解　説

日本文エディタ†

小野芳彦††

1. はじめに

エディタで日本語を処理する必要のある分野としては，いまのところ文書処理が主なものである．もっと極端に言えば漢字入りの文書作成以外には大口の用途はないと言っても過言ではない．したがって，現在にいたる日本文のためのエディタはワードプロセッサとして開発されたものがほとんどであり，それ以外の用途のものは単なる入力の補助で，とてもエディタとは呼べない．そのワードプロセッサにしても，最近の欧米文用のエディタと比べると使い心地の点で数段劣っているように感じられる．

筆者のような計算機を道具に使う研究者にとって，エディタは特に重要な道具であり，最近のテキストエディタの進歩で，プログラミングや英文の作成の能率がよくなって，非常に大きな恩恵をうけている．この恩恵を日本文にもと願うのは筆者だけではないと思う．テキストの編集という作業からみれば欧米文と日本文の間にそう大きな差があるとは思えず，使い心地の差は，多分日本文エディタの歴史の浅さ，すなわち，まだ使いこまれていないことからきているのではないだろうか．

本稿で日本文エディタを論ずるにあたっては，日本文エディタが現在どうなっているかという点を個々に解説するのではなく，エディタの使い心地をよくするために取り上げられるべき事項について，日本文エディタと欧米文用との比較を重点に考察する．

2. 社会的環境

2.1 タイプライタと社会

アメリカにおけるタイプライタの利用には，現在のスタイルのものが発明されてから100年以上の歴史がある．その入力具の上に現在のエディタの発展が生れたといってよいであろう．欧米では，まずプログラミング用としてエディタの発展が始まったが，プログラミング言語自身が自然語に近いシンタクスを持つのでワードプロセシング用に無理なく転用可能であった．

コンピュータによるワードプロセッサが登場するまえから，欧米のタイプライタは単なる浄書機としてではなく，ワードプロセシングの一部として働くようにシステム化されていた．すなわち，機関銃のように打つといわれる高速性のために，文章を練るための道具として草稿段階から使用可能だったのである．それを社会全体で可能にしたものとして，専任のタイピストの存在を忘れることはできない．結局，ワードプロセッサはそのシステムの中の機械化可能な部分をコンピュータで機械化することで，タイピストあるいはタイピングをうまくこなす人がより便利に正確に文書を作成することができるような専門の道具として開発されたものであるということができる．

わが国においては，先発的な役割りを果たした和文タイプライタのスピードが遅く，単なる浄書機であったため，日本文の機械によるエディティング，さらに機械による高度なワードプロセシングは，いわゆる日本語ワープロとして使われるビジネス用パーソナル／小型コンピュータの普及で，ようやく始まったところといった状況である．

しかし，オフィスにおける入力の専門職が育つ以前に日本語処理の機械の急激な開発競争と導入ブームが始まったことで，そこに採用されている方式が日本文のワードプロセシングに最適であるというふうにはなっていない．日本文のワープロやエディタは売れるヒット商品として，ユーザの最初の取り付きやすさに重点が置かれており，家庭用電化製品と同じレベルの素人向きの製品として設計されている面がある．

一方，入力の専門家としての日本文タイピストという職業は育たない，すなわち，一般のオフィスでは誰もが自分で文書を作ってしまい，日本文タイピストがいて書類を打つことを専門の仕事とするという状況に

† Editors for Japanese Text by Yoshihiko ONO (Department of Information Science, Faculty of Science, University of Tokyo).

†† 東京大学理学部情報科学科

はならないのではないかと考える人も多い．効率が手書きより少ししか改良されなくてもすべての人が楽に書類を作れるようにしよう，という方向の研究が現在の入力方式の研究開発者に多くみられる．

素人にとっては使いやすくても専門家にとっては使いにくいということがよく指摘されるものがある．エディタもその代表的なものである．日本語のワードプロセッサの利用者は，最初はほとんどが素人であったが，いつまでも素人でいるのではなく，やがて専門家になるはずである．そのとき，最初の取り付きやすさが使いやすさにつながっているかはいささか疑問なシステムが多いように思われる．このことについては後の章でのべる．

2.2 文字と正書法

次に，日本文には字種が多いという問題がある．欧米の一般の文書では100以下の文字種でこと足りるのに比べて，日本語の文書では，1人が一時に使用する文字種で500〜700はある．ところが，社会全体となると，それをはるかに越える数千種以上の文字が必要であると考えられている．

このことが日本文システムの設計に多大の負担になっていることは容易に推察できる．製品が実用化できるかどうかは，この数千種の文字が入力できるかどうかにかかっているとして，各社力を注いでいるが，その取り組み方には，字種を広げようという考えばかりで，うまく絞ろうという考えが抜けているようにみえる．一生に1度使うか使わないような文字についてはもっと迫害すべきであろう．現在の JIS 規格の2つの水準や，常用漢字によるクラス分けは大き過ぎるきらいがあり，われわれの実感に近いものがほしい．これからの日本語システムとしては，使用字種の個人的なチューニングが可能なものが望まれている．

ところが逆に，漢字の字種をどの程度常用すべきかという議論に，カナ漢字変換日本語ワードプロセッサが係わりをもつようになった．つまり，難しい漢字あるいは熟語は読めさえすればよく，書く方は機械にまかせればよいので，字種の制限をやめよう，あるいは，積極的に常用字種や慣用熟語を増やそうとの意見である．一般に，現在の社会にはコンピュータに対する過大な期待感があり，これもその1つであろう．漢字変換型ワードプロセッサを効率よく運用するには，できるだけ字種を減らし，余分な変換を減らすことこそ肝要であるということが知られていない．

もう1つ指摘しておかなければならない点は，日本語においては書き方・つづり方という基本的なレベルで正書法が確立していないことである．同じ語をカタカナ・ひらがな・漢字の3種の文字を適当に使い分けてつづることが可能であり，その社会的な容認の度合もかなり大きい．送りがなのようにここ30年の間に2度も方針が変わったものもある．これらも，先の漢字の字種を増やそうとする動きと同様の負担を引き起こしている．

3. 入力方式依存性

この章では，エディタとして入力方式に対する依存性を取り上げる．

キーボードは人間が計算機に大量のデータを入力する器具として人間工学的に優れたもので，現在までのところそれに勝るものは発明されていない．漢字混りの日本文を英文と同じようなキーボードで入力する可能性が未知であったころの入力方式は，全文字配列型のいわゆるペンタッチ式が主流であった．現在はカナ漢字変換方式が商業的に実用になって主流になりつつある．（筆者らはカナ漢字変換が人間工学的には高速タイピング向きでないと主張しており，それに代わる無連想の2ストロークコード入力方式を提唱している．）エディタはテキストの大量入力をサポートするものであるから，今後もキーボードを使った入力方式が主流となるであろう．

現在，将来に向けての入力法の研究としては音声入力・オンライン／オフライン手書入力などがあり，万人向きのシステムとして実用化が期待されている．これらについてもエディタが必要であり，その設計が行われている．キーボードによるものとは共通点もあれば本質的に異なる点もあるであろうが，本稿ではキーボードによるものに話を限定することにする．

キーボードによる入力方式に限定しても，英文には存在しない問題がある．日本語では複数打鍵で1字あるいは1単語を入力する．一般に，文字・単語を入力するルーチンはエディタを含むコンピュータシステム全体で共通に使用するようにしたほうがよい．しかし，エディタは直接打鍵列を扱うか共通の入力ルーチンから単語をもらうかによって大きく設計が変わってくる．たとえば，後者で画面型エディタを実現するとして，画面に打鍵列を自然な形で表示することは難しい．そのため，打鍵列を表示する専用のエリアを画面にもうけ，変換後に画面の該当位置に移動することになろう．

さらに，打鍵列を変換後捨ててしまうと，タイプミスに対する修正がまったくの打ちなおしになってしまうし，漢字変換方式では，変換の続きを後から再び行うというようなことができないかあるいは非常に難しくなる．このような点を考えて逆変換を用意するか打鍵列を取っておくシステムが最近開発されている．

現状では入力方式に依存する部分があまりに大きいため，エディタを全面的に作り直さなければ別の方式に変えられない．入力方式に依存しない部分をふくらませて各種の入力方式を取り込むことのできるトランクエディタのようなものがこれからの研究課題となっている．

4. 入出力装置依存性

この章では，エディタの入出力装置に対する依存性の問題を論ずる．

4.1 情報交換符号系

まず，入出力装置とプログラムの間の情報交換の符号系の問題がある．文字に関しては国際標準が決められており，それに沿って各国がその国の標準を決めることになっている．わが国では，カタカナを含めた1バイトの文字セット JIS-C 6220 と，漢字を含めた2バイトの JIS-C 6226 が公式の規格となっている．また，制御文字に関しても，印刷製版を考慮した JIS-C 6225 が制定されている．

1バイトのコード系と2バイトの漢字コードとは共存させるのが通例で，JIS-C 6228 に従ってその間を切り換えることになる．1バイトの文字セットを呼びだす制御文字列は使用する国の文字セットごとに決められているのであるが，現状では ESC (Hを採用しているものと ESC (J を採用しているものがある．さらに，JIS-C 6226 の呼出しは，1983年の改定で ESC$@ から ESC$B に変わったという経緯がある．切り換え制御文字列が3バイト（1バイトが7ビットならば SI がさらに必要で計4バイト）と長いのを嫌って，短い独自の制御文字列を採用しているシステムもある．このように同じ JIS コードでも制御文字にいくつかの差異が生じている．

さらに，JIS 以外のコード系で漢字を表わすシステムも存在する．その中で現在よく使われているものは，次の2種類である．1つは，IBM 互換の大型計算機などで採用されているもので，それぞれ別の名があるので仮に拡大 JIS コードと呼ぶことにする．1バイトのコードが8ビット EBCDIC 系であるため，JIS コードの 94×94 文字より広い 256×256 の空間をもっている．しかし，JIS コードの各バイトの MSB を1にしたものが拡大 JIS コードであり，変換は非常に簡単である．

もう1つは，パーソナルコンピュータシステムで使われ出したもので，シフト JIS と呼ばれている．第1バイトとして JIS-C 6220 で文字が割当てられていないコード 81～9F，E0～FC を使い，第2バイトとして 40～7E，80～FC を使う方式である．第2バイト 188字に JIS コードの2つの区を対応させることで，第1バイトを半分に減らし，JIS-C 6220 に押しこめたもので，文字セットの切り換えの呼びだし手順を省略した方式である．

シフト JIS は，切り換えがないので，プログラムで単純に英数カナ文字のつもりで出力するだけで漢字を表示できるようになっていて，既存のシステムからの拡張が容易であるという特徴をもっている．(第2バイトに 20～3F を含めなかったのは，たとえば，文字列定数を括るコーテーションマークが第2バイトにあると，文字列定数が途中で終わってシンタクスエラーとなってしまうことなどを防ぐためであるという．)

そのほかに，各メーカなどが独自で持っているコード系もある．これらは，JIS-C 6226 制定以前から使っていた漢字タブレットの座標であったり，独自の辞書のインデックスであったりしたものである．現在では，上記3種のどれかに変換するような方針で運用されているが，印刷業界では JIS に制定された文字では足りないために，独自のコードに依然としてとどまっているところもある．

現在，コード系は上記3種にかたまりつつあるが，3種もあるのは多い．また，呼出しのための制御コードにばらつきがあるのも問題である．歴史の浅い今のうちに1つに統一すべきである．しかし，われわれが利用する一般用のエディタをこれから設計するような場合には，プログラム・データ・入出力機器の互換性のために，上記3種への変換をすることを考慮しておくのが現実的であろう．

4.2 画面制御コード

次に，出力装置への依存性を取り上げる．エディタの場合，特に問題となる点は画面制御コードの機器依存性である．すなわち，ディスプレイ装置の機種ごとにその画面制御コードがばらばらなので，極端な場合は機種ごとのエディタを作る必要があることである．この問題は，欧米文のエディタでベストセラーのもの

がでるようになって顕在化してきた．エディタがメーカの手によってシステムごとに作られている段階である日本文の場合でも，最近はパーソナルコンピュータ用にメーカとは独立に日本語ワードプロセッサが作られるようになってきた．それらは依存としてシステムやディスプレイを特定しているものばかりのようであり，システムのディスプレイの上位への交換で動かなくなるなどは，そろそろ問題となりそうである．

英字のキャラクタディスプレイ端末の画面制御コードは，ANSI の規格はあるが，各社各種のものを統一するのはもはや不可能であろう．日本には制御文字の規格としての JIS-C 6225 があるが，これは印刷組版を念頭に制定されたもので，画面スクロールなどの基本的なものが規定されていないような中途半端なものである．1984 年がその改定予定年で現在改定作業中のはずであるが，はたしてどうなるであろうか．

現実の日本語キャラクタディスプレイ端末は，ベースとなる英字システムの上に拡張されたものであり，ベースの制御コードの多様性に拡張の多様性が加わるといった形で，とても統一コードが期待できる状況にない．満足な基準もない現状では，ディスプレイの機種依存性をへらすには，すべての制御コードを受入れられるような設計にする必要がある．

先発の英字エディタには，ほとんどの端末に適用可能なように設計されたものがすでにあり，その方式はだいたい次のようなものである．あらかじめ，ディスプレイを制御するのに必要と考えられる標準の制御機能を設定し，それをインタプリットするコード列を端末の種類ごとに集めてデータベース化しておく．エディタは，標準の制御機能を基に設計されており，端末の種類を知ってデータベースからそのコード列を得て，実際の制御コードを出力するというものである．

この種のエディタが設定している標準の機能は単一レベルではない．たとえばカーソルの移動機能をとってみても，1 文字 1 行分の移動を使えば任意の位置への移動は実現できるが，任意の場所への移動とか，画面のある特定の場所への移動とかを使って制御をより速くしている．このように，標準機能を基にした設計といっても，低いレベルの機能だけを使うようではだめで，端末のより高い機能を十分使えるようにしておく必要があり，また，実際にそうなっている．

以上のような工夫で多数のディスプレイに適用可能なエディタを設計することができるが，実際には非常にきめ細かいレベルで手当をしなければならない．たとえば，ある種のディスプレイ端末では，画面の右端から文字があふれて次の行に移ったちょうどそのときのラインフィードが無視されるものとか，ある特定の文字が表示できないものとかが存在する．現実のその英文エディタではこれらにきめ細かく対処している．

このように，たくさんの機種についてノウハウの蓄積があって初めて，真の端末適応型のエディタの実現が可能である．日本文でもこのようなきめの細かい手当を行うエディタを作成しなければならないが，英文の場合を参考にすることで，スタートラインをずっと先に進めることができるであろう．

4.3 プリンタ依存性

プリンタに関しては，前記の文字コード・制御コードの異なりのほかに書式制御をどのように扱うかの依存性が存在する．たとえば，文字フォントやサイズの変更のように，レーザプリンタのような高精度プリンタでは可能でも，文字フォント内蔵のドットマトリクスプリンタでは不可能な清書機能もある．

前述の JIS-C 6225 は，書式制御にとってはほとんど十分な機能が含まれており，機能不足が問題になることはないと思われる．むしろ，問題となる点はそれらの制御機能のレベルに高低の不ぞろいがあって，エディタ（あるいはフォーマッタ）がどのレベルまで実行してよいかの判断を必要とすることである．たとえば，行やフィールドの中で左右の端をそろえるには，制御コード JF を使って出力装置にまかせてしまうことも，自分で各文字の間隔を計算して可変幅のスペース PS を間に入れるようにすることもできる．このような判断は，先に述べた画面制御コードの取り扱いとまったく同様に，プリンタの能力を記述したデータベースを利用すればよい．この場合の標準の機能として，JIS-C 6226 を利用できるのではないかと考えている．

5. コマンド体系

編集コマンドを分類する座標軸として，次の 3 つを挙げることができる．

1—対象の大きさ
2—テキストの内容への依存性
3—出力・書式への依存性

これまでの日本文エディタは 1 と 2 の軸についてはほとんど広がりを持っていない．すなわち，英文エディタの初期の段階にまだとどまっているといえる．この点が使い心地の最大の差であろう．

5.1 ポインティング

エディタの最も基本的な操作は，編集の対象を指示するポインティングである．現時点での日本文エディタのポインティングは通常1文字であり，その移動量も上下左右1字分である．それより大きな移動は，画面とかページとかを単位とするもので，テキストの内容とはあまり関係がない．カーソルの移動は，見えている部分（画面）を変える大きいものと，画面の中で詳しく位置を指定する最小のものを組合せればそれで十分という考え方であろう．しかし，ポインティングの方式が通常はカーソル移動キーによる間接的操作であるために，目的の位置への移動がスムースには行いにくい．特に横方向の大量の移動（キーを押し続けると速く動くことを利用する）はキーを押し過ぎて，カーソルが行き過ぎたりして（エディタの反応速度が遅いせいであることも多い），能率が悪いだけでなく，イライラして使い心地も悪くなる．

欧米文では単語単位のカーソル移動がスペースを目印として簡単に実現でき，それによって修正位置へのポインティングが大変楽になっている．漢字かな混りの日本文はスペースによる語の分かち書きをしないが，修正は欧米文と同様であるから，1文字単位だけでなく単語単位の移動をコマンドとして用意するのが望ましい．さらに，英文ではすでに実現している句・文・段落などを移動単位とすることも必要である．

エディタが文章の内容に関係する単語としての区切りをどのようにして見つけるかは，重要な問題である．文節単位の漢字変換方式では，文節・単語の切れ目で変換か無変換を押すことになるので，それをテキストに残しておけば目印となる．また，ローマ字入力の分かち書きのスペースも，表示上はなくなっていても実際にはテキスト中に残しておくことで，同様に目印となる．漢字変換以外の方式でも，簡便な代用として，字種の変わり目を利用する方法がある．これは厳密には単語の部分であったり数単語であったりすることになるのであるが，もともと中くらいの移動量を得ることと編集位置へのフィットをよくするためのものであるからこの程度で十分であるともいえる．逆にたとえば活用語尾などの変更の際にうまくフィットするという効用もある．

句や文を認識するのは句読点の文字を判別するだけであるから容易である．また，段落も通常は改行とスペースが連続している場所をさがせばよい．

このような対象の大きさを大きくするというポインティング位置の移動のほかに，指定された文字列をさがすというポインティングもある．その方式として，文字列を一括してマッチングをとるものと，インクリメンタルにマッチングをとるものがある．両者の最も大きな違いは，前者がテキスト表示域の外にさがす文字列を表示するための場所が必要なのに比べて，後者はそれがいらないという点であり，後者のほうが画面エディタ向きである．しかし，漢字変換方式の場合には，変換前の入力文字列を残しておかなければ，インクリメンタルな方式は不可能である．

5.2 新しいポインティングデバイス

前節に述べた文の構造に依存したポインティングのほかに，直接的なポインティングを可能にする新しいデバイスがエディタ用に開発・実装されている．中でも最近評判の高いデバイスはマウスである．以前からこの種のデバイスとしてトラックボール・ジョイスティックなどがあったが，マウスの場合，実際に物体を動かすので操作が直接的で動きを微妙に調整しやすい点と，2つないし3つキーがついていて，選択的コマンドの実行に便利な点が以前のものと異なる点であろう．

ポインティングデバイスは，新聞のように複雑なレイアウトを必要とする場合，その能力が十分に発揮される．しかし，ポインティングとテキストの打鍵を交互に繰り返す場合は，手をキーボードから大きく離すことが作業の円滑性を損なってしまい必ずしも使いやすくはならない．したがって，キーボードによるテキストの編集には，ポインティングデバイスだけでなく，コマンドによるものも併用する必要があろう．

一方で将来の入力方式に適応したデバイスとして，たとえば，手書文字のオンライン認識方式のエディタならばその入力ペン，音声認識方式ならばマウスというように，適材適所で利用されるようになるであろう．

5.3 画面エディタの基本的編集コマンド

一般に，エディタで純粋な編集操作といえば，「削除」(delete)・「挿入」(insert)・「置換」(replace)・「移動」(move)・「複写」(copy) である．欧米文のエディタでは，行エディタから画面エディタに変わるにつれて，このような基本的であると思われる操作への対応が変わってきた．

行エディタでは，このような作業を1つのコマンドとして用意するが，作業が2段階になっていて画面をまたぐことの多い「移動」・「複写」を，画面エディタで1つのコマンドにするのは難しい（あるいは，他のコマンドと異質になってしまう）．そこで最近は，これ

らを2つの作業に分けるようになった．まず，delete または copy によって移動／複写元のテキストをバッファに退避し，次に put で移動／複写先へそれを挿入するという方式である．

日本文ワードプロセッサではコマンドにパラメータを与えるとき，いちいちどんなパラメータを入れるかをエディタが指定して聞いてくるという素人向きの方式をとっている．たとえば，削除コマンドは削除開始位置をポイントしてから「削除」キーを押す．すると，「どこまで？」と聞いてくるので，削除の終端位置までをポイントし，「実行」キーを押すという2段操作になっているのが一般的である．「移動」と「複写」も同様な方式をとっているので，3段の操作である．これらは，操作の対象となる部分を光らせるなどして意図どおりであるかを確認させるという意味では安全な方式であるが，たった1文字の削除でも多段の操作をして確認しなければならないのは，慣れたユーザにとって苦痛である．

行エディタでは新しいテキストの「挿入」をいつ始めるかをコマンドで知らせる方式が普通であったが，画面エディタではすべての図形文字の入力は「挿入」になるという方式が流行している．コマンドはすべて制御文字で始まるようにするのである．この場合「置換」は「削除」を先に行うことと同じになるのでコマンドとしてはなくなる．

素人向きの日本文ワードプロセッサは，もっぱら文字を入力するキーボードとは別に，その周りに配置されたコマンド名を刻んだキーでコマンドを入力するという形式が一般的である．したがってテキストの入力を「挿入」型にすることができる．しかし，初めのうちは，画面上で古いテキストを上書きする「置換」型であった．これは，誤まって上書きしてしまうと元のテキストを失ってしまうため，あまり良い方式とはいえない．現在でもそれが残っているのは問題である．

画面エディタの基本コマンドのうち，delete と copy はすでに存在するテキストに関する操作である．これらを**5.1**節で述べたような方式のポインティングと組み合わせ，現在ポイントしている場所から次のポインティングによってカーソルが移動する場所までを編集するような方式をとっているエディタが存在する．これは，1文字・1語・1行・1文・1段落・1ページなどの編集を一度にやってしまうことのできる，まさに専門家むきのエディタである．

このように，コマンドとポインティングを直交させることによって，編集の能率を上げることと記憶すべきコマンドの種類を少なくすることが同時に可能となる．ただし，こうすることによって，すべてのコマンドが複数打鍵になるが，よく使われるたとえば「前の1文字の削除」などは，組み合わせ方式のほかに特別に1キータッチで実行できるようにして，編集を滑らかに進める考慮がなされている．

5.4 出力・書式依存性

出力・書式依存性とは，でき上りの文書の形態・配置に対する関係の深さのことである．削除などの通常のコマンドは依存度が0であるが，清書型のワードプロセッサにはセンタリングなどの依存度の高いコマンドが装備されている．通常のプログラム用エディタでも，自動インデント付けなどの出力依存の機能を持つものがないわけではない．

書式依存型の機能には，文字の形でテキストに含まれているものと，コマンドとして実行するものとがある．前者の例として，スペースと改行がある．この2つは図形文字ではないが，句読点のような区切り記号としての役割をも果たしている．

逆に，文字列を一定の幅に広げるために文字の間を空けるとか，長い文や段落を折り返して何行かにわたって画面表示するとかするために，本来テキストに存在しないスペースや改行文字を入れなければならない場合がある．このような制御文字を，たとえばスペースの場合，前者をハードスペース，後者をソフトスペースと呼んで区別している．

一般にプログラミングエディタではこのようなソフトな制御文字を必要としないが，清書型のワードプロセッサでは，それらを積極的に利用して書式調整作業と編集作業を分けているものがある．編集作業においてはハード・ソフトの区別はしないが，書式調整のコマンドを実行するときには，ソフトな制御文字を取り去って改めて調整しなおすという方式をとっている．

タブのような位置整形機能は，より基本的なスペースに展開しておくこともできる．タブは JIS-C 6220 の制御文字であるのでスペースに展開してしまうエディタはほとんどないが，センタリングとか右寄せのような機能は先に述べた JIS-C 6226 の制御文字であるにもかかわらずコマンドになっていて，スペース（ハードスペースであることすらある）に展開されるエディタがほとんどである．

整形コマンドはその場でスペースなどに展開してしまうので，編集でフィールド幅や文字列の長さを変え

るたびに再度コマンドを実行する必要がある．制御文字方式では画面表示の上でだけソフトスペースで調整されているのであるから，表示を改めるだけでよい．

整形の機能にはアンダラインのように整形の始めと終りを組で指定するものもある．これらを制御文字として2回に分けて別々にテキストに挿入するのはあまりうまい方法とは言えないが，現実にそのような方式をとっているワードプロセッサは多い．このような整形は制御文字を挿入や削除することによって編集できるのでまだましであるが，半角や倍角のようにモードキーで切り換えるようにして，入力のときにしか変更しないようになっているものは，全体を打ち直さなければ編集できない．このような範囲を指定する整形機能は，削除コマンドと同様でポインティングと組み合わせるようにすれば問題が解決する．

特に，モードキーによるモード切り換えが編集できないことは，エディタの使い心地を悪くする大きなファクタとなる．これは整形コマンドだけではなく入力一般についていえることである．たとえば，JIS キーボードを使ったカナ漢字変換ではすでに文字種用のモードだけでもカタカナ・ひらがな・かな記号・英小文字・英大文字・英記号とある上，たいていはひらがなモードでなければ変換機能が働かない．文章には英数字・英記号・カタカナが単語としてたびたび出現するが，そこで無変換操作を行ったり，カーソル移動を行ったりするとモードを元に戻し忘れることがかなりある．いちいち画面を見て打つ初心者でない限り，モードの誤りはすぐには気付きにくいので，せっかく指の位置としては正しく打ってもすべて打ち直しとなってしまう．

かといって字種のモード切り換えまで制御文字としてテキストに残すのはやりすぎであろうが，入力途中の変換前の文字列にモードだけの訂正操作をすることは可能である．さらに進んで，変換入力をモードにかかわらずに打鍵列としてとらえるような方式も考えられるが，このような入力の訂正を考慮したものはないようである．ローマ字から変換する方式では入力のモードがないのであるから，無・カタカナ・ひらがな・漢字の4つの変換を用意するというのが本筋である．しかし，まずローマ字をひらがなに自動変換し，次にカナ漢字変換を行うように設計されているものがほとんどであり，カタカナや英字を入力するにはそのためのエスケープ文字を入れるというような使いにくいものになっている．

字種を変えることは，一種の書式整形とも考えられる．小文字と大文字の間の変換コマンドが英文の場合用意されていたりするのに対応して，カタカナとひらがなの間の変換コマンドくらいはあっても害はない．

書式に関するさらに複雑な操作は，エディタのレベルでは処理できないので，フォーマッタあるいはプリントプログラムへのコマンドをテキストの中に埋めこんで処理することになる．そのようなものとして，宛先などの可変部分に実際の住所氏名を差し込むといった業務に密接した機能が，最近の日本文ワードプロセッサのセールスポイントになっている．

このように，現在の日本文ワードプロセッサは，よほど特別な機能でないかぎり，すべてエディタのコマンドで書式を処理する方式をとっている．英文では，各種の書式に共通の記述が可能で書式の変更が容易であることで，外付けのフォーマッタを積極的に利用することが多い．書式が持つ論理的な情報を無視して形式のみを整える方式には，一旦ある書式に合わせたものを別の書式に変更することが細心の注意を必要とする大変な作業になるという不便さがある．書式を大きく変更することを考慮することがめったにないのは，日本文のタイプ清書が時間のかかる大変な作業であった昔のなごりであろうか．

6. おわりに

日本文のワードプロセッサを使って感じる使いにくさを，英文エディタと比較して分析し，その改良点を述べた．比較の対象としたものについては，あえて実名を上げていないが，初期のものから最新のものまでを網羅したつもりである．すでに改良されていたり，別の方式でもっとよいものがあるかもしれないが，その場合はご容赦願いたい．英文のエディタについては，本号に詳しい解説があるので，参照されたい．

現在の日本文ワードプロセッサは，1に入力に汎用性と取り付きやすさを持たせ，2に書式処理に汎用性を持たせるという順に力が注がれており，編集作業は最後に回されているようである．

さらに，日本語の場合，様々な負担が機械に対しかかっており，テキストの入力を人間の能力でまかなうかぎり，汎用性を追及することと使いやすさを追及することが背反の関係になっている．使いやすい日本文エディタを設計するには，現状ではまず汎用性を表に出さず，使用頻度の高いものに照準を合わせて設計すべきであると考えている．（昭和59年5月21日受付）