JUS Tコード合宿 (高知) 2002年9月15・16日

Tコード入力環境tc2の活用

北嶋 暁

 2

目次

- 私とTコード
- Tコード入力環境tc2使いこなし術
- Tコードの今後

 3

はじめに -私とTコード-

- なぜTコードを始めたか?
  - かな漢字変換への不満
- Tコードの練習
  - ひたすらEELLL
  - Tコードは右利き用?
- Tコード入力環境の開発
  - はじめは改良→パッチ
  - いろいろ直しているうちにメンテナー

 4

使って分かったTコードの良さ・悪さ

- 入力のストレスが軽減される。
  - ただし...
    - 打ち間違いが多い気がする。
    - ど忘れしたとき困る。
    - 字を間違えることがある。
- コードをある程度覚えれば実用的に使える。
  - ただし...
    - たくさん、しっかり覚えた方が効率がよい。

 5

基本的な入力

- Tコード入力 (40個のキーによる入力)
- Tコードで使用するキー以外での入力
- シフト時の入力
- スペースキーを用いた入力

 6

表(コード)の変更

- 【関数】tcode-set-action-to-table
- 【変数】tcode-after-load-table-hook
- 例

```
(add-hook 'tcode-after-load-table-hook
          (lambda ()
            ;; 2ストローク以上のキーの変更
            (cond ((eq tcode-input-method 'tcode)
                   ;; ○自分の名前の字は2ストロークで。
                   ;; 硝→嶋
                   (tcode-set-action-to-table '(31 5) "嶋"))))
```

 7

表を変更する際の注意

- 使わない字の代わりに使う字を入れる。
  - デフォールトのコードをそのまま使う必要はない。
  - 空いている部分は打ちにくい。
- 左利きの場合は左右反転?

 8

キーの番号

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39

 9

【余談】キーシーケンスをどう呼ぶか?

- アルファベット
  - 連想式になり、よくない?
  - 普遍性がない。
    - (Qwerty/Dvorakなどの違い)
- かなの割り当て(橋田さん)

  げごぐぎが かきくけこ
  ぞぜずじざ さしすせそ
  どでづぢだ たちつてと
  ぼべぶびば はひふへほ

 10

ヘルプ表の表示法

- キャラクタベース

- ビットマップベース・文字列
  - (setq tcode-help-with-real-keys t)
  - 変数`tcode-stroke-to-string-option'の設定

 11

その他のキーの設定

- 【関数】set-tcode-mode-key

```
(add-hook 'tcode-after-load-table-hook
          (lambda ()
            ;; 1ストロークのキーの変更
            (cond ((eq tcode-input-method 'tcode)
                   (set-tcode-mode-key ?- 'tcode-insert-ya-outset))))
```

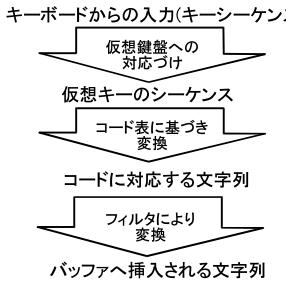
 12

シフト時の動作

- アルファベットの小文字を入力する。
- アルファベットの大文字を入力する。
- 変換フィルタを適用する。
  - SKK風の前置型交ぜ書き変換
  - ひらがなをカタカナに
  - 異体字などへの変換
    (自分で関数を作れば)

 13

Tコード入力時の処理流れ



 14

フィルタの設定

- 変数 `tcode-input-filter-functions'
  - 適用条件とフィルタ関数との組のリスト
  - 【設定例】

```
(setq tcode-input-filter-functions
      '((tcode-katakana-mode . japanese-katakana)
        (tcode-bushu-nest . tcode-do-prefix-bushu)
        (tcode-alnum-2-byte . tcode-1-to-2)
        (tcode-shift-state . tcode-mazegaki-prefix)))
```

- リストの先頭から順に適用される。

 15

カタカナの入力

- ひらがなよりは忘れやすい。
  - 練習テキストも不足
- ひらがなからカタカナへの変換
  - コマンド tcode-katakana-previous-char
  - シフトの利用

 16

記号類の入力

- 部首合成変換
  - 辞書symbol.revを使う。
  - 【例】白丸→○
- 交ぜ書き変換
- 2打鍵目にスペース
  - TUTコードでの方法（Tコードでも利用可）
- コード表の拡張
  - 3打鍵以上も設定可能。
- コマンド tcode-insert-ya-outset

 17

【交ぜ書き変換】前置型・後置型の違い

	前置型	後置型
操作の確定性	確定的	非確定的
変換範囲の指定法	読み入力直前・直後に指定	読み入力後に対話的に指定
汎用性	大	小

 18

【交ぜ書き変換】後置型での変換範囲の指定

- 対話的に指定する。
  - 最長一致
  - 非活用語優先
- 文字数で指定する。
  - 18 28 38 48
  - 29 39 49 59

 19

【交ぜ書き変換】候補の選択法の設定

- 変数`tcode-mazegaki-inline-kouho-count'
  - 候補一覧が出るまでの回数
- 変数`tcode-mazegaki-priority-list'
  - 一覧表への候補の並べ方
- 変数`tcode-mazegaki-alternative-select-left-keys'・`tcode-mazegaki-alternative-select-right-keys'
  - 二者択一の場合のキーの割り当て

 20

【交ぜ書き変換】後置型交ぜ書き変換の実際

- 活用語とそうでない語を区別する。
  - 活用語の変換は58で。
- 不要な候補は辞書から削除する。
  - コマンド tcode-mazegaki-delete-kanji-from-dictionary
  - 消しすぎに注意
    - 特殊な読み
    - ど忘れ

 21

【部首合成変換】前置型・後置型の違い

- 打ち間違ったときのやり直し方
  - 前置型
    - 2文字目を間違うと、undo後、最初からやり直す。
    - 予想と違う字に変換された場合も同様。
  - 後置型
    - 2文字目を間違うと、消して入力し直す。
    - 予想と違う字に変換された場合は、undoすると変換前に戻る。

 22

【部首合成変換】アルゴリズムの概要

1.各文字を部首に分解する。
　A = a b c
　B = x y z

2.関連の強い字を辞書から探す。
　C = a b x y
　D = a b c x
　　:

3.優先度の一番高い字が合成結果になる。

 23

【部首合成変換】アルゴリズムのカスタマイズ

- 変数`tcode-bushu-functions'
  - 選ぶ順番を指定
    - 独自に作ることも可能。
  - 【設定例】

```
(setq tcode-bushu-functions
                    '(tcode-bushu-complete-compose-set
                      tcode-bushu-complete-diff-set
                      tcode-bushu-strong-compose-set
                      tcode-bushu-strong-diff-set
                      tcode-bushu-weak-compose-set
                      tcode-bushu-weak-diff-set
                      tcode-mazegaki-lookup-with-prefix))
```

 24

【部首合成変換】対話的部首合成変換の活用

- 文字を部首から探す。
- 交ぜ書き変換と連係する。
  - 補完型入力の一種
- 【課題】
  部首合成変換用辞書の改訂

 25

補完機能

- 交ぜ書き変換を利用した補完
  - ユーザが明示的に補完する。
- 補完用辞書からの補完
  - 入力文字列が候補にマッチしたら、その候補を自動的に表示する。
  - 新し目のEmacsでないと使えない。
  - 現在、交ぜ書き変換辞書との連係を実装中。

 26

tc2の今後の課題

- 部首合成変換用辞書の改訂
- EELLLTXTの充実

- 【あったらいいと思う機能】
  - 日本語スペルチェッカ
  - 日本語の動的補完 (dabbrev)
  - ユーザに合ったコードの自動割当て（練習テキストも含む）

 27

Tコードのこれから

- ユーザは増えるか?
  - 入力システムの整備
  - 多様な練習法の確立
  - 広報活動

 28

おわりに

- 開発者・協力者求む。
- バグレポート・要望などは Tコードメーリングリスト(tcode-ml)へ。