)

5.1 skkserv とは

skkserv は, SKK のための辞書サーバである.

skkserv (サーバ) は, SKK/Nemacs (クライアント) とは独立に動作する. ひとつのサーバは, 複数のクライアントの要求を処理することができる.

サーバは, 起動されると, 最初に SKK 形式の辞書を読みこむ. 次に, 無限ループにはいり, クライアントからの要求を待つ. サーバは, 接続を要求したクライアントごとに通信路を作り, 以後, そのクライアントから文字列変換の要求が起きる毎に, 辞書を検索して, 変換文字列を返す. クライアントは, 文字列変換要求以外に, サーバのバージョン番号要求や接続終了などのメッセージをサーバに送る.

サーバとクライアントの間の通信には, BSD 系 UNIX のソケットを利用する. System V 系 UNIX でも, BSD 系のネットワーク機能を取りこんでいるものは動作することがある. 詳しくは, 'skkserv/Makefile' の記述を参照.

サーバとクライアントは, ネットワークで接続された異なる計算機上で動かすことができる.

5.2 skkserv のインストール方法

5.2.1 スーパーユーザの権限を持つ人のインストール方法

1. 'Makefile' を編集する.

で検索をするファイルを指定する．このファイルで見つかった候補については自動的に確定する．

4.28 skk-initial-search-jisyo

skk-search-prog-list の設定がディフォールト値のときに, skk-kakutei-jisyo の後に検索され, skk-jisyo の前に検索されるファイルを指定する．skk-jisyo にある見出し語については, 同じ見出し語について複数の候補があるときには, 確定により, 確定した候補が次回の変換では最初の候補になる．skk-initial-search-jisyo については確定しても候補の順序は変らない．したがって, このファイルには, 常に一定の順序で候補がでてきてほしい見出し語を置いておくとよい．

4.29 skk-large-jisyo

skk-search-prog-list の設定がディフォールト値のときに, バッファに読み込まれる共用辞書を指定する．このファイルはユーザ辞書で候補が見つからなかったときに検索される．

4.30 skk-aux-large-jisyo

skk-search-prog-list の設定がディフォールト値のときに, バッファに読み込まれる共用辞書を指定する．普段辞書サーバにより共用辞書を検索するように設定しているユーザが, このファイルを指定しておくと, 辞書サーバが利用できなくなったときには, 自動的にこのファイルをバッファに読み込んで検索するようになる．

4.23 skk-backup-jisyo

ディフォールト値は "~/.skk-jisyo.BAK". SKK 終了時に, この変数に指定したファイルに, 前回 SKK を終了したときにセーブしたユーザ辞書を移動する. 変数の値が nil のときは backupfile は作成されない.

4.24 skk-dabberv-like-completion[nil]

読みの補完の動作を dabbrev と同じようにするかどうかを指定する変数. この変数が t のときは, TAB を 2 回以上連続して続けたときに, 2 回目以降についても最初に指定した文字列についての補完を行うようになる.

4.25 skk-kakutei-key

確定を実行するキーを指定するための変数. ディフォールト値は "C-j" である.

4.26 skk-search-prog-list

見出し語を変換するための辞書について, どの辞書を, どのような方法で, どのような順番でアクセスするかを指定する変数. ディフォールトの値は

```
((j-search-kakutei-jisyo-file skk-kakutei-jisyo 10000 t)
 (j-search-jisyo-file skk-initial-search-jisyo 10000 t)
 (j-search-jisyo-file skk-jisyo 0 t)
 (j-okuri-search)
 (j-search-jisyo-file skk-large-jisyo 10000)
 (j-search-server skk-aux-large-jisyo 10000))
```

である. より詳しくは, see 6.1 [SKK 辞書の形式], page 53.

4.27 skk-kakutei-jisyo

skk-search-prog-list の設定がディフォールト値のときに, 関数 j-search-kakutei-jisyo-file

4.19 skk-server-host

ディフォールト値は (getenv "SKKSERVER")の値. 変数の値が nilでないときは, その値を辞書サーバのホスト名として SKK は辞書サーバを用いた変換を行う. (see 5.2.2 [スーパーユーザの権限を持たない人のインストール方法], page 49)

4.20 skk-serv

ディフォールト値は (getenve "SKKSERV")の値. 辞書サーバ・ホスト上での SKK 辞書サーバ・プログラムの名前.

4.21 skk-num-type-list

数をパラメータとする表現の入力のときに, 数字列をどのように解釈するかを決めるタイプとそのタイプでの解釈を行う関数の対のリスト. ディフォールト値は:

```
((?0 . j-id)
 (?1 . j-zenkaku-num-str)
 (?2 . j-kanji-num-str)
 (?3 . j-kanji-num-str2)
 (?9 . j-shogi-num-str))
```

see 3.13 [数をパラメータとする表現の入力], page 26

4.22 skk-numeric-conversion[t]

変数の値が tのときは数をパラメータとする表現の入力が可能になる. (see 3.13 [数をパラメータとする表現の入力], page 26) この機能を使いたくないときは値を nilにすればよい.

4.12 skk-record-file

SKK での変換に関する統計を記録するファイルの名前.

4.13 skk-jisyo-code[nil]

辞書バッファにユーザ辞書ファイルを読み込むときと, 辞書バッファをユーザ辞書ファイルに書き出すときの kanji-fileio-codeを指定するのに用いる. 0ならば EUC, 1ならば Shift-JIS, 2ならば JIS となる.

4.14 skk-date-ad[nil], skk-number-style[1]

『@』による日付の入力での出力の形式を指定する. skk-date-adの値が tのときは西暦で表示され, nilのときは元号 (平成) で表示される. skk-number-styleが 0または nilのときは年月日がアスキー (半角) の数字で表示され, 1または tのときは全角の数字で表示され, 2のときは漢数字で表示される.

4.15 skk-mode-string["--SKK:"]

モード行で SKK のモードが ascii モードであることを知らせる文字列.

4.16 skk-hirakana-mode-string["-- かな "]

モード行で SKK のモードがかなモードであることを知らせる文字列.

4.17 skk-katakana-mode-string["-- カナ "]

モード行で SKK のモードがカナモードであることを知らせる文字列.

4.18 skk-zenei-mode-string["-- 全英 "]

モード行で SKK のモードが全英モードであることを知らせる文字列.

うご *k】となり, 『UgoKu』までタイプして, 始めて, かな漢字変換が起動されて, 画面は【▼動く】となる.

したがって, 辞書が以下のようになっているとき:

おお k /大/多/[く/多/]/[か/多/]/[き/大/]/

変数の値が nilのときは『OoKii』はつねに【大きい】となり, 『OoKu』はつねに【多く】となるが, tのときは『OoK』までで変換を開始するのでそのようにはならない.

4.8 skk-auto-okuri-process[nil]

変数の値が tのときは送り仮名の自動処理を行う. see 3.8 [送り仮名の自動処理], page 21

4.9 skk-egg-like-newline[nil]

変数の値が tのときは, ▼モードで RETをタイプしたときに確定をするだけで改行は挿入されない.

4.10 skk-kakutei-early[t]

▼モードでかつ skk-process-okuri-earlyの値が nilのときこの変数の値が tならば, 印字可能な文字または RETをタイプすると直ちに確定する. ▼モードでかつ skk-process-okuri-earlyの値が nilのときこの変数の値が nilならば以下のキーをタイプすると副作用として確定する.

( ) [ ] , . A-Z RET

skk-process-okuri-earlyの値が tのときは, skk-kakutei-earlyの値は SKK の動作には影響しない.

4.11 skk-jisyo

ユーザ辞書ファイルの名前.

```
Thu Apr 18 21:00:00 1991 登録:  2 確定:  366 確定率:  99% 語数:  5184
```

変換に関する統計をとりたくなければ,

```
(setq skk-keep-record nil)
```

とすればよい.

4.6 skk-use-vip[nil]

SKK と VIP を併用するユーザは, skk-use-vipの値を tにセットする必要がある.

さらに, '.vip'ファイルに以下を追加する必要がある.

```
(defun vip-read-string (prompt &optional init)
  (setq save-minibuffer-local-map (copy-keymap minibuffer-local-map))
  (define-key minibuffer-local-map "\C-h" 'backward-char)
  (define-key minibuffer-local-map "\C-w" 'backward-word)
  (let (str)
    (condition-case conditions
        (setq str (read-string prompt init))
      (quit
       (setq minibuffer-local-map save-minibuffer-local-map)
       (signal 'quit nil)))
    (setq minibuffer-local-map save-minibuffer-local-map)
    str))
```

4.7 skk-process-okuri-early[nil]

skk-process-okuri-earlyの値が tのときは,送り仮名のある単語の変換を, 送り仮名を構成するローマ字の最初の 1 文字を読んだ時点で, 開始する. 例えば,『UgoK』までタイプすると画面は【▼動 k】となる. skk-process-okuri-earlyの値がディフォールト値の nilのときは,『UgoK』までタイプすると画面は【▽

```
("oh" "h" ("" . ""))
```

を追加すればよい.

この他,

```
tp      ⇒ 東北大学
skk     ⇒ skk
skK     ⇒ SKK
```

といった変換も規則:

```
("tp" nil ("東北大学" . "東北大学"))
("sk" "sk" ("" . ""))
("skk" nil ("skk" . "skk"))
("skK" nil ("SKK" . "SKK"))
```

を追加すれば可能である. 自分の名前を入力することはよくあるので, 適当な省略形を用いて, このリストに追加しておくと便利かもしれない.

4.4 skk-echo[t]

以下の変数の説明では, 変数名の後にそのディフォールト値を書いておく.

skk-echoの値が tのときは, 入力中のローマ字でかなに変換されていない部分が画面にエコーバックされる. たとえば, 『chotto』とタイプすると, 画面は,

```
c → ch →ちょ→ちょ t →ちょっ t →ちょっと
```

と変化する. 値を nilにセットするとエコーバックはされなくなる.

4.5 skk-keep-record[t]

skk-keep-recordの値が tのときは, 辞書バッファをユーザ辞書ファイルにセーブするときに, 変換に関する統計が以下のような形で‘.skk-record’に記録される.

（現在の状態@入力 次の状態 出力）

という形のリストで表現している．ここで，「状態」は変換される前のローマ字の文字列であり，「入力」は今タイプされた文字を長さ 1 の文字列とみなしたものである．「@」は文字列の連接をあらわす．「出力」は次の状態に遷移するときにバッファに挿入される文字列の対であり，かなモードでは右要素，カナモードでは左要素がバッファに挿入される．「次の状態」が nil のときには最終状態に遷移し，ローマ字かな変換を終了する．

したがって，上のリストの初期値は，『nn』または『n'』をタイプするとかなモードでは【ん】となりカナモードでは【ン】となることを示している．また，もし，'.skk'等で，

```
(setq skk-rom-kana-rule-list
    '(("nn" "n" ("ン" . "ん"))
      ("n'" nil ("ン" . "ん"))))
```

とすれば，

```
hannou ⇒ はんのう
han'ou ⇒ はんおう
```

のようにローマ字かな変換が行われる．

SKK のローマ字かな変換は，また，

```
ohsaka  ⇒ おおさか
ohta    ⇒ おおた
ohhira  ⇒ おおひら
...
```

のように『oh』を特別に処理している．このために，例えば，『ohhonn』とタイプすると【おおほん】となり，【おっほん】とはならない．

```
ohhonn  ⇒ おっほん
```

のように変換したければ，規則:

```
("g" . ("が" . "ガ")) ("gy" . ("ぎゃ" . "ギャ"))
("h" . ("は" . "ハ")) ("hy" . ("ひゃ" . "ヒャ"))
("j" . ("じゃ" . "ジャ")) ("jy" . ("じゃ" . "ジャ"))
("k" . ("か" . "カ")) ("ky" . ("きゃ" . "キャ"))
("m" . ("ま" . "マ")) ("my" . ("みゃ" . "ミャ"))
("n" . ("な" . "ナ")) ("ny" . ("にゃ" . "ニャ"))
("oh" . ("は" . "ハ"))
("p" . ("ぱ" . "パ")) ("py" . ("ぴゃ" . "ピャ"))
("r" . ("ら" . "ラ")) ("ry" . ("りゃ" . "リャ"))
("s" . ("さ" . "サ")) ("sh" . ("しゃ" . "シャ"))
("sy" . ("しゃ" . "シャ"))
("t" . ("た" . "タ")) ("th" . ("てぁ" . "テァ"))
("ty" . ("ちゃ" . "チャ"))
("v" . ("う゛ぁ" . "ヴァ")) ("w" . ("わ" . "ワ"))
("x" . ("ぁ" . "ァ")) ("xk" . ("か" . "ヵ"))
("xw" . ("ゎ" . "ヮ")) ("xy" . ("ゃ" . "ャ"))
("y" . ("や" . "ヤ"))
("z" . ("ざ" . "ザ")) ("zy" . ("じゃ" . "ジャ"))))
```

4.3 skk-rom-kana-rule-list

ローマ字かな変換は上述のような表にしたがって行われるが, リスト skk-rom-kana-rule-list より表現される有限状態オートマトンを用いた変換も併用される. このリストの初期値は

```
(("nn" nil ("ン" . "ん"))
 ("n'" nil ("ン" . "ん")))
```

である. リストの各要素はオートマトンの状態遷移規則:

<現在の状態，入力，次の状態，出力>

を

4.2 skk-prefix-list , skk-roma-kana-[aiueo]

SKK ではローマ字かな変換もカスタム化することができる. 例えば, 訓令式でしか入力できないようにすることもできる. (そうすると, 『c』, 『f』, 『j』を他のよく使う関数に割り当てることができる.)

カスタム化は以下のようにして行う. まず, 使いたいローマ字かな変換の対応表を用意する. ただし, かなに変換されるローマ字の列は, 【ん】に変換される『nn』と『n'』以外はすべて最後の文字は母音であるとする. この最後の母音を除いたローマ字の列をプレフィクスという.

変数 skk-prefix-listに長さが 2 以上のすべてのプレフィクスのリストをセットする. 初期値は次のようになっている.

```
(setq skk-prefix-list
      '("by" "ch" "cy" "dh" "dy" "fy" "gy" "hy" "jy"
        "ky" "my" "py" "ry" "sh" "sy" "th" "ts" "ty"
        "xk" "xt" "xts" "xw" "xy" "zy"))
```

次に, 対応表にしたがって, 変数:

skk-roma-kana-a, skk-roma-kana-i, skk-roma-kana-u,
skk-roma-kana-e, skk-roma-kana-o

をセットする. ここで, 例えば, skk-roma-kana-aは『a』で終るローマ字列の変換を定める変数である. 具体的には, 以下の skk-roma-kana-aの初期値を参考にして行えばよい.

```
(setq skk-roma-kana-a
  '(("" . ("あ" . "ア"))
    ("b" . ("ば" . "バ")) ("by" . ("びゃ" . "ビャ"))
    ("ch" . ("ちゃ" . "チャ")) ("cy" . ("ちゃ" . "チャ"))
    ("d" . ("だ" . "ダ")) ("dh" . ("でゃ" . "デャ"))
    ("dy" . ("ぢゃ" . "ヂャ"))
    ("f" . ("ふぁ" . "ファ")) ("fy" . ("ふゃ" . "フャ"))
```

```
      (copy-keymap (default-value 'skk-input-vector)))
  (aset skk-input-vector ?! nil)
  (aset skk-input-vector ?, nil)
  (aset skk-input-vector ?. nil)
  (aset skk-input-vector ?: nil)
  (aset skk-input-vector ?; nil)
  (aset skk-input-vector ?? nil))
```

フック関数の中で上のような関数を実行すれば Emacs のモードに依存して自動的に input vector をカスタム化できる.

同様に変数 skk-zenkaku-vectorを set することにより, 全英モードでのキー入力をカスタム化できる.

skk-zenkaku-vectorの初期値は以下のようである.

```
[nil  nil  nil  nil  nil  nil  nil  nil
 nil  nil  nil  nil  nil  nil  nil  nil
 nil  nil  nil  nil  nil  nil  nil  nil
 nil  nil  nil  nil  nil  nil  nil  nil
 "　" "！" "”" "＃" "＄" "％" "＆" "’"
 "（" "）" "＊" "＋" "，" "－" "．" "／"
 "０" "１" "２" "３" "４" "５" "６" "７"
 "８" "９" "：" "；" "＜" "＝" "＞" "？"
 "＠" "Ａ" "Ｂ" "Ｃ" "Ｄ" "Ｅ" "Ｆ" "Ｇ"
 "Ｈ" "Ｉ" "Ｊ" "Ｋ" "Ｌ" "Ｍ" "Ｎ" "Ｏ"
 "Ｐ" "Ｑ" "Ｒ" "Ｓ" "Ｔ" "Ｕ" "Ｖ" "Ｗ"
 "Ｘ" "Ｙ" "Ｚ" "［" "＼" "］" "＾" "＿"
 "‘" "ａ" "ｂ" "ｃ" "ｄ" "ｅ" "ｆ" "ｇ"
 "ｈ" "ｉ" "ｊ" "ｋ" "ｌ" "ｍ" "ｎ" "ｏ"
 "ｐ" "ｑ" "ｒ" "ｓ" "ｔ" "ｕ" "ｖ" "ｗ"
 "ｘ" "ｙ" "ｚ" "｛" "｜" "｝" "～" nil]
```

したがって, かなモードで「*」や数字等を対応する全角の文字で入力したければ, 以下を'.skk'に入れておけばよい.

```
(setq skk-input-vector
  [nil  nil  nil  nil  nil  nil  nil  nil
   nil  nil  nil  nil  nil  nil  nil  nil
   nil  nil  nil  nil  nil  nil  nil  nil
   nil  nil  nil  nil  nil  nil  nil  nil
   nil   "！" "”" "＃" "＄" "％" "＆" "’"
   "（" "）" "＊" "＋" "，" "ー" "．" "／"
   "０" "１" "２" "３" "４" "５" "６" "７"
   "８" "９" "：" "；" "＜" "＝" "＞" "？"
   nil  nil  nil  nil  nil  nil  nil  nil
   nil  nil  nil  nil  nil  nil  nil  nil
   nil  nil  nil  nil  nil  nil  nil  nil
   nil  nil  nil  "「" nil  "」" "＾" "＿"
   "‘" nil  nil  nil  nil  nil  nil  nil
   nil  nil  nil  nil  nil  nil  nil  nil
   nil  nil  nil  nil  nil  nil  nil  nil
   nil  nil  nil  "｛" "｜" "｝" "～" nil])
```

バッファに依存して異なるinput vectorを用いることもできる. skk-input-vectorはグローバルな変数なので, そのためにはバッファにローカルな同じ名前の変数をつくりその変数の値の設定を変更すればよい. たとえば, 次の関数をあるバッファで実行すれば, そのバッファでは区切記号「!」, 「,」, 「.」, 「:」, 「;」, 「?」をascii文字のまま入力できるようになる.

```
(defun skk-ascii-punc ()
  "use ascii punctuation marks."
  (interactive)
  (make-local-variable 'skk-input-vector)
  (setq skk-input-vector
```

```
  ! " # $ % & '
( ) * + , - .
0 1 2 3 4 5 6 7
8 9 : ; < = > ?
      [   ] ^ _
`
      { | } ~
```

これらのキーに対応して入力される文字列は変数 skk-input-vectorで定まり, その初期値はベクトル:

```
[nil  nil  nil  nil  nil  nil  nil  nil
 nil  nil  nil  nil  nil  nil  nil  nil
 nil  nil  nil  nil  nil  nil  nil  nil
 nil  nil  nil  nil  nil  nil  nil  nil
 nil  "！" nil  nil  nil  nil  nil  nil
 nil  nil  nil  nil  "、" "ー" "。" nil
 nil  nil  nil  nil  nil  nil  nil  nil
 nil  nil "：" "；"  nil  nil  nil  "？"
 nil  nil  nil  nil  nil  nil  nil  nil
 nil  nil  nil  nil  nil  nil  nil  nil
 nil  nil  nil  nil  nil  nil  nil  nil
 nil  nil  nil  "「" nil  "」" nil  nil
 nil  nil  nil  nil  nil  nil  nil  nil
 nil  nil  nil  nil  nil  nil  nil  nil
 nil  nil  nil  nil  nil  nil  nil  nil
 nil  nil  nil  nil  nil  nil  nil  nil]
```

である. このベクトルはキーの ascii code の順に, 対応する文字列を並べたものである. 要素が nil のところは, 使用されないか, またはそのキーがそのまま入力されることを示している. したがって, 例えば, 『!』は【！】と入力されるが, 『*』は, そのまま, 【*】として入力される.

第 4 章

カスタム化

ファイル‘~/.skk’があれば, SKK をロードしたときに, このファイルもロードされるので, このファイルにより SKK をカスタム化することができる. また‘.skk’をバイトコンパイルしたファイル‘.skk.elc’があれば‘.skk’のかわりにこれがロードされる.

注意: Nemacs のバージョンが古い (v2 以前) 場合は, ‘~/.skk’は EUC code でないといけない.

また, 関数 skk-modeは最後に

```
(run-hooks 'skk-mode-hook)
```

を実行するので, skk-mode-hookによりモードに依存したカスタム化も可能である.

基本的なカスタム化は変数のディフォールト値を変更することにより行われる. これら, ユーザがカスタム化できる変数はすべて, ‘skk.el’で defconstにより定義されているので, ‘.emacs’で設定しても SKK がロードされたときに再設定されてしまう. したがって, カスタム化は‘.emacs’の中でなく‘.skk’または skk-mode-hookで行わないといけない.

4.1 skk-input-vector , skk-zenkaku-vector

かなモードで以下のキーにより入力される文字列は自由に設定し直すことができる.

るとき『$』をタイプすると:

```
『Ａ』  EUC: a7a1 (167, 161), JIS: 2721 ( 39,  33)
```

がミニバッファに表示され, この文字がロシア文字であることがわかる.

skk-isearch-start-mode	サーチ・パターンの入力モード
nil	バッファで skk-mode が実行されていれば, そのバッファの入力モード. そうでなければ ascii モード.
ascii	ascii モード.
kana	かなモード.

バッファ・ローカルな変数 skk-isearch-start-mode と skk-isearch-use-previous-mode は '.skk' で setq-default でセットしたり, フック関数の中でセットしたりできる.

さらに変数 j-ignore-expを新たに導入した. この変数の値が nilでないときは, その値は正規表現を表す文字列として, 探索の対象となる文字列の連続した日本語文字の間に挿入され, その結果が regular expression search の対象となる. j-ignore-expのディフォールト値は "[\n\t <>|]*"である. このようにすると, 例えば, 探したい単語の途中に改行が入っていたり, さらにその結果が『>>』等で引用されていても見つけることができる.

同じようなことは, 英文でもおきる. 例えば, 【regular expression】という文字列を探すときに, 通常のインクリメンタル・サーチでは【regular】と【expression】の間に改行が入っていると見つからないが, このような場合にも SKK のインクリメンタル・サーチではマッチするようにした. すなわち, アスキー文字の直後にスペースがあるときは, このふたつの文字の間に j-ascii-ignore-expの値の文字列を挿入し, スペースを削除して regular expression search をする. j-ascii-ignore-expのディフォールト値は j-ignore-expのディフォールト値と同じである.

インクリメンタル・サーチを『C-u C-s』のようにプレフィクスを与えて起動すれば, そのときのサーチは j-ignore-exp, j-ascii-ignore-expをともに nilにした状態で起動される.

3.16 文字コードを知る方法

バッファでカーソルのある位置にある文字のコードを知りたいときは, かな / カナモードで『$』をタイプすればよい. 例えば, カーソルが【A】の上にあ

3.15 インクリメンタル・サーチ

SKK では, 日本語のインクリメンタル・サーチを英字と同様に行なうことができる. より詳しくは以下のようになる.

『C-s』等でインクリメンタル・サーチを起動すると

【I- さーち】 【I- サーチ】 【I- s 全英:】 【I-search:】

の 4 種類のプロンプトのどれかがエコー・エリアに表示され, これによりサーチ・パターンの入力モードを知ることができる. たとえばプロンプトが【I- さーち】のときに『ai』とタイプすれば「あい」がサーチされる. 同様に残りのプロンプトについては, それぞれ, 「アイ」, 「ａｉ」, 「ai」がサーチされる.

この 4 種類のプロンプトのどれになるかは次のようにして決まる.

まず, 現在のバッファが日本語を含むかどうかをチェックする. 日本語を含んでいないときは (日本語をサーチする必要はないので) 【I-search:】となる. (このときは従来のインクリメンタル・サーチの関数が実行される.)

日本語を含むバッファについては次のようになる.

1. このバッファでは, 今回が最初のインクリメンタル・サーチの実行のとき:
   バッファ・ローカルな変数 skk-isearch-start-modeの値にしたがって決める.

2. 1でないとき:

   (a) バッファ・ローカルな変数 skk-isearch-use-previous-mode が t のとき:
   このときは, このバッファでの前回のサーチでの最後の入力モードを用いる.

   (b) skk-isearch-use-previous-mode が nil のとき:
   1 と同様, skk-isearch-start-mode の値にしたがって決める.

ここで, skk-isearch-start-mode の値は nil, ascii, kana のどれかで, それぞれ以下のように解釈する.

変換の対象となる数字列のリスト (今の場合は ("3" "4")) を保持しているからである.

西暦と元号の間の変換や, 各種の単位の変換もそのためのプログラムを用意すれば実現できる.

次にタイプ「4」について説明する. たとえば, 辞書に,

```
125 /東京都葛飾区/
```

があるとき, 『/125 』を【▼東京都葛飾区】と変換したければ, 辞書に,

```
# /#4/
```

を登録しておけばよい. 同様に, 辞書が

```
p# /#4/
125 /東京都葛飾区/
```

となっていれば, 『/p125 』は【▼東京都葛飾区】と変換される.

SKK では以上述べた機能をディフォールトでサポートしている. これらの機能を用いたくないときは, 変数 skk-use-numeric-conversion を nil にすればよい. (see 4 [カスタム化], page 33)

3.14 自動右揃え

(skk-mode)のかわりに (skk-auto-fill-mode)を『C-x j』で起動すれば, 行の右揃えが自動的に行われる. すでに SKK が起動されているときも同様である. これを自動右揃えモードという. 右端の column の値は変数 fill-columnの値で定められる. 右揃えを行う関数は, 確定のたびに実行されるようになっている. また, 現在自動右揃えモードのときに『C-x j』をタイプするとこのモードから抜けることができる.

```
#m#d /#0 月 #0 日 /
```

となっていれば『/2m25d』は【2 月 25 日】となる.

さらに将棋の棋譜の入力用におまけでつくったタイプ「9」があって, これを使えば, 『Q34kin』(または, 『/34kin』) を【3 四金】と変換できる.

これらの処理には各タイプとそれに対応して用いられる 1 変数関数の名前を対にした association list が用いられる. これらの関数は見出し語で対応する場所にある変換の対象となる数字列を引数として受け取り実行される. どのタイプにどの関数を対応させるかは変数 skk-num-type-list で管理され, ユーザが変更することができる. (あまり自由にタイプを設定すると辞書を共有して使えなくなる.)

辞書への登録は以下のようにする. たとえば, 「2 がつ 25 にち」を変換しようとして, 見出し語「# がつ # にち」が辞書に無いときは, 【# がつ # にち】というプロンプトで辞書登録モードになる. このとき, 「2 月 2 5 日」と変換したいときは, ミニバッファで「#1 月 #1 日」をつくり, RETをタイプすると

```
# がつ # にち /#1 月 #1 日 /
```

が辞書バッファに登録される.

プログラム実行変換を利用すれば, もっと一般的な変換が可能である. 例として, 「3x4」を変換して, (かけ算の結果)「12」を得ることを考えてみる. '.skk'等で,

```
(defun times ()
  (int-to-string
   (* (string-to-int (car j-num-list))
      (string-to-int (car (cdr j-num-list))))))
```

と定義して, 辞書バッファに

```
#x# /(times)/
```

を登録すれば, 『/3x4』 +SPCでかけ算ができるようになる. これで目的の計算ができるのは, 関数 times の実行時に, 変数 j-num-list が見出し語の中の

3.13 数をパラメータとする表現の入力

「４月１０日」,「第１２回」といった数をパラメータとしてもつ表現を入力するための特別の方法を SKK では用意している. 日付の入力の項で見たように, 日本語では数を表現する方法が何通りもある. 「第１２回」についても, 他に「第 12 回」,「第一二回」,「第十二回」などの表現が可能である. このことに対処するために, SKK の (送り仮名なしの) 辞書では

だい # かい /第 #0 回 / 第 #0 回 / 第 #1 回 / 第 #2 回 / 第 #3 回 /

のような行を用いる. SKK ではこのように見出し語中に「#」があるとき, これは任意の空でない数字の列とマッチするとして処理をする. したがって, 『Dai15kai』とタイプすると画面は【▽だい１５かい】となり, ここでスペースをタイプすると, 辞書の検索がされて上の行の項目をもとにした候補の表示がされ, スペースキーを押すことにより, 順次, 以下のように変換される.

第 15 回 ⇒ 第 15 回 ⇒ 第１５回 ⇒ 第一五回 ⇒ 第十五回

これで想像できるように, 辞書の項目の中にある「#」+ ascii 文字の部分は, 変換の対象となる数字列 (今の場合は "15") をどのように処理するかという, 処理のタイプを「#」の直後の ascii 文字で指定するようになっている. そして, これらの文字が「0」,「1」,「2」,「3」の場合は上のように変換される.

もうひとつ例をあげると, 辞書が

/#3/

となっているときに, 『Q100200300400500』(『/100200300400500』でもよい) とタイプして SPCで変換すると【百兆二千三億四十万五百】となる.

また, 辞書が

がつ # にち /#2 月 #1 日 /

となっていれば『Q2gatu25niti』は【二月２５日】となる. 同様に,

3.11 日付の入力

かなモードで『@』をタイプすれば今日の日付が入力される. 日付の形式は変数 skk-date-ad(ディフォールト値は nil) と skk-number-style(ディフォールト値は 1) により選択することができ, 以下の表のようになる.

skk-date-ad	skk-number-style	表示される日付
nil	0 or nil	平成 3 年 1 月 16 日 (水)
nil	1 or t	平成 3 年 1 月 1 6 日 (水)
nil	2	平成三年一月一六日 (水)
t	0 or nil	1991 年 1 月 16 日 (水)
t	1 or t	1 9 9 1 年 1 月 1 6 日 (水)
t	2	一九九一年一月一六日 (水)

また 'SKK-JISYO.L' には, 見出し語「today」のもとに上のすべての組み合わせが, プログラム実行変換の機能を用いて登録されている. したがって, 『/today』としてスペースを続けて押せば, 今日の日付が上の形式で順次候補として表示される.

3.12 プログラム実行変換

送り仮名のない辞書の変換の候補に Emacs Lisp のコードが書いてあれば, SKK はそのコードを Lisp のプログラムとして実行し, その結果の文字列を画面に挿入する. 例えば, 辞書に

```
now /(current-time-string)/
```

という行があるとき, 『/now 』とタイプすれば画面には現在の時刻が表示され, 【▼ Fri Apr 10 11:41:43 1992】のようになる. このような項目の登録は通常の辞書登録により行うことができる.

ここで使える Lisp のコードは改行を含んでいないものに限られる. またこのコードは結果として文字列を返すようなものでないといけない.

の文字がコードの順に表示される．これを第 2 段階のメニューという．例えば，第 1 段階のメニューが上のようになっているときに，『d』をタイプすると第 2 段階のメニューは以下のようになる．

A: ～ S: ‖ D: | F: … G: ‥ H: ‘ J: ’ K: “ L: ” Q: (W:) E: [R:] T: [Y:] U: {

第 2 段階のメニュー中のキーを (小文字) でタイプすると，対応する文字がバッファ挿入されて，メニューによる入力が終る．第 2 段階のメニューが表示されているときも，『SPC』と『x』で第 2 段階のメニューを前進，後退させることができる．また『<』，『>』によりメニューを 1 文字分だけずらすことができる．例えば，第 2 段階のメニューが上のようになっているときに，『<』をタイプすると，メニューは，

A: ＼ S: ～ D: ‖ F: | G: … H: ‥ J: ‘ K: ’ L: “ Q: ” W: (E:) R: [T:] Y: [U:]

となる．さらに，第 2 段階のメニューが表示されているときに『?』をタイプすると，そのときのキー【A】に対応する文字のコードが表示される．

3.10 読みのコンプリーション

「▽モード」で TABキーを押すと，「読みのコンプリーション」が行われる．TABを押す迄に「▽モード」で入力された文字列を σ とすると，送り仮名なしの辞書バッファの中で，先頭が σ と一致し，長さが σ よりも長い読みを捜して，そのような τ がもしあれば，σ の代わりに τ が表示される．

コンプリーションは辞書バッファを最初から順に調べて行われるので，文字列 σ と先頭が一致する読みがバッファに複数個あるときは前の方にある読みにコンプリーションされる．これが意図したものでなかったときには，TABの直後に『.』(period) をタイプすると 2 番目の読みが表示される．以下同様に，『.』を続けてタイプすると，読みの候補が順次表示される．『.』を打ちすぎて意図した読みを通りすぎたときは『,』(comma) で前の候補に戻る．

Shall I register this as okuri-ari entry: ありがs / 有賀 / ? (y or n)

と聞かれたときに『n』をタイプすればよい. この結果, 送り仮名なしの辞書バッファは

ありがさつき / 有賀さつき /

となる.

3.9 コードまたはメニューによる入力

第2水準の漢字等は, JIS または EUC コードによる入力が可能である. コードによる入力は, 『\』キーによる. 例えば, 「卍」の JIS コードは 5244なので,

\5244 RET

とタイプすれば, 【卍】が入力される.

コードが不明の文字の入力には2段階メニューを用いた入力が可能である. この場合も『\』キーをタイプすると, ミニバッファに

JIS or EUC code (00nn or CR for Jump Menu):

というプロンプトが出るので, 『RET』キーをタイプすると, 2段階メニューの第1段階のメニューが次のように表示される.

A: S: ¯ D: 〜 F:} G: = H: ¢ Q: ◆ W: E: ∩ R: T: ≡ Y:

第1段階のメニューでは JIS 漢字をコードの順に 16 文字ごとに 12 文字表示する. 上の例では【A: 】として JIS コード 2121の全角スペースが表示され, 以下 2131, 2141, ... となる. ここで『SPC』をタイプすると次のメニューが表示される. このとき表示されるのは前のメニューより, コードの値が 16x12 = 192 ずつ大きな文字である. 『x』により前のメニューに戻るのは変換のときと同様である. メニューのキー『a, s, d, f, g, h, q, w, e, r, t, y』のどれかを (小文字で) タイプすれば, そのキーに対応する文字から始まる 16 個

となっていれば, 今回はブロックが 3 つあるので「わたす」, 「わたさ」, 「わたし」について「わたした. 」の先頭と一致するかのチェックが行われ, 画面は【▼渡した. 】となる.

また, 仮に送り仮名ありの辞書バッファでの検索がすべて失敗するか, 辞書バッファでの候補がなくなるまでスペースをタイプすると, (辞書サーバを使用するような設定になっているときは) 辞書サーバに見出し語「おもいます」の検索を依頼する. 辞書サーバは見出し語「おもいます」を単純に検索するだけで, 送り仮名ありの辞書バッファでの検索のような処理はしない.

もう一度画面が【▽わたした. 】のときにスペースをタイプした場合を考えてみよう. ただし, 今回は送り仮名ありの辞書バッファが

```
わた s /渡/
```

となっているとする. このときは対応するブロックがひとつもないのでこの辞書バッファでの変換は失敗する. また辞書サーバを使用する設定になっていても (通常は) 検索に失敗するので辞書登録モードになりミニバッファに【わたした. 】というプロンプトがでる. そこで, 『WataSita.』とタイプすればミニバッファに【渡した. 】ができるのでリターンキーをタイプする. このとき辞書への登録がされるが, 今の場合のように登録する語の最後が平仮名で終るときは, その最後の平仮名の文字列 (今の場合「した」) が見出し語の最後と一致するかを調べて, 一致する場合にはシステムが, 辞書の登録を送り仮名のある辞書バッファにするのか送り仮名のない辞書バッファにするのかをユーザに質問をする. 今の例については

```
Shall I register this as okuri-ari entry: わた s /渡/ ? (y or n)
```

という形式で質問するので『y』をタイプすればよい. この結果, 送り仮名ありの辞書バッファは

```
わた s /渡/[し/渡/]/
```

となる.

【▽ありがさつき】を【▼有賀さつき】と変換できるようにしたければ, 上と同様に辞書登録をし,

3.8 送り仮名の自動処理

変数 skk-auto-okuri-processを tにしておくと (see 4 [カスタム化], page 33) 文法的な知識を用いない送り仮名の自動処理がされるようになる. これは前の節で述べた基本的な方法のように送り仮名の最初のローマ字表現を大文字で始める必要がないという利点があるが, 同時にユーザがシステムに与える情報が少なくなるので意図しない変換をされる割合が増えるという欠点がある. どちらの使い方をするかはユーザの好みの問題である.

このように設定されているときには▽モードでスペースをタイプしたときの処理は以下のようになる. 例えば, 画面が【▽おもいます】のときにスペースをタイプすると, SKK はまず送り仮名なしの辞書バッファで見出し語「おもいます」を検索する. 通常はそのような見出し語はないので, 次に送り仮名ありの辞書バッファで, 見出し語の先頭の何文字かが「おもいま」, 「おもい」, 「おも」, 「お」のどれかと一致するものをこの順に, 見つかるまで検索する. 通常最初のふたつについては一致するものがバッファにないが, 「おも」については, 例えば,

```
おもｉ /思/重/[い/思/重/]/
```

のようになっていると, これの先頭が「おも」と一致する. このとき, 対応する変換の候補中の「[」と「]」で区切られたブロックの先頭の文字 (今の場合, そのようなブロックはひとつしかなく「い」になる) を「おも」の後に追加した文字列「おもい」が変換の対象の文字列「おもいます」の先頭と一致するかどうかをチェックする. この例では確かにそうなっているので, そのブロックにある候補「思」と「重」を用いて, 【▼思います】(ここでもう一度スペースをタイプすると) 【▼重います】と変換する.

もうひとつの例として, 画面が【▽わたした. 】のときにスペースをタイプしたとする. このときは, 送り仮名なしのバッファで見出し語が「わたした. 」となるものは (通常) ないので, 送り仮名ありのバッファで検索をする. このときバッファが

```
わたｓ /渡/[す/渡/]/[さ/渡/]/[し/渡/]/
```

動きたいけど動けない.

となる. もし, 辞書に「うご k」が登録されていなければ, 『UgoKi』までタイプすれば, 『i』をタイプした時点で, 変換中モードになり, 画面は

▼うごき

となり, ミニバッファは

うご * き

となる. そこで, ミニバッファに【動】を作ればよい. (ミニバッファに【動き】を作ってしまうと, 以後, 『UgoKi』は【動きき】と変換されてしまうので注意が必要である.)

注意: 名詞の後に「する」がついて動詞になることがよくあるが, この場合は「する」を送り仮名としないほうが効率がよい. 例えば, 【運動する】は, 『Undou suru』により得られる. 名詞から作られる形容詞等も同様である.

送り仮名のある単語の変換と, それに対応する辞書の内容の変化を, もう少し詳しくみてみよう. 辞書が,

おお k /多/大/

となっているとすると, これは, 「おお」の後につく送り仮名の最初の文字のローマ字表現が『k』である単語の変換に用いられ, そのときの第一候補が「多」で, 第二候補が「大」であることを示している. したがって, このとき, 『OoKii』とタイプすれば, 画面は, 【▼多きい】となり, さらにスペースをタイプすると【▼大きい】となる. ここで, 確定すると, 辞書は,

おお k /大/多/[き/大/]/

となる. 次に『OoKu』とタイプすると, 第一候補の「大」が表示され, さらにスペースをタイプして, 確定すると画面は, 【多く】となり, 辞書は,

おお k /多/大/[き/大/]/[く/多/]/

となる. 以後は, いつでも, 『OoKii』は【大きい】となり, 『OoKu』は【多く】と変換されるようになる.

3.4 カタカナの入力

カタカナは, すでに述べたように, カナモードで入力することができるが, この他▽モードを用いた入力も可能である. すなわち, ▽モードで『q』をタイプすれば【▽】からカーソルの位置までのひらがなの文字列がカタカナに変換され, モードは平仮名確定入力に戻る.

3.5 英字変換

『file』を【ファイル】と展開させたければ, 『/file 』とタイプすれば画面は

▼ file

となり, 同時にミニバッファに

file

というプロンプトがでるので, ミニバッファで【ファイル】を作ればよい. 以後, 『/file 』により【ファイル】が得られる.

3.6 全英文字の入力

全英文字の入力についても上で述べたカタカナの入力と同様の方法がある. 平仮名確定入力モードで全英文字の単語, 例えば, 「ｆｉｌｅ」を画面につくりたいときは『/file』とタイプした後で『C-q』をタイプすればよい.

3.7 送り仮名のある単語の入力

送り仮名の最初の仮名を大文字で始める. 例えば, 【動く】を入力するには, 『UgoKu』とタイプすればよい. 辞書には,

うごｋ /動/

と登録されているので, 『UgoKitaikedoUgoKenai.』とタイプすれば,

変換中

となる. (辞書登録モードも終了する.) このように, 通常は『C-j』をタイプしなくても自然に確定するようにできている.

副作用として確定をともなうキーは印字可能な文字すべてと『RET』である. ただし, see 4.10 [skk-kakutei-early[t]], page 42.

変換中に変換の候補が多いときは, スペースを続けて, 求める候補を通過してしまうことがある. このときは『x』により前候補に戻ることができる.

なお, ミニバッファは再帰的に使うことができるので, 辞書登録も再帰的にできる.

誤まって登録した単語の削除は, 削除したい単語を変換により求め, その単語が得られた時点で, 『X』を入力すればよい. ただし, 送り仮名のある単語については, 削除は以下のように行われる.

仮に, 辞書が,

おおｋ /多/大/[く/多/]/[き/大/多/]/

となっているとき, 【多き】をパージすると, 辞書は,

おおｋ /多/大/[く/多/]/[き/大/]/

となる. 次に【多く】をパージすると, 辞書は

おおｋ /大/[き/大/]/

となる. さらに【大き】をパージすると, 辞書から, 「おおｋ」というエントリーが削除される. (辞書の形式に関する説明は以下にもある.)

誤まって辞書登録モードに入ったときは, 『C-g』により辞書登録モードから抜けることができる. あるいは, 何も登録しないで『RET』キーをタイプしてもよい.

がたくさんあるので, 通常は『C-j』を入力する必要は無い.) このとき【▼】も画面から消え, 変換が確定したことを知らせる. 求めるものが【漢字】でなければ, スペースを入力すれば辞書の中の次候補が検索され,

▼幹事

となる.

候補が 5 つ以上あるときは, 5 番目以降の候補は, 7 つずつまとめてミニバッファに表示される. 例えば, 辞書が

きょ /距/巨/居/裾/嘘/拒/拠/虚/挙/許/渠/据/去/

となっているとき, 『Kyo』の後にスペースを 5 回続けて打てば,

A:嘘 S:拒 D:拠 F:虚 J:挙 K:許 L:渠 [残り 2]

がミニバッファに表示される. ここで, 仮に【許】を選択したければ, (小文字の) 『k』を入力すればよい.

候補を次々と探しても求めるものがなければ, 自動的に辞書登録モードになる. (辞書登録モードは▼モードのサブモードである.)

例えば, 辞書に「変換中」がない場合, 『Henkantyuu 』(最後はスペース) とタイプすれば, 画面は

▼へんかんちゅう

となり, 同時に, ミニバッファに【へんかんちゅう】というプロンプトが表示される. 辞書登録モードでは入力はミニバッファに対して行われる.

したがって, 辞書に「変換」と「中」があれば, ミニバッファの中で, 『Henkan Tyuu 』とタイプすれば, ミニバッファは

変換▼中

となる. ここでリターンキーを押せば, 「変換中」が辞書に登録され, 同時に画面は

と表示される. 【▽】は▽モードであることを示し, また同時に, 変換開始点も示している. 『K』を大文字にするのを忘れて, 『kanji』と入力すれば, 画面は【かんじ】となるが, この時点で, カーソルを【か】に戻して, 『Q』をタイプすれば, 【か】の前に【▽】が入りカーソルの位置を変換開始点とする▽モードになる.

誤って▽モードに入ってしまったときには次のふたとおりの処理の仕方がある. 今, ▽モードで, 画面が【▽かんじ】となっているする. ここで『C-j』をタイプすれば▽モードから抜けて平仮名確定入力モードとなり画面は【かんじ】となる. また, 『C-g』をタイプすれば【▽かんじ】が全部消去され平仮名確定入力モードになる. これは, この後で述べる英字変換 (see 3.5 [英字変換], page 19) のときも同様である.

3.3.2.3 ▼モード

このモードでは, 漢字変換の対象となる文字列は確定しており, この文字列の変換を行う. ▽モードから, ▼モードへ移る方法は 2 通りある. 最初に, 『 』(スペースキー) の入力による方法を説明する. ▽モードでスペースキーをタイプすれば, 【▽】からカーソルの位置までの文字列が変換の対象となる文字列 σ として確定する.

このとき, 【▽】は【▼】で置き換えられ, これにより▼モードになったことが分かる.

また, 同時に文字列 σ が辞書の中で検索される.

前の例で, 『Kanji』に続けてスペースを入力すれば, 【かんじ】が変換の対象として確定し, 辞書の中で検索される. 辞書の中に【かんじ】で始まる行があれば, そのような最初の行が選択される. 仮に, 辞書が

かんじ /漢字/幹事/

となっているとすれば, 「漢字」が最初の候補になり, 画面は :

▼漢字

となる. もしこれが求めるものであれば, この時点で『C-j』をタイプすれば変換が確定する. (しかし, 後で述べるように, 副作用として確定をともなうキー

3.3.2 変換モード

変換モードは以下のどれか:

1. 平仮名確定入力モード
2. ▽モード
3. ▼モード

これらのモードは, 通常は

$$1 \Longrightarrow 2 \Longrightarrow 3 \Longrightarrow 1 \Longrightarrow 2 \Longrightarrow 3 \Longrightarrow \cdots$$

と推移する.

3.3.2.1 平仮名確定入力モード

平仮名確定入力モードでは, 漢字変換は行われないが, 入力モードに応じてローマ字かな変換等は行なわれる.

3.3.2.2 ▽モード

▽モードは, 漢字変換の対象となる文字列を入力するモードである. 平仮名確定入力モードから▽モードへ移る方法は何通りかあるが, 通常はローマ字入力を大文字で始めることにより, ▽モードになる.

例えば,

```
Kanji
```

とタイプすれば, 画面には

▽かんじ

カタカナに変換してバッファに取り込む．アスキー文字入力モードでは入力した文字がそのまま (半角文字として) 挿入される．アスキー文字入力モードは，`skk-mode`を起動したときのモードとほぼ一致する．(ただし，アスキー文字入力モードでは『`C-j`』は後で説明する意味を持ち，モードラインの表示も異なる) また，全角英字モードでは，英数字が全角で入力される．

例えば，ローマ字ひらがな入力モードで，

`kon'nitiha` (または，`konnnitiha`)

とタイプすれば，

こんにちは

と変換されて入力される．かなの小文字を単独で入力したいときは，『`x`』の後に小文字にしたいかなのローマ字表現を入力すればよい．例えば，『`xya`』で【ゃ】を入力できる．

ローマ字ひらがな入力モードのとき，mode line の先頭には【かな】と表示される．ローマ字カタカナ入力モードでは，【カナ】と表示される．全角英字入力モードでは，【全英】と表示され，アスキー文字入力モードでは【SKK:】と表示される．

これらのモードの間の推移は以下のように行われる．

a 「かな」，「カナ」間の推移は『`q`』キーによる．

b 「かな」または「カナ」から「SKK」へは『`l`』(latin の頭文字) キーによる．

c 「かな」または「カナ」から「全英」へは『`L`』キーによる．

d 「SKK」または「全英」から「かな」または「カナ」へは『`C-j`』．

実際にはカナモードや全英モードで長時間入力を続けることはほとんどないので，かなモードのままでカナ文字や全英文字を入力する便法が用意されている．これについては，see 3.4 [カタカナの入力], page 19, see 3.6 [全英文字の入力], page 19.

また『C-x j』をタイプすれば, SKK が自動右揃えモード (skk-auto-fill-mode) で起動される (see 3.14 [自動右揃え], page 28).

SKK は overwrite モード, picture モードにも自動的に対応し日本語の入力をサポートする.

SKK は最初に起動されたときにユーザ辞書ファイル (通常は '.skk-jisyo') を読み込み, 送り仮名のある単語からなるバッファと送り仮名のない単語からなるバッファを作成する.

3.3 モード

SKK が起動されている状態のことを SKK モードという. SKK は Emacs の他のメジャーモードやマイナーモードと共存可能なように, これらと直交する形の設計になっている. すなわち, SKK をあるバッファで起動してもそのバッファのメジャーモードやマイナーモードに変更はない. したがって, SKK 起動時のフックを利用して, SKK をバッファのモードに応じてカスタム化することが可能である (see 4 [カスタム化], page 33).

SKK モードは 4 つの入力モードと 3 つの変換モードの組により決定される. 入力モードが何であるかはモードラインに表示され, 変換モードはバッファ中に表示される.

3.3.1 入力モード

入力モードは以下のどれか:

1. ローマ字ひらがな入力モード
2. ローマ字カタカナ入力モード
3. アスキー入力モード
4. 全角英字入力モード

ローマ字ひらがな入力モードでは, ローマ字による入力をひらがなに変換してバッファに取り込む. 同様に, ローマ字カタカナ入力モードでは, ローマ字を

3.1.4 skk-record

SKK は最初に使用したときに, '.skk-record'をホームディレクトリに作る. '.skk-record'には SKK の使用に関する統計がとられる.

3.1.5 SKK チュートリアル

このマニュアルの他に SKK にはチュートリアルプログラムも用意されている. チュートリアルは『C-x t』をタイプすれば起動される. 英語版のチュートリアルテキストを用いたいときは, '.emacs'で

```
(setq skk-tut-file (concat exec-directory "SKK.tut.E")
```

と設定すればよい. 同様にプライベートな場所に置いてあるチュートリアルテキストを用いるようにもできる. また,

```
(setq skk-file ".../skk.elc")
```

等により, チュートリアルで用いる SKK を指定することもできる.

3.1.6 ローマ字・かな対応表

また, SKK でのローマ字・かな対応表が NTT jTeX のファイル 'skk-romkan.tex'として用意されている.

3.2 起動と終了

SKK は『C-x C-j』とタイプすることにより起動される. あるバッファで SKK が起動されているかどうかは, 次項で述べるように, モードラインを見ることにより知ることができる. SKK が起動されているバッファで『C-x C-j』をタイプすれば SKK を終了し, 起動する前の状態に戻る. しかし, 後で述べるように (see 3.3.1 [入力モード], page 13) , SKK モード中の「アスキー文字入力モード」と起動する前の状態はほとんど同じなので, SKK を終了する必要があることはほとんどない.

```
(autoload 'skk-tutorial "skk-tut" nil t)
(autoload 'skk-isearch-forward "skk-isearch" nil t)
(autoload 'skk-isearch-backward "skk-isearch" nil t)
(autoload 'skk-check-jisyo "skk-tools" nil t)
(autoload 'skk-merge "skk-tools" nil t)
(autoload 'skk-diff "skk-tools" nil t)
```

注意: Demacs の場合は Emacs の環境設定用ファイルは '_emacs' である. また, DOS でのファイル名の制限があるために,

```
(setq skk-init-file "~/_skk")
```

とする必要がある.

3.1.3 .skk の設定

SKK の使用形態のカスタム化のための設定はすべて, ユーザのホームディレクトリにあるファイル '.skk' で行われる. とりあえず必要な設定は共用辞書の利用に関するものだけであり, これは以下のようにすればよい.

Emacs のバッファに共用辞書を読み込んで利用する場合は, '.skk' に以下を追加する.

```
(setq skk-large-jisyo ".../SKK-JISYO.L")
```

辞書サーバを用いて共用辞書を利用する場合は, '.skk' に以下の例のような行を追加する.

```
(setq skk-server-host "mars")
(setq skk-serv "/usr/local/soft/nemacs/etc/skkserv")
(setq skk-aux-large-jisyo ".../SKK-JISYO.L")
```

変数 skk-aux-large-jisyo はサーバが利用できなくなったときに用いる共用辞書ファイルを指定するものである.

欠点は変換の速度が遅いこととサーバがダウンしているときには使えないということである.

変換の速度については, 機種に依存するが, 最近のワークステーションでは (イ) の方が (ロ) よりも数十倍速いという実験結果がある. しかし, サーバを利用した場合も変換の速度が気になることはない. また (ロ) の方式で利用している場合, サーバが使えなくなると自動的に (イ) の方式に移行するようになっている.

辞書については, see 6 [SKK 辞書とユーティリティプログラム], page 53, により詳しい解説があるので, ある程度 SKK に慣れてから参照してほしい.

3.1.2 .emacs の設定

'.emacs'ファイルに以下の行をいれる. (この部分はファイル'skk.tex'からとってくればよい.)

注意: System V 系の Unix の一部にはファイル名の長さの上限が 14 というものがある. そのようなシステムでは'skk-isearch.elc'という名前のファイルがつくれないので, SKK のインストールの際に's-isearch.el'と's-isearch.elc'を代りに用いているはずである. その場合には, 以下の "skk-isearch" を "s-isearch" とする必要がある. また, Demacs については DOS でのファイル名の制限があるために, いくつかのファイル名を変更する必要がある.

また, C-x j と C-x J は Emacs では本来コマンド register-to-point に割り当てられているので, skk-auto-fill-mode との使用の割合を考えてキーをセットすればよい.

```
(global-set-key "\C-x\C-j" 'skk-mode)
(global-set-key "\C-xj" 'skk-auto-fill-mode)
(global-set-key "\C-xJ" 'register-to-point)
(global-set-key "\C-xt" 'skk-tutorial)
(global-set-key "\C-s" 'skk-isearch-forward)
(global-set-key "\C-r" 'skk-isearch-backward)
(autoload 'skk-mode "skk" nil t)
(autoload 'skk-auto-fill-mode "skk" nil t)
```

第 3 章

SKK の使用法

3.1 準備

3.1.1 辞書

SKK version 5 までは辞書の利用形態は (イ) ユーザ辞書のみを用いるか (ロ) ユーザ辞書と共用辞書を併用するかのいずれかであった. version 6 以降ではより一般的な辞書の利用形態を採用しており, ユーザが利用する辞書を自由に指定し, それらの辞書にアクセスする順序も自由に設定できるようになっている. また大規模な共用辞書を Emacs のバッファに読み込んで利用することも可能になっている.

したがって, 基本的な利用形態は, いくつかのユーザ辞書といくつかの共用辞書を使用することになる. 共用辞書の使用形態には以下の 2 種類がある.

(イ) Emacs のバッファに取り込んで利用する.

(ロ) 辞書サーバによりアクセスする.

(イ) の方式は, 最初にアクセスするときにファイルをバッファに読み込むのに少し時間がかかることと, Emacs の使用するメモリが大きくなるという欠点がある. 一方, この方式の長所は変換の速度が速いことと, サーバに依存しないために Emacs が動いている限りいつでも使用できるということである.

(ロ) の方式は (イ) と相補的な関係にある. したがって, その長所は, 1 回目のアクセスが速いことと, Emacs の使用するメモリを増加させないことであり,

[6] 辞書サーバのインストール. (see 5 [辞書サーバ (skkserv)], page 47 for more info)

```
cd skkserv
Makefile を編集する.
(普通は, ETCDIR の値を $NEMACS/etc にするだけでよい. )
make
make install
```

2.3 その他 (インストールについて)

プログラミング言語 Perl が使えるサイトでは 'skktools'の下にあるパッケージをインストールすれば, SKK 辞書関連のユーティリティプログラムが使用可能になる (see 6 [SKK 辞書とユーティリティプログラム], page 53).

'SKK-JISYO.JIS2'は JIS 第 2 水準の漢字を部首の読みで検索できるようにした辞書である. この辞書を利用したいサイトは 'SKK-JISYO.L'と 'SKK-JISYO.JIS2'を skk-mergeを用いてマージした結果を共用辞書とすればよい (see 6.3.1 [Emacs Lisp ユーティリティプログラム], page 56).

ユーザごとの初期設定に関しては see 3.1 [準備], page 9.

2.2 インストール

配布されたパッケージのあるディレクトリでコマンドConfigureを実行し, システムからの質問に答えればインストールできる. その際, 特権ユーザになることが可能ならばrootになってからコマンドを実行する. なお, Configureの代わりに, Configure.euc を実行すると, メッセージが日本語 (EUC コード) になる.

また特権ユーザになることができなくて, SKK を個人の環境で利用する場合は辞書サーバのインストールは不要である.

インストールの詳細は以下の通りである.

[1] 'SKK*'とマッチするファイルすべてと'emacs-sort'を'$NEMACS/etc'にコピーする. ('$NEMACS' は Nemacs の distribution のあるディレクトリ.)

[2] '*.el'とマッチするファイルをすべてバイトコンパイルして, 対応する'.elc'ファイルをつくる.

[3] '*.el'または'*.elc'とマッチするファイルをすべて'$NEMACS/lisp'にコピーする. このときファイル名の長さが 14 文字までという制限のあるシステムでは's-isearch.el'と's-isearch.elc'をコピーする. たいていのシステムではこのような制限はないので, これらのファイルの代りに'skk-isearch.el'と'skk-isearch.elc'をコピーする. 'skk-isearch'と's-isearch'は名前が違うだけで内容は同じである. OS によってはさらに制限がきつい可能性があるが, その場合は適宜対応してほしい.

[4] 'latexinfo.sty', 'skk.tex', 'tabular.sty', 'skk.ps'を'$NEMACS/man'にコピーする.

[5] 'skk'を'$NEMACS/info'にコピーする. さらに, '$NEMACS/info/info'をエディットし, メニューに以下の行を加える.

```
* SKK: (skk).     Nemacs のための日本語入力システム.
```

```
latexinfo.sty
s-isearch.el
skk
skk-isearch.el
skk-romkan.tex
skk-tools.el
skk-tut.el
skk.el
skk.ps
skk.tex
skkserv/Makefile
skkserv/README
skkserv/add-rc.local
skkserv/add-services
skkserv/skkserv.c
skkserv/skkserv.h
skktools/Makefile
skktools/README
skktools/adddummy.t
skktools/codeconv.pl
skktools/convprog.t
skktools/kana2roma.pl
skktools/list2skk.t
skktools/pubdic2list.t
skktools/removedummy.t
skktools/roma2kana.pl
skktools/skk2list.t
skktools/skkconv.t
skktools/sub.t
tabular.sty
```

第 2 章

SKK のインストールの方法

すでにシステムに SKK がインストールされている場合, 一般のユーザはこの章を読む必要はない. しかし, その場合でも, この章に目を通すことにより SKK の構成についての知識を得ることができる.

2.1 パッケージの確認

SKK のパッケージは以下のファイルからなる.

```
COPYING
Configure
Configure.euc
Makefile
ReadMe
ReadMe.English
SKK-JISYO.JIS2
SKK-JISYO.L
SKK-JISYO.M
SKK-JISYO.S
SKK.tut
SKK.tut.E
emacs-sort
```

1.2.4 再帰的辞書登録

辞書に無い項目については自動的に辞書登録のモードになるので, 辞書に登録する読みをユーザが入力する必要がない. また, 再帰的な辞書登録もできる.

- 日本語文法についての知識を用いないで変換のアルゴリズムをつくる.
- 日本語入力モードでも最小のキーストロークで英字や全角英字が入力できること.
- 日本語入力モードでも Emacs のたいていのコマンドはそのまま使えること.

1.2 SKK の特徴

SKK は以下のような特徴をもつ.

1.2.1 ユーザ辞書と共用辞書によるかな漢字変換

変換はユーザ辞書と共用辞書を用いて行われる. 共用辞書へのアクセスの方式は Emacs のバッファに読み込んで利用する方式と辞書サーバに依存する方式の 2 つがある.

SKK バージョン 6 以降では共用辞書をバッファに読み込み, 高速にアクセスする方式が実現されている. これにより, SKK を Emacs の中で完全に閉じた形で安定して使えるようになっている.

1.2.2 文体に依存しないかな漢字変換

変換の単位を小さくおさえることにより, 文法的知識を用いないで, 効率的なかな漢字変換を行なう. SKK での変換単位は通常 1 単語である. 変換のアルゴリズムが文法的な知識を用いないので, 古語, 現代語, 方言, 文語, 口語等, いずれについても同様に効率的な変換ができる.

1.2.3 Nemacs との高い親和性

SKK での日本語入力は, ローマ字でまず入力しそれをかなに変換し, それをさらに適当なコマンドを用いて漢字に変換する形でなされる. 日本語入力のモードでも大部分の Emacs コマンドはそのまま使えるようになっている. 実際, 『C-j』 (control-j) 以外の control key はすべて Emacs のコマンドとして解釈される.

第 1 章

SKK とは

SKK は Nemacs, Mule および Demacs の上に高速で効率的な日本語入力環境を提供するシステムであり, GNU General Public License に従ったフリー・ソフトウェアとして配布される.

SKK は

1. いくつかの emacs lisp プログラム
2. 辞書サーバ用の C プログラム
3. SKK 辞書
4. ユーティリティプログラム

から構成される. (1) は佐藤が作成し, (2) は亀山が作成した. (3) の SKK 辞書は多数の SKK ユーザから提供された辞書をもとに作成したものである. また, (4) はシャープの増井俊之氏により提供されたものである.

本マニュアルでは SKK バージョン 6.32 について, 設計方針, 特徴, インストールの方法および使用法を中心にして述べる.

1.1 SKK の設計方針

SKK は以下のような設計方針でつくられている.

- ユーザの意図した日本語の文字列がそのまま容易に入力できること.

6.3 辞書ユーティリティプログラム 56
6.3.1 Emacs Lisp ユーティリティプログラム 56
6.3.2 Perl によるユーティリティプログラム 58

7 旧バージョンとの相違点 63
7.1 SKK2.25 との相違点 63
7.2 SKK3.47 との相違点 63
7.3 SKK4.33 との相違点 64
7.4 SKK5.30 との相違点 65

A SKK コマンド一覧 67

B SKK メッセージ一覧 73

C SKK メイリングリスト 79

D SKK の入手方法 81
D.1 メールによる入手方法 81
D.2 Anonymous Ftp による入手方法 82
D.3 その他の入手方法 82

E SKK 関連ソフトウェア 83
E.1 ari 83
E.2 kakasi 86
E.3 skkfep 87

F 変数および関数索引 89

G 事項索引 93

4.16 skk-hirakana-mode-string["-- かな "] 43
4.17 skk-katakana-mode-string["-- カナ "] 43
4.18 skk-zenei-mode-string["-- 全英 "] 43
4.19 skk-server-host 44
4.20 skk-serv 44
4.21 skk-num-type-list 44
4.22 skk-numeric-conversion[t] 44
4.23 skk-backup-jisyo 45
4.24 skk-dabberv-like-completion[nil] 45
4.25 skk-kakutei-key 45
4.26 skk-search-prog-list 45
4.27 skk-kakutei-jisyo 45
4.28 skk-initial-search-jisyo 46
4.29 skk-large-jisyo 46
4.30 skk-aux-large-jisyo 46

5 辞書サーバ (skkserv) 47

5.1 skkserv とは 47
5.2 skkserv のインストール方法 47
5.2.1 スーパーユーザの権限を持つ人のインストール方法 . . . 47
5.2.2 スーパーユーザの権限を持たない人のインストール方法 . 49
5.3 skkserv の起動 49
5.4 SKK 形式の辞書 50
5.5 その他 (辞書サーバについて) 51
5.5.1 skkserv が動作する環境 51
5.5.2 テスト方法 51
5.5.3 限界・改良の余地 52

6 SKK 辞書とユーティリティプログラム 53

6.1 SKK 辞書の形式 53
6.2 辞書バッファのセーブ 56

3.3.2 変換モード 15
3.4 カタカナの入力 19
3.5 英字変換 19
3.6 全英文字の入力 19
3.7 送り仮名のある単語の入力 19
3.8 送り仮名の自動処理 21
3.9 コードまたはメニューによる入力 23
3.10 読みのコンプリーション 24
3.11 日付の入力 25
3.12 プログラム実行変換 25
3.13 数をパラメータとする表現の入力 26
3.14 自動右揃え 28
3.15 インクリメンタル・サーチ 29
3.16 文字コードを知る方法 30

4 カスタム化 **33**

4.1 skk-input-vector, skk-zenkaku-vector 33
4.2 skk-prefix-list, skk-roma-kana-[aiueo] 37
4.3 skk-rom-kana-rule-list 38
4.4 skk-echo[t] 40
4.5 skk-keep-record[t] 40
4.6 skk-use-vip[nil] 41
4.7 skk-process-okuri-early[nil] 41
4.8 skk-auto-okuri-process[nil] 42
4.9 skk-egg-like-newline[nil] 42
4.10 skk-kakutei-early[t] 42
4.11 skk-jisyo 42
4.12 skk-record-file 43
4.13 skk-jisyo-code[nil] 43
4.14 skk-date-ad[nil], skk-number-style[1] 43
4.15 skk-mode-string["--SKK:"] 43

もくじ

1 SKK とは **1**

1.1 SKK の設計方針 1

1.2 SKK の特徴 2

1.2.1 ユーザ辞書と共用辞書によるかな漢字変換 2

1.2.2 文体に依存しないかな漢字変換 2

1.2.3 Nemacs との高い親和性 2

1.2.4 再帰的辞書登録 3

2 SKK のインストールの方法 **5**

2.1 パッケージの確認 5

2.2 インストール 7

2.3 その他 (インストールについて) 8

3 SKK の使用法 **9**

3.1 準備 9

3.1.1 辞書 9

3.1.2 .emacs の設定 10

3.1.3 .skk の設定 11

3.1.4 skk-record 12

3.1.5 SKK チュートリアル 12

3.1.6 ローマ字・かな対応表 12

3.2 起動と終了 12

3.3 モード 13

3.3.1 入力モード 13

Copyright © 1991, 1992, 1993 Masahiko Sato and Yukiyoshi Kameyama

This is the manual for SKK Version 6.32.

Permission is granted to make and distribute verbatim copies of this manual provided the copyright notice and this permission notice are preserved on all copies.

Permission is granted to copy and distribute modified versions of this manual under the conditions for verbatim copying, provided that the entire resulting derived work is distributed under the terms of a permission notice identical to this one.

Permission is granted to copy and distribute translations of this manual into another language, under the above conditions for modified versions, except that this permission notice may be stated in a translation approved by Masahiko Sato and Yukiyoshi Kameyama.

日本語入力システム SKK

(Version 6.32)

東北大学電気通信研究所 佐藤 雅彦
(masahiko@sato.riec.tohoku.ac.jp)

東北大学電気通信研究所 亀山 幸義
(kam@sato.riec.tohoku.ac.jp)

1993 年 4 月 24 日