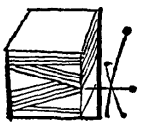
論　説



入力効率を良くするための文書処理システム†

小　野　芳　彦††

1. 文書処理システム

1.1 文書処理システムによる文書作成過程

従来，日本において活字あるいはそれに近い品質のきれいな文書を作成するには，印刷業あるいは和文タイプによらねばならなかった．これらは，原稿から印刷物へのターンアラウンドタイムが長くて，手軽に利用できるものではない．このため，校正段階での大幅な手直しや，たび重なる校正，印刷後の改良などを，著者の方で遠慮しがちになり，文書の質は良くならないし，逆に質を良くしようとするためには草稿作成時に必要以上に時間をかけなければならない．このような欠点が日本文による文書の生産性の低さの原因であると思われる．

一方欧米においては，簡易な印刷機器として百年以上の歴史を持つタイプライタが存在し，タッチタイプ技法によって草稿を書くスピードより速くきれいな文書を作ることができるため，文書の生産性は非常に高くなっている．ただし，この生産性の高さは，職業としての専門のタイピストが各オフィスにいることにあることに注目しなければならない[1]．

マイクロコンピュータの出現と周辺エレクトロニクス機器の価格低下により，文書作成に電子計算機が使われるようになってきた．計算機で文書を扱うことによって次のような利点が生まれた．

（1）　文書の訂正が全体を打ち直すのに比べて格段に少ない作業量・時間で行える．

（2）　文書の高度な書式化を手軽に行える．

（3）　文書の質を高めるために種々の支援ができる．

（4）　多くの文書の保存・管理が行える．

文書を計算機で処理する場合，文書の作成過程（図-1）は，草稿を文書ファイルに蓄え（原始入力），それを適当な書式で印刷して校正刷りを得（清書），校正してそれを文書ファイルに反映する（編集校正）ことを繰り返し，最終印刷文書を得る（再び清書）となる．さらに校正を助けるために，計算機によって種々の支援情報を取り出す（校正支援）こともできる．これら4つの処理プログラムをそれぞれ，原始入力エディタ，編集校正エディタ，フォーマッタ，校正支援プログラムと呼ぶことにする．

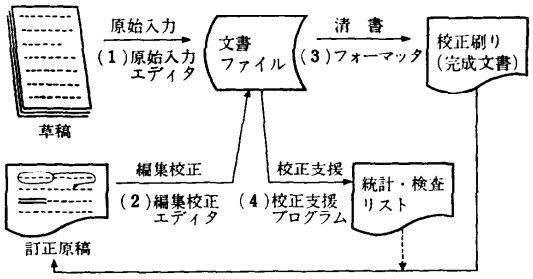


図-1　文書処理システムによる文書の作成過程

1.2 文書処理とタイピスト

欧米の文書生産性の高さが，タイピストの存在にあることを前に触れたが，日本文も英文と同程度に速く打つことができることがわかってきたのであるから[2]，日本文の文書処理においてもタイピストの存在を考慮の中心に置くべきであろう．すなわち，文書処理システムを扱う人は草稿を書く人（著者）ではなくてタイピストである方が能率的であり，そうするためには草稿や訂正指示方式を含めた全体としての文書作成方式を議論しなければならない．もちろん，著者自身がシステムを操作することを否定するものではなく，そのような場合においても，素人向けの文字入力方式を使えば，十分な能率が得られるように設計する必要がある．

1.3 テキストと書式制御指令

文書は，テキストと書式制御指令が結合されてできあがったものである．この結合は，従来は人手によっ

† Toward a Document Processing System Supporting Efficient Japanese Input 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

て行われていたものであるが，文書処理システムにおいては，原始入力あるいは編集校正エディタの機能を利用したり，フォーマッタに行わせたりした方が正確で能率的である（たとえば行のセンタリング）．

特に，すべての書式制御をフォーマッタに行わせる方法は，フォーマッタが強力であればあるほど便利なものである．この方法の長所は，

（a） 書式情報が保存されているため，書式の全面的あるいは部分的な変更に対して修正が少なくてすむ，

（b） 原始入力や編集校正時に使う表示装置では実現できないが，印刷装置では可能な機能を記述できる，

の2点である．

一方，すべてをエディタで行う場合と比較すると，

（c） 清書出力とエディタが処理するテキストとの間の対応がつけにくく，また後者が読みづらい，

（d） 原始入力，編集校正中に清書出力のイメージが得られない，

（e） テキストへの書式制御指令の挿入を草稿から判断して行う作業は，著者以外には難しい，

という欠点がある．

しかし，文書処理システムにおいては編集校正を手軽に行えることが重要であるため，長所(a)は必須の条件である．したがって，上の欠点はエディタ側でできるだけカバーするシステムとする必要がある．

1.4 文書の論理的構造と文書処理システム

書式制御が行われる単位は，文書を構成している論理的階層の一層であるのが通例である．たとえば先頭の一文字を下げるという制御がパラグラフに対して行われるとか，図や表をページにまたがらないように制御するなどがそうである．

文書の論理的階層は，細かくしていけば文の構成要素にまでいたる．たとえば，ゴシックにするなどの単位が，句や単語であることなどが挙げられる．

また，編集校正の単位も，文章の構成単位であることが多い．訂正は語句に対するものが最も多く，移動や削除は文やパラグラフに対するものが多く見られる．また，書式の部分的訂正は，その制御範囲の及ぶテキストに対する処置として扱うことによって，具体的な訂正結果を推定しやすくなり，間違いも少なくなる．

校正支援プログラムでは，単語・文・パラグラフに対する統計情報や辞書照合が有力な道具として使用される．

このように，文書の階層的構造は，文書処理システムの個々の処理において，重要かつ有用な役割を果たすことができる．したがって，単なる一次元の文字列として文書を保存しておくことは情報の損失であり，前述の書式制御指令を含めた文書の論理的階層構造を文書ファイルに保存するようにしなければならない．

2. 編集校正エディタ

ここでは，1章に述べた条件の下で，効率的編集校正作業を達成できる1つの解としてのエディタの仕様を提案し，現在の日本文/英文用テキストエディタとの比較をする．

2.1 編集方式と操作

タイピストは原稿を見ながらの機械的な入力は高能率で行うことができるが，継続的に判断を必要としたり，原稿から頻繁に目を離さなければならない作業の効率は上げられない．そこで，編集校正エディタは，上のような特性を考慮した上で作業が効率的であるような仕様でなければならない．

著者は，校正刷りの上に朱を入れることで訂正を指示する．タイピストは編集エディタがCRTディスプレイ上に表示した文書のテキストと渡された校正刷りを比べて，朱の入った部分を訂正していく．

現実の文書処理システムに見られるような校正刷りとエディタの表示テキストが同一配置でない場合，著者の指定した訂正個所を捜すことは，手間のかかる間違いの発生しやすい作業である．しかし，編集の対象となるテキストを，校正刷り1ページの単位で，同じ配置で画面に表示することにすれば，校正刷りが手元にあるため，訂正の位置を見つけることはきわめて容易な作業となる．

このように画面と校正刷りを一致させるからには，実際に訂正が行われた後も配置を変えてはならない．実際に紙の上に訂正されているのと同じように，削除は文字の上に線を重ね書きし，挿入は適当な行間の余白に記入するようにして，訂正で原テキストを動かさないようにする．挿入の量の多い場合は，別ページを設けて，そこに表示するようにする．(図-2)

訂正の位置を指定するにはカーソルを使う．カーソルは，訂正の対象となっている部分（前述の階層構造の一層）全体を覆うような，長さの可変なものにする．カーソルの移動のためには，カーソルの指す層のレベルを変えるコマンドと，カーソルを1単位分前後させるコマンド（あるいは二者を組み合わせたコマン

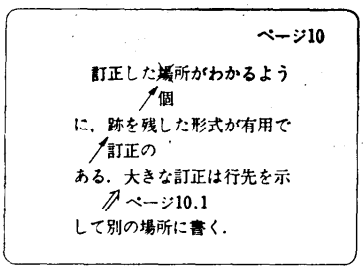


図-2　画面上の訂正の例

ド）を用意するだけで十分である．

書式制御指令を訂正するには，それが制御している階層のテキスト部分にカーソルを位置づけして，修正を施す．書式制御指令は校正刷りに陽に表示されてはいないのであるから，同様に画面に表示されてはいないが，編集のモードを変えることで，挿入されたテキストと同じ形式で行間に表示させ，通常の訂正方法と同じように訂正ができるようにしておく．

文書中の表・図・グラフなどでは，グラフの形式とか，表の罫線などは書式であり，タイトルや表・グラフの数値などがテキストである．したがって，それらは通常のテキスト・書式と同じ方法で訂正を行うことにし，実際に表を書いたり図を埋めこんだりする特別な作業はフォーマッタや専用のプログラムで行う．

2.2　方式の比較と評価

画面上に清書されたテキスト1ページを表示し，さらに挿入用の余白を空けておくためには，100 行×60 字くらいの表示能力が要求されるが，現在の CRT の技術では 24×24 ドット以上の品質の字を，一画面にそれだけの数表示することは困難である．しかし，注目していない部分を縮小するとか，半ページ単位にするなどの方法は，現在でも実行可能な方式である．

現在商品化されている英文/日本文ワードプロセッサの大部分は，編集エディタに書式制御の機能を持たせ，画面上で清書を行ってしまい，文書ファイルへは清書されたテキストを保存するという方式をとっている．この場合，編集が開始されると校正刷りと画面との一致は失われるし，書式制御指令の大部分が失われているため，書式の軽微な変更に対しても，テキストの修正が大量になることが頻繁に生じる．特に後者の修正をタイピストに確実に指示することは，著者にとって難しい作業である．

現在みられるカーソル制御の方式は，カーソルを上下左右に1文字ずつ動かすキーを使用するものが大部分である．この方式は，大きな移動が必要な時，非能率であるので，連続して速く移動させるキーをつけたり，アナログカーソル制御（マウス，ジョイスティックなど）を利用したりしている．しかし，これらによる遠隔操縦は意外に面倒だし，特に後者はキーボードから手を離さなければならない点が問題である．

ここに提案した，カーソルの長さを可変にして，編集対象となる部分全体を指すようにしているエディタは，現在筆者の知る限りでは存在しない．ただし，1文字のカーソルの移動の単位として，パラグラフや単語を，また訂正の単位として単語を指定できる英文エディタ[3]が存在していて，筆者は実際にその使い勝手の良さを体験している．

3. フォーマッタ

3.1　論理的書式制御

書式とは，文書のテキストをどのような形態で各ページに配置するかを規定するものであり，その制御指令は，テキストの形態やページ内の配置をその書式に合わせるために用いられる．書式制御指令自体は，行間を1行空けるとか，字体をゴシックにするとかいった物理的な指令であり，文書の内容に無関係にテキスト中に埋め込んでしまっても，当初目標とした書式制御を行わせることはできる．しかし，それらが一貫している保証はなく，与えられた書式と適合しているかどうかは，実際の出力の細部まで確認してみなければわからない．さらに書式だけを変更する場合も，これらの物理指令それぞれを，希望する書式に合うかどうか確認しながら変更しなければならず，その手間は，エディタで書式制御を直接行ってしまっている場合の修正作業と大差なくなってしまう．

したがって，行間を1行空ける場合でも，章のタイトルの後だから1行空けるというように，個々の指令が前述の文書の階層構造と結びつけられていなくてはならない．さらに，それら物理的指令はその階層を形成しているテキストに直接結びつけるのではなく，その階層に共通に適用される複数の指令（それらをまとめたものを論理指令と呼ぶことにする）の中の1つとして組み込んでおいて，間接的に結びつけておかねばならない．たとえば先の例の1行空けは，「章タイトルの書式1」とでも名付けられた論理指令の中に「1行空ける」という物理指令が含まれており，（他には，「字体をゴシックにする」などが含まれている）文書中のすべての章タイトルには，上の論理指令が付随して

いるようにしておくというような具合である．この場合書式制御の一貫性は，明らかに保証されており，書式変更は論理指令中の物理指令の変更のみでできる．

このようにしておけば，論理指令の定義の中にしか物理指令が存在しないから，テキストと書式制御指令とは，事実上分離されており，書式制御指令だけを集めておくことが可能になる．これらの論理指令の定義の集合がすなわち書式である．したがって，その集合に書式名を対応させて保存しておき，清書時に書式名を指定してそれらを取り込むようにすれば自由に書式の変更が可能となる．たとえば，フォーマッタに「情報処理」と指定すると本誌の書式になり，別の名前を指定すると別の書式になるという具合にするのである．このような書式が多数用意されていれば，フォーマッタの利用はきわめて便利なものとなる．

フォーマッタは，多種類の書式に適用できるように基本的な物理指令が充実していると同時に，著者・タイピストが新しい書式を使用しなくてはならなくなった時に利用できるように，一般的な論理指令をも多数用意しておく必要がある．

3.2 フォーマッタの機能

書式を定義するためには，論理指令の内部構造が記述できなくてはいけない．単純な形として，具体的物理指令と，地の文（テキスト）が入る場所を示すだけの，ちょうどアセンブリ言語のマクロ定義のような方式が考えられる．すなわち，論理指令は，その結びつけられているテキストをパラメータとするマクロであると見るのである．ただし，たとえば章のタイトルが長すぎて2行に渡ってしまうというような特別な場合でも臨機応変に対処できる強力な論理指令を定義するために，テキストの長さや行数などを与える組込関数，ページ書式のパラメータなどを与えるグローバル変数，作業用に使うローカル変数，およびそれらを使った算術式や条件付の指令のコントロール機構（if とか while のようなもの）などが使えるものでなければならない．ここではマクロを考えたが，このような機能さえ持っていれば，テーブル駆動型や，プログラミング言語型でもよい．

このような書式定義機能を持つ現実のフォーマッタとして，NROFF/TROFF[4]と Scribe[5]を挙げることができる．前者は，物理指令をインタープリットするフォーマッタであるが，強力なマクロ機能を備え，論理指令にあたるものを構築できるようになっており，実際に使用する時には，直接物理指令を使うのではなく，一般の科学論文，テクニカルメモ，マニュアルなどの書式定義として用意されているそれぞれのマクロを使っている．

使いやすさの点で評判の高い後者は，出力テキストをコントロールする100近くの変数が作る環境に従って，清書を行うフォーマッタであるが，データベース中に，環境を設定する論理指令を定義しておき，テキスト中の論理指令名から環境を取り込む方式をとっている．

4. 原始入力エディタ

原始入力エディタとは，著者が作成した草稿を最初にタイピストが入力するのを支援するエディタである．この作業は，タイピストにとって最も能率を上げられるものであるが，草稿に記入されていない書式制御指令を推定して挿入しながらの作業では，その利点は失われてしまう．しかし，逆に著者がそれらすべてを明確に書かなければならないとすると，そのことが草稿作成時に負担となってよくない．そこで原始入力エディタが，草稿に自然に含められている書式情報を抽出し，自動的に指令を生成挿入するようにしたい．

現在，このような自動生成は技術的には未完である．しかし，3章に述べた論理的書式制御を前提とすると，テキスト中に指定しなければならない論理指令はその名前だけであり，それも文書の論理的階層構造に対応しているのであるから，実質的には階層構造を原始入力エディタに判別させるだけで十分自動挿入が可能となる．

構造の判別には，(1)テキストの形態から得られる一般的な情報，(2)草稿上に仮定された書式，(3)完成文書の書式，(4)特殊な制御を記述するための若干の約束事，などを使うことにする．(1)として，たとえば句点が文の終りであるというような基本的な事項の他，「図1」，「表 3.5」のような用語から，図・表の引用場所を知って，それらの配置に役立てることなどが含まれる．(2)，(3)としては，たとえば文の先頭に「(1) 」や「a)」が現れたら，それを列挙型書式であると判定するなどがあり，(4)としては，たとえば脚注の本文を決められた区切り記号でくくって，注をつける語の直後に置く約束にしておく，などが役立つであろう．

さらに文字の入るわくだけを完成文書の書式に合わせて表示しておいて，その中に入力文字を順次表示していくようにすると，タイピストはエディタが仮定し

表-1 校正支援のための統計項目とその目的

統計項目	目的
(1) 漢字	常用漢字表外の漢字の検出
(2) 熟語	誤字，当て字，難読語の検出
(3) 送りがな	送り方の不統一の検出
(4) 同一発音語	表記法の不統一の検出
(5) 文末の型	文体の不統一の検出
(6) 特殊用語	いい回しのアンバランスの検出

ている書式が見えるので安心できるし，書式の誤認も速く検出できる．

このように，具体的書式制御指令を混ぜないでフォーマッタを使おうとする試みとしては，前述のTROFF/NROFF の前処理として開発された NPP[6] がある．NPP では行の先頭と末尾に通常現れることのない文字を書くことでセンタリングなどの制御を指示したりするようになっている．

NPP は大量のマニュアルを入力する際に大いに有効であったということであるが，文書書式を限定することができれば，無指令の原始入力は大いに有望であるといえよう．

5. 校正支援プログラム

英文のシステムにおいて綴りのチェック用に用いられている Spell[4]のように，日本文の文書処理システムにおいても漢字辞書・国語辞典を利用して **表-1** のような項目についての統計検査プログラムを用意しておくと，校正作業への大きな支援となる．

実際にこのような統計検査による支援を行っているものとしては，かな漢字変換を基本とする文書処理システム「ことだま」[7]がある．このシステムでは，かな漢字変換の過程において得られた漢字の読みや語の区切りを残しておいて，QWIK, QWOK をとることによって，表-1 の(2)，(3)，(4)の目的を達している．

6. おわりに

入力が難しいとされている日本文では，英文以上にタイピスト経由で文書処理が行われるべきである．本稿では，そのタイピストの操作性に重点を置き，(1)原始入力にあたって入力に専念できるように，書式制御を無指令型にし，(2)著者が書式名を指示するだけで，望みの書式が得られ，また簡単に変更できるような論理型のフォーマッティングにし，(3)著者の訂正を直接反映する操作で編集校正ができるようにすることなどを強調してきた．

しかし，日本文の文書処理が途についたばかりで，社会的環境が整っていないために，タイピスト経由の間接的な文書処理システムの利用については欧米ほどの理解は得られていないように思われる．また技術的に未熟な部分もあり，マンマシンインタフェースとして今後の一層の研究が必要である．

紙数の制限のため，フォーマッタの物理指令について具体的に触れることができなかったが，基本的には英文のそれと大差はない．文献 8) がそれらを手ぎわよくまとめており，そちらを参照されたい．

最後に，本論説を書くにあたり，有益な助言を下さった，日本 IBM の藤崎哲之助氏に感謝いたします．

参考文献

1) Yamada Hisao: A Historical Study of Typewriters and Typing Methods: from the Position of planning Japanese Parallels, JIP, Vol. 2, No. 4, pp. 175-202 (1980). 同翻訳 (小笹和彦), bit, Vol. 13, No 7-11, 13, 共立出版 (1981).
2) 山田尚勇：日本語テキスト入力法の人間工学的比較，プログラミングシンポジウム「日本語情報処理」報告集，情報処理学会 (July, 1978).
3) Joy, W. and Horton, M.: Ex Reference Manual, UNIX Programmer's Manual, 7-th Ed., Vol. 2C, Computer Science Division, UCB (1980).
4) Kernighan, B. W., Lesk, M. E. and Ossanna, Jr, J. F.: Document Preparation, Bell Syst. Tech. J., Vol. 57, No. 6, Part 2, pp. 2115-2135 (1978).
5) Reid, B. K.: A High-Level Approach to Computer Document Formatting, 7-th ACM Symposium on Principles of Programming Languages, pp. 24-31 (Jan. 1980).
6) Barach, D. R. and Fram, D. M.: NPP—An Easy to Implement Preprocessor for Text Formatting, Softw. Pract. Exper., Vol. 10, No. 5, pp. 335-346 (1980).
7) 藤崎，大河内，諸橋，戸沢：日本語文書処理システム（ことだま）—概念と設計思想—，日本 IBM サイエンティフィックセンターレポート N: G 318-1512 (June 1980).
8) 河田勉，天野真家：日本語のワードプロセッシング，情報処理，Vol. 21, No. 8, pp. 894-901 (Aug. 1980).

(昭和57年3月5日受付)