日本語テキストフォーマッタの試み

松房一郎・大岩　元
（　豊橋技術科学大学　）

１．　まえがき

いわゆるワードプロセッサの普及に伴い、日本語を入力しプリンタに打ち出すことは比較的容易になった。しかし文書のレイアウトを整えるためには、ワードプロセッサのファンクションキーを操作したり、必要な空白を挿入したりという手作業が必要になる。そのため文書のレイアウトを変更しようとすると、行長をすこし増すだけでも大変な量のキー操作が必要となる。

欧文用には Scribe[1]、TEX[2]や UNIX の nroff/troff[3] といったバッチ式のテキストフォーマッタがある。これらは、レイアウト変更が容易であるばかりでなく、レイアウトのライブラリを持っているので、文書の作成時には具体的な体裁を考えることなく入力に専念できる。

一方日本文用のフォーマッタは汎用計算機上のものは欧文用を日本語化したものを含めて少し試みられている[4][5][6] だけで、広く用いられるには至っていない。また電算写植機に附属するフォーマッタは十分な機能を持っているが、そのコマンドが低レベルすぎるので専任者（コーダ）しか使いこなすことができない。

そこで今回試作したフォーマッタ（JFORM と呼ぶ）には次のような特徴を持たせた。

(1) Scribe風の環境指示をコマンドとしたので、文字の大きさ・行送りなどの体裁の指示及び処理が簡単になった。
(2) 図・表・上下余白・脚注などを直接表現するデータ構造（領域）を導入した。
(3) 文字の位置決めを行なうミニフォーマッタをフォーマッタ内部で用い、必要に応じて同じページを再処理することで、ページレイアウト処理を自動的に行なえる。
(4) 出力ドライバを交換することで各種のディスプレイ・プリンタ・写植機に、整形した文書を出力できる。

２．　英文フォーマッタと日本文フォーマッタの違い

先に述べたように、欧文用のフォーマッタには実用にされているものもかなりある。これをそのまま日本文のフォーマットにも使うことができればよいが、日本文と欧文の間には次のような違いがある。

１）　英文

単語の途中では決まった位置でしか改行できない（ハイフネーション）。

文字によって幅に違いがある。同じ文字でも書体によって幅が違う。

単語の間の空白量を調整して行の右端を揃える。

２）　日本文

原則としてどこで改行してもよい。ただし特定の文字は特定の場所（行頭・行末等）にきてはならない。

文字の大きさは同じであるから、何もしなくても右端は揃う。欧文が入ったなどで右端が揃わ

なくなった時は、文字間をあけるか字の幅を小さくするなどして調整する。
英文と同じ横組みの他に縦組みがある。

これらの違いは全て組版規則、それも主としてどのように段落を行に分割するかに関するものである。それ以外の点ではフォーマッタに要求される機能は共通している。なお日本文のフォーマットにあたっても単語の形あるいは引用として、欧文のフォーマットが要求されるのでこれにも対応しなければならない。

3．　フォーマッタの使いやすさ

使いやすいフォーマッタの条件として、次の３つが考えられる。

1）　フォーマッタへの入力（文章と体裁指示の混じったもの－原稿）が作りやすいこと

2）　作った原稿から文章の内容が読みとりやすいこと

文章の内容を変更したり、チェックしたりするときに体裁指示（コマンド）が目障りになると、内容が理解しにくい。このようなことにならないように、コマンドの構文には十分注意を払う必要がある

3）　レイアウト変更がしやすいこと

レイアウト変更が必要な状況は２通り考えられる。

（1）　標準レイアウトからの変更

文書を作ろうとするときまず試してみるのは、マニュアルに例として示されたレイアウトや普段使っているレイアウトなどの整形結果のわかっている標準的なレイアウトである。しかしそれが気に入らず、すこし違うレイアウトを使おうとしたときに、レイアウトの修正が簡単にできなければ困る。このためにはレイアウト指示の記述が理解しやすい表現になっている必要がある。

（2）　入力済原稿からの変更

レイアウトの変更は単に出力する用紙の大きさを変えるだけでも必要になる。いったん原稿を入力し、あるレイアウトで出力した後で、レイアウトを変更しようとするときに、原稿中のコマンドを全部エディタで修正しなければならないとなれば、誰もそんなことはしたくないだろう。

このためには原稿中には、文の大きさや行送り量といった具体的な体裁指示を入れないで済ませられることが必要である。

以上の条件を満足するためには文章中に入力しておくコマンドは、手続き形の具体的な整形指示ではなく、Scribe風に、文書のその部分はいったいどういう性格のものであるか（見出し、段落など）を記述するものがよい。

また脚注や図など、「なりゆき」でそのページ内の位置が決まるものの処理の記述は、入力文書中に記述すべきでない。これは条件とその場合の処理という形になるので、文書の入力とは独立してプログラムの形で記述したほうが解りやすい。

4．　フォーマッタに要求される機能

4．1　字種の変換

商用の電算写植機は扱う文字すべてをそのまま入力できるが、それ以外のフォーマッタの場合は入力できる文字に制限があって、たとえばギリシア文字や数学記号は出力できても入力できない。また文字として扱われる空白（単語の区切りにならない）といった特別な機能を持つ文字も入力する必要がある。こういった文字は特殊記号との組み合わせ等によって表現されるので内部コードへ変換しなければならない。

4．2　段落を行に分割する

入力にはどこで改行するかをいちいち示していないので（普通、入力ファイルの中にある改行コードは改行指示として処理しない）、指定された行長におさまるように適当な所で改行する。

4．3　ページアップ

上の処理でできた行を適当な所で切ってページにする。

このとき上下の余白をとる、余白部分にページ番号や表題を入れる、脚注を入れるなど、ページとしての体裁を整える処理も必要である。

4．4　カウンタ

ページ番号を自動的にカウントしてヘディングあるいはフッティングに表示する機能はたいていのフォーマッタにある。これと同じように図・表や章の番号をフォーマッタが自動的にカウントし本文や見出しに挿入する機能があれば、文章の内容に追加があった時にも番号をいちいち付け換えなくてすむ。

4．5　索引・参考文献リストの自動作成

索引はその語句が文章のどの位置（ページ・行）にあらわれたかを示すものである。これを手作業でやろうとすれば、全ページが出力された後で語句がどこに現われたかを誤りも見落としもないように捜し出して、索引用の入力ファイルを作成しなければならない。したがって本文の修正があると位置がずれるので、全作業をやり直すことになる。一方索引の対象となる語句に印をつけておくことさえできれば、これを自動化するのはさほど難しくない。

参考文献リストもそのデータをデータベースから持ってくるのは、むしろ計算機の得意とする作業である。

以上の機能のうち、２と３だけが文字の位置決めに関するものであり、１・４・５は入力原稿を変更するだけの処理である。JFORM ではこの点に注目して整形処理を、文字の位置決めを行うミニフォーマッタとミニフォーマッタへの入力を作る入力スキャナの二つに分けることにした。

4．6　ユーザ拡張

文書の書式には、さまざまな多様性があり、したがってフォーマッタの機能も多様化しなければ

ならない。もしその多様性に対応することができなければ、それはすなわち機能の不足であり、そのフォーマッタは使われなくなるであろう。しかしコマンドの設計時に全ての可能性を考えておくのは不可能であるから、ユーザがフォーマッタの機能を追加あるいは変更できるようにすることは重要な機能である。

また機能が多様化すると、コマンドやパラメータが複雑になり、原稿中のコマンドの量が増す。その結果として原稿が解りにくくなるから、原稿に入れるコマンドを減らす工夫が必要になる。

フォーマッタに関する多様性には次のようなものがある

(1) 表現の多様性

脚注や参考文献の引照記号をどうするか。

見出しの大きさや行数など。

(2) 漢字の表現

JIS コードかシフトJIS コードか。JIS の場合１バイトと２バイトのコードを切り換えるシフトコードは何か。

(3) 禁則事項に関する多様性

拗促音（ゃゅょっ等）を対象とするかどうか等の禁則条件。

追込むか、追い出すかなどの対策に関するもの

(4) 出力に関するもの

出力デバイスにルビやセンタリングなどの機能があるかどうか。（もしあれば、フォーマッタではその処理をしなくてもすむ。）

コマンドとそのパラメータの意味と構文、文字コードが機種によって異なる

このうち(1)(2)は入力スキャナの機能の問題である。(3) はミニフォーマッタをユーザが拡張できるようにして対応する。また(4) に対応するためにミニフォーマッタの処理結果を出力する出力ドライバを導入する。

５．　ＪＦＯＲＭのデータ構造（文書モデル）

試作したフォーマッタ JFORMの内部では文書の各ページを、次の３つのデータ構造により表現している。

１）　文字セル

文書の各文字に対応する。文字のコードの他に、書体、大きさ等の属性を持つ。原稿（入力ファイル）の内部表現は文字セルの配列である。なお書体番号０はコマンドを表わす。

２）　行

行を構成する文字セルを表わすための左端・右端の文字セルへのポインタと行の位置からなる（図１）。

３）　領域

図・表のための空白部分、上下左右の余白（マージン）、段、脚注などを表わす。領域の内部に別の領域があってもよく、ページも一つの領域として扱っている（図２）。領域の属性には、位置・大きさ・強さ（優先順位）などの他に、領域内の文字を示すための行へのポインタもある。

領域の優先順位は、段と脚注など領域同士が重なった場合にその部分がどの領域に属するかを決定するためのものである。

6．　ＪＦＯＲＭの構成と動作

このJFORM は入力スキャナ・ミニフォーマッタ・出力ドライバの３つからできている。（図３）

6．1　入力スキャナ

入力スキャナはコマンド（合図）と文字（文章）の混在した入力を読み込み、内部表現である文字セルの配列にセットする。図や脚注などページのレイアウトに関する処理も入力スキャナが行なう。

6．2　ミニフォーマッタ（文字配置ルーチン）

図や脚注など他の領域に文字が入らないように、行の長さを計算しながら文字セルを行に分割（文字がどの行に属するか決定）し、行で文字領域を埋めて行く。フォーマッタの本質的な処理はここで行なわれる。

入力コマンドによる指示の大部分は行送り量やインデント量といった変数の値となってミニフォーマッタの動作を制御することになる。なお入力コマンドの指示のうち文字の大きさなど一部のものは、入力スキャナによって文字セルの属性に変換されている。

6．3　出力ドライバ

一ページ分の文字をまとめて出力する。

両端揃えやセンタリングで必要になる行内の文字の位置決めはここで処理する。

6．4　JFORM の動作

入力スキャナが入力原稿を読みながら本文用の文字セルをセットしてゆく。文字セルが適当な長さになると、入力スキャナはミニフォーマッタを呼び出して行への分割を行わせる。入力に脚注の指示が現われると、入力スキャナはその文章を脚注用の文字セル配列にセットしてミニフォーマッタを呼ぶ。

ミニフォーマッタが作り出した行でページが埋まると入力スキャナは出力ドライバを起動する。出力ドライバによって、ページの各行は指示通りに揃えられて出力される。

6．5　再フォーマット

現在の印刷の現場では原稿と共に各ページのレイアウト（主として、図の位置）も客先より指示されることが多い。図の位置が解っていればレイアウトに従ってインデントや行長を変更して空白を確保するだけでページアップが可能である。

しかし JFORMでは、文章中に図の大きさを示すだけで図の位置もフォーマッタが決定しようとすることも行なう。もし図の位置がすでに文字のある部分に重なった場合、つまり空白が文章の途中に割り込んだ場合には、その空白の位置からフォーマッティングをやりなおす必要がある。（図４）

再フォーマットは、必要がある領域に印をつけておくとミニフォーマッタが呼び出された時に自動的に行なわれる。

7．　まとめ

使いやすいフォーマッタの条件として、(1) 入力原稿が作りやすい、(2) 原稿の内容が読みやすい、(3) レイアウトの変更が容易、(4) ユーザがプログラムを記述できることなどを考えて、フォーマッタを試作した。現在 JFORMは UNIX 上にＣで書かれている。ユーザ拡張もＣで記述しているため拡張機能自体はあまり実用的でないが、文書の整形機能自体は満足できるものである。

今後はユーザ拡張を高レベルで記述できるように改良を行なってゆきたい。

参考文献

[1] Reid, B. K. and Walker, J. H.: "Scribe User's Manual (Third Ed.)", Unilogic, Ltd. (1983).

[2] Knuth, D. E.: "TEX and METAFONT, a new direction in typesetting", American Mathematical Society and Digital Press(1979).

[3] Ossanna, J. F.: "NROFF/TROFF User's Manual", UNIX Programmer's Manual.

[4] 藤田　博: " 技術文書整形出力システム: TEX", 情報処理, Vol.25, No.8, pp. 848-853(1984).

[5] Takahashi, N. et al.: " 浄書: Japanese Output Server with HOspitality", Proc. of ICIP '83(1983).

[6] 角田博保: "移植性の高い和文文書フォーマッタの一構成について", 情報処理学会第28回全国大会予稿1M-8(1984).

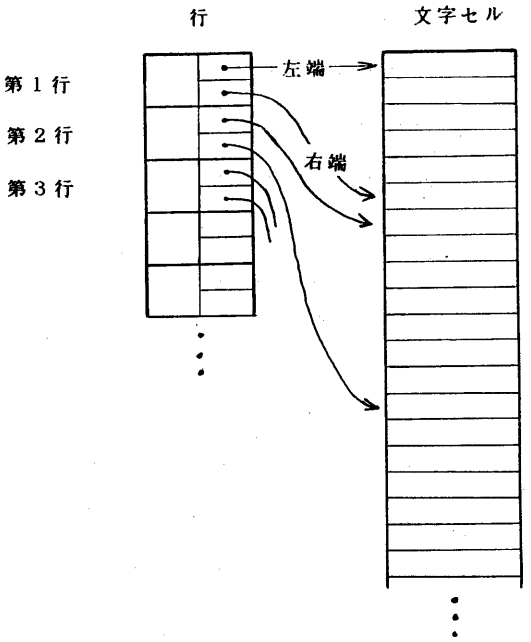

図１　行と文字セルの関係

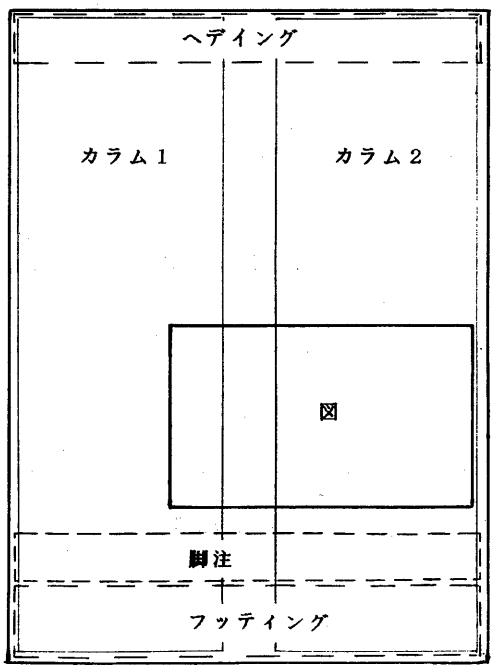

図２　ページ内の領域

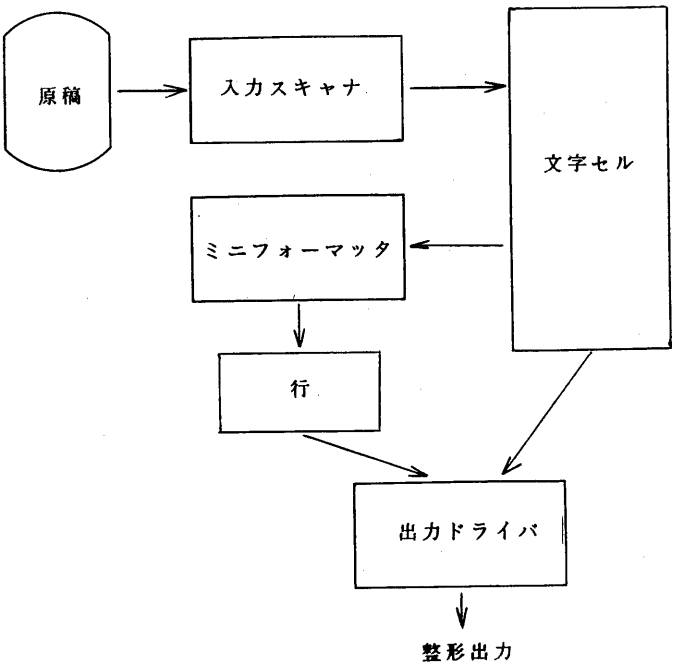

図３　JFORM の構成

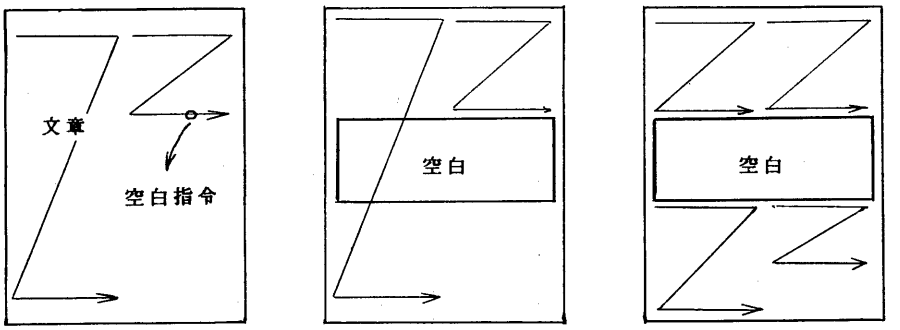

図４　再フォーマット