解　説

3. DTPの要素技術—技術動向の現状を知るために—

3.2 文書整形システムの現状と将来†

斉　藤　康　己††

1. はじめに

現在のDTPシステムは，プリンタ，ビットマップのディスプレイ，高品質のフォントそして各種のソフトウェアを統合したかなり総合的なシステムになりつつある．システムがサポートする機能もアイデアの発想から始まって最終的な出版物の仕上げまで，いわゆる出版という活動の全体をカバーしようという方向に動きつつある．

しかし各メーカがこうしたDTPシステムを発表し出したのはごく最近のことで，それに至るまでに多くの技術の蓄積があった．特に計算機科学の分野では，かなり昔から，計算機を文書の作成に使うことが試みられ，多くのシステムが作られ，進化してきた．

そうしたシステムの中でも最も基本的で，DTPシステムを作り上げるときの核になっているのがエディタとフォーマッタ（文書整形システム）である．大まかに言えば，エディタは計算機へのテキストの入力を受け持ち，文書整形システムは計算機からの出力の面倒をみる．3.1ではエディタを取り上げたので，ここでは後者の文書整形システムを取り上げて，その現状と問題点，そして将来の姿を考察する．

2. 文書整形システムとは

文書整形システムの基本的な仕事は文書の紙の上への配置（レイアウト）である．ユーザがエディタを使って作成したソースファイル（この中に文書の中味とそれをどのように配置したいかを記述した指示とが含まれている．）を入力として受け取って，それをプリンタに出力できる形に変換するのが文書整形システムの役割である．もう少し正確に言うと，最終出力はプリンタドライバ（文書整形システムの出力をプリンタが解釈できるコード列に変換するプログラム）とかプレビューア（ビットマップの画面上で出力イメージを見るためのユーティリティ）とかの一応独立したソフトウェアが行うことが多い．そこで，これらのソフトウェアが処理できるような「文書の物理的配置の記述」を含んだ中間ファイルが文書整形システムの通常の出力になる（図-1）．

二次元の紙の上へのレイアウトというとマウスを盛んに使うお絵描きプログラムや新聞のレイアウトシステムなどを思い浮かべる人もいるかもしれない．これらの図形を主に扱うシステムと文書整形システムとの大きな違いは，配置する対象が図形ではなく文書であるという点にある．もちろん後で述べるように図形の取り扱いは大きな問題であるが，それはあくまでも文書の中に図形を埋め込むのであって，処理の第一の目的は文書の配置である．

文書は言うまでもなく文字の列から構成されるが，3.1でも述べたように段落，章といった構造をもっていて，それをページの上に配置するにあたっては，いろいろの約束ごとが存在する．段落の初めは一字下げるとか，各行の右は揃えるとかいうのがその例である．こうした約束を守りつつユーザの指示に従って，文字の配置を決定していくのが文書整形システムの主な仕事である．これに図形，写真，表，数式，化学式などといういろいろな要素が加わって文書整形の仕事は複雑になっていく．

文書整形システムの機能を列挙してみると，以下のように基本的なものから，応用的なものまで千差万別である．

（1） 文字のフォントを選択すること
（2） 文字を並べて行を形成すること
（3） 行を積み重ねてページを形成すること
（4） 水平/垂直方向に文書の位置を揃えること
（5） 表，数式，図などを作成し配置すること
（6） 番号付け，目次，索引，文献表などの作成

† Document Formatting Systems—Present and Future by Yasuki SAITO (NTT Software Laboratories).
†† NTTソフトウェア研究所

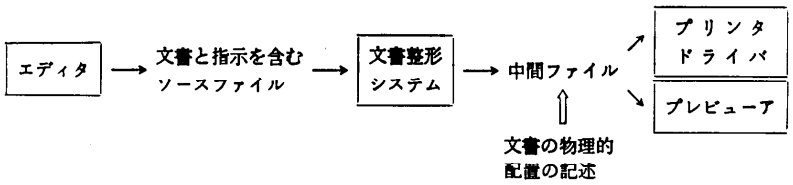

図-1 文書整形システムの入力と出力

文書整形システムでは，このような要求の多様性に対処するためにマクロなどを使った拡張機能やカスタマイズの機能をそなえているものが多い．この拡張機能の使い良さがシステムのカギを握っていることについては後で触れることにする．

次章で具体的なシステムの解説と検討に入る前に，ワープロと文書整形システムの違いに簡単に触れておきたい．ワープロはここでの分類に従えば，エディタと文書整形システムとプリンタドライバの三つのプログラムを合わせたような機能をもつ．しかしながら，文書整形の機能は不十分なものが多い．一つには，日本語の簡単な文書を主な対象にしているために，固定幅の文字をただベタベタと並べていくだけというのが基本になっていること．数式，索引など高度な機能に乏しいこと．また，日本語に特化してしまって，欧米語の整形に必要な機能を十分には考慮していないことなどが原因である．ただし，一つだけ特筆に値するのはワープロが後でも述べる WYSIWYG 方式（What You See Is What You Get の略で，画面上で見えるものが，そのまま出力イメージになっている方式）にほぼなっている点である．

3. 代表的な文書整形システム

ここでは，かなり長い歴史をもっていて，かつ現在も広く使われているシステムの中からそれぞれ特徴的な三つのシステムを取り上げて解説を試みる．文書整形システムのサーベイや解説はすでに数多く存在するので[1)~5)]，ここではなるべく重複をさけて，以下の議論に必要となる要点のみを例を使って概観することにする．これら三つの典型的なシステムを軸としてすべての文書整形システムの空間を張ることができるわけではないが，必要な論点はこれで，十分明らかになると思われる．

3.1 UNIX 環境での文書整形システム

troff, nroff, ditroff など総称して roff と呼ばれる文書整形システムが UNIX にはある．これらのおおもとは J. H. Saltzer の RUNOFF[6)] である．troff はタイプセッタ用の出力を出し，nroff は通常のタイプライタふうのプリンタへの出力用，ditroff は各種レーザビームプリンタの出現に対応して現れた “device independent troff” というわけである．これらはどれも似ているので，ここでは troff を例にしてその概要をみることにする[7)]．

roff の仲間の大きな特徴は表，数式，図形などの処理を一応 roff とは独立のプリプロセッサで処理することである．tbl, eqn, pic と呼ばれるプリプロセッサがあって，それぞれ表の作成，数式のタイプセット，図形の記述を担当する．といっても，これらはどれも前処理の結果として roff のコードを生成するので，全体で一つのシステムと考えたほうがいい．また，グラフを作るための grap とか，文献の作成を容易にする refer というプリプロセッサもある．

roff の基本的なコマンド（request と呼ばれ，ピリオドで始まる 1 文字または 2 文字の名前をもつ）は全部で約 90 個で，それほど多くはない．このうち約 1/3 が本当の意味でテキストを整形するためのコマンド（たとえば “.ce” はセンタリング，“.fi” はいわゆる追込み，“.ad” は右揃えといったコマンド）で，残りはフォントを選んだり，条件文を作ったり，各種のパラメータを設定したりするコマンドである．

まず図-2 のソースファイルと図-3 の出力例を見ていただきたい．これは UNIX (4.2 bsd) に標準についてくる ms マクロを使って記述したものである．roff ではコマンドのあとに引数がつく場合もつかない場合もある．テキストの中でバックスラッシュ (\) で始まっている文字列はエスケープ・シーケンスとよばれ，これも簡単なコマンドで 40 種類ほどある．エスケープ・シーケンスはフォントを切り替えたり，特殊文字を指定したり，いろいろな長さのスペースを挿入したりするのに使われる．

小文字のコマンドは roff 本来のコマンド，大文字のコマンドは主に ms マクロパッケージの中で定義されているマクロである．たとえば，図-2 に出てくる大文字コマンドである “.PP” (字下げ付きの段落の書

```
.pl 555p
.nr PI 20p
.nr PD 2p
.ds CF %                    \" ページ中央下のページ番号用
.ds tx T\h'-1.667p'\v'2.15p' E\v'-2.15p'\h'-1.25p'X \" TeX ロゴ用のストリング
.TL                         \" タイトル
Sample Document
.AU                         \" 著者
by Yasuki Saito
.MC 192p 26p                \" 二段組の指定，一段の横幅と段と段の間のスペース
.PP                         \" パラグラフの開始
This is a very simple sample document, yet full of commonly used constructs
in everyday typesetting. Some of the exhibited features are listed below:
.in +24p                    \" 24 ポイントの段下げ
.LP                         \" 字下げなしのパラグラフの開始
(1) narrow paragraph
.LP
(2) table
.LP
(3) mathematical formulae
.in -24p                    \" 字下げをもとに戻す
.PP
First of all, let's see how to make narrower paragraph:
.PP
.SM                         \" フォントを小さいもの (small) に変更する
.in +20p                    \" 20 ポイントの段下げ
.ll -20p                    \" 行の長さを 20 ポイント短くする，これで狭い段落になる
.vs 10p                     \" 行間隔を 10 ポイントに設定
This paragraph is narrower and typeset using smaller font.
Also, this paragraph contains lots of special characters such
as \o''a\'', /o''e'', \o''o^'', , , \o''A\(de'', , \(dg, \(sc, , \(co and \*(tx.
.NL                         \" フォント・サイズを通常 (normal) に戻す
.ll +20p                    \" 行の長さを戻す
.in -20p                    \" 字下げもリセットする
.PP
Next is a table of typesetting systems treated in this article:
.TS                         \" 表 (テーブル) の開始
allbox, center;             \" 全ての項目を箱で囲み，全体をセンタリングせよ
c | c | c                   \" 一行目用のフォーマット，'c' はセンタリング
l | l | r.                  \" 残りの行のためのフォーマット，'l' は左詰め，'r' は右詰め
System Author University
RUNOFF J. H. Saltzer MIT
Scribe B. K. Reid CMU
\*(tx D. E. Knuth Stanford
.TE                         \" 表 (テーブル) の終わり
.PP
And finally comes two mathematical formulae:
.EQ                         \" 数式モードの始まり
delim $$                    \" テキスト中の数式のデリミッタとして '$' を使う
.EN                         \" 数式モードの終わり
The $n$th Fibonacci number $F sub n$ is given by*:
.FS *                       \" 脚注の始まり
A sample from the ''joy of \*(tx''.
.FE                         \" 脚注の終わり
.EQ (1)                     \" 数式モードの始まり，数式番号として (1) を使う
F sub n={{left ( {{1+{sqrt 5}} over 2} right) sup n}-
{ left ( {{1-{sqrt 5}} over 2} right) sup n}}
over {sqrt 5}
.EN                         \" 数式モードの終わり
And this is an example with integrals**:
```

```
.FS **                          \" 脚注の始まり
A sample from the "\*(txbook".
.FE                             \" 脚注の終わり
.EQ                             \" 数式モードの始まり
left ( int from {- inf} to inf e sup {-x sup 2} dx right) sup 2
mark=int from {- inf} to inf int from {- inf} to inf
e sup {-(x sup 2+y sup 2)} dx dy
.EN                             \" 数式モードの終わり
.EQ                             \" 数式モードの始まり
lineup=int from 0 to {2\(*p} int from 0 to inf e sup {-r sup 2} r dr d\(*h
.EN                             \" 数式モードの終わり
.EQ                             \" 数式モードの始まり
lineup = int from 0 to {2\(*p} left ( left " " -{{e sup {- r sup 2}} over 2}
right | sub {r = 0} sup {r = inf} right ) d\(*h
.EN                             \" 数式モードの終わり
.EQ (2)                         \" 数式モードの始まり，数式番号として（2）を使う
lineup=\(*p.
.EN                             \" 数式モードの終わり
```

図-2 troff への入力ソースファイルのサンプル

Sample Document

by Yasuki Saito

This is a very simple sample document, yet full of commonly used constructs in everyday typesetting. Some of the exhibited features are listed below:

(1) narrow paragraph

(2) table

(3) mathematical formulae

First of all, let's see how to make narrower paragraph:

> This paragraph is narrower and typeset using smaller font. Also, this paragraph contains lots of special characters such as à, é, ô, ¸, Å, ¿, §, © and TEX.

Next is a table of typesetting systems treated in this article:

System	Author	University
RUNOFF	J.H.Saltzer	MIT
Scribe	B.K.Reid	CMU
TEX	D.E.Knuth	Stanford

And finally comes two mathematical formulae:

The nth Fibonacci number F_n is given by*:

$$F_n=\frac{\left(\frac{1+\sqrt{5}}{2}\right)^n-\left(\frac{1-\sqrt{5}}{2}\right)^n}{\sqrt{5}} \qquad (1)$$

And this is an example with integrals**:

$$\left(\int_{-\infty}^{\infty} e^{-x^2}dx\right)^2=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-(x^2+y^2)}dxdy$$
$$=\int_0^{2\pi}\int_0^{\infty} e^{-r^2}rdrd\theta$$
$$=\int_0^{2\pi}\left(-\frac{e^{-r^2}}{2}\Big|_{r=0}^{r=\infty}\right)d\theta$$
$$=\pi. \qquad (2)$$

* A sample from the "joy of TEX".
** A sample from the "TEX book".

図-3 図-2 から得られた troft の出力サンプル
[実際には itroff を使い IMAGEN(300 dpi) に出力したもの]

き始め）の定義は図-4 のようになっている．一般にはマクロは9つまでの引き数を取ることができ，その引き数を "\$n" という形でマクロ定義の本体の中で参照できるが，図-4 の例は引き数なしのマクロである．

一つ一つのコマンドを逐一説明していくのはここでは避けるが，図-2 や図-4 には必要なコメント（"\'" から行末まで）を入れておいたので，それと出力を見較べればおのおのがどのような意味かはだいたい分かるようになっている．いくつかのコマンドをかいつまんで説明しよう．

".nr" は変数 (number register) の宣言およびそれへの初期値の設定コマンドである．図-2 の例の中では "PI" および "PD" という変数の値を変更するのに使われている．これらの変数は ".PP" や ".LP" コマンドの定義（図-4 参照）の中で字下げおよび段落間のスペースの量として使われている．

".ds" はストリング変数の宣言および初期値設定

```
.\"  PP - regular paragraph    \" これはコメント行.
.de PP                         \" PP というマクロの定義開始.
.RT                            \" RT(reset) という別のマクロを呼ぶ. これは
                               \" 段落の始めで各種パラメータを初期化する.
.if \\n(1T .sp \\n(PDu         \" 1T という変数の値が正ならば垂直方向に
                               \" PD ユニット分のスペースをあける.
.ne 1.1                        \" 1.1 行以上の行を確保する (need コマンド).
                               \" いわゆる 'orphans' (ページの最後に一行だけ
                               \" 次の段落の先頭行がきてしまうこと) を防ぐ.
.ti+\\n(PIu                    \" 一時的に PI ユニットの字下げを行う
                               \" (temporary indent コマンド).
..                             \" これで定義終わり.
```

図-4 .PP の定義

(define string) である. 図-2 の 4 行目の ".ds" は各ページの一番下の中央に印刷すべき文字列 ("CF") として, 自動的にページ数におき変わる特殊な文字: '%' を設定している. 次の ".ds" は TEX というロゴを水平および垂直移動のコマンドを使って作り, それに "tx" という名前をつけている.

表は ".TS" と ".TE" で囲まれた部分に記述があり, 数式は ".EQ" と ".EN" で囲んである. これらはそれぞれ前に述べた tbl, eqn と呼ばれるプリプロセッサで前処理されて roff のコマンド列に変換される. tbl で処理される表作成用のコマンドは直観的で分かりやすい. ただし, あまり細かい制御まではできない. また脚注には ".FS" と ".FE" を使う. これらもすべて ms パッケージの中で定義されたマクロである.

roff の基本コマンドは上でみたようにかなりプリミティブなものであるが, マクロパッケージという形でマクロの機能をフルに活用してドキュメントの内容に応じたより高度の処理を実現していることがよく分かる. あと二つだけ roff に特徴的なコマンドを説明しておこう. diversion と trap である.

diversion は, 処理済みのテキストの一部を出力せずに名前のついたレジスタ (あるいはバッファ) に一時的に格納する機能である. これにはマクロと同じように二文字以内の名前がつき, あとで必要なところでそれを取り出すことができる. 典型的な使用例は脚注の作成や, 多段組の作成, 段落の途中での改ページの回避などである. 格納されたテキスト全体の幅および高さはそれぞれ 'dl' と 'dn' という変数に納められるので, これを利用していろいろな条件を計算することができる.

trap は非常に一般的な機能で, 現在処理中のテキストの出力がページ内の指定した位置を越えた瞬間に特定のマクロを起動するというものである. これを page trap という. このほかに, diversion trap と input-line-count trap という二つの trap がある. 前者は diversion でレジスタに格納中のテキストの高さが指定した位置を越えたときに trap が起動され, 後者は指定した行数の入力行が読み込まれた直後に trap が起動される. trap はいろいろな目的に利用されるが, 改ページごとのパラメータの再設定, 脚注の挿入などが典型的な使い方である.

roff のコマンドは今から考えるとまったく非人間的な 2 文字という制限のために, まるで判じ物のように分かりにくい. 私のように時々しか使わないユーザはいつもマニュアルと首っ引きである. さらに悪いことには変数も 2 文字までと制限されているので, 変数名を見ただけではそれが何を意味する変数かまったく見当がつかない. しかしマクロの機能はかなり強力なので多くのユーザはエキスパートが開発したマクロ・パッケージを使うことで満足しているようである. UNIX には ms マクロのほかにも me マクロ, man マクロ (UNIX のマニュアル用のマクロ), System V には mm マクロなど各種のマクロが揃っている. それぞれのマクロの詳細にはここでは触れないが, それぞれタイトルとか, 段落とか文献とか目次といった意味のあるかたまりを自動的にタイプセットしてくれるように作られている.

最後にフォントであるが, roff では標準でついてくるフォントのセットはそれほど豊富ではない. 通常は標準のフォント (Times Roman) のほかにボールドとイタリック, それに特殊記号用の 4 種類のフォントがあるのみである. 特に数式用の記号類は貧弱でそのことが図-3 の出力例にも現れている. (たとえば積分の記号を見よ.) ditroff などではプリンタ内蔵のフォントも使えるようになっているので, フォントの選択の幅は広がるが, それでも限りがある.

3.2 Scribe

Scribe* は，1978年ごろにカーネギー・メロン大学の学生であった Brian K. Reid が開発した文書整形システムである．roff などとはその設計思想上大きな違いがある．それは，一言で言えば，「ユーザには細かな制御はさせない」という徹底した姿勢である．もちろん Scribe にもカスタマイズおよび機能拡張の手段はあって，新しいフォーマットを定義することも可能なのだが，Scribe はあまりそうした改良を奨励しない．

マニュアル[8]を読むと，「Scribe が用意した書式で満足しろ」とか「大切なのは書式に凝ることではなくて，書く中味である」などと書いてあってユーザがあまり深みにはまらないように警鐘を鳴らしている．それでも，どうしても手を加えたい人は Scribe データベースの中をいじることになる．そしてそれにはマニュアルよりさらに分厚い Srcibe Database Administrator's Guide[9] が必要となる．このガイド片手にユーザの要求を実現してくれる DBA (Database Administrator) が各サイトに一人いれば，ユーザはその人に頼めばいいから，細部を知る必要はないというわけである．

Scribe は roff とは違いプログラミング言語的ではない．ユーザは細かなタイプセットの仕方をシステムに指示することはなく，単に「これは論文だ」とか，「ここは段落だ」といったレベルで文書の種類を宣言するだけでいい．あとは，その種類に適した（とScribe の開発者が考えた）書式でユーザの文書を整形してくれる．

この前もって用意された書式のことを document type と呼ぶが，Scribe には**図-5** に掲げる 10 数種類の document type が揃っている．これらの詳細は Scribe データベースに記述されていて，若干のパラメータはユーザレベルで変更することができる．たとえば，letter type には Address, Body, Greeting などという環境（いわゆるブロックに相当する）が用意してあって，ユーザはその環境名を指定して，そこにその中味を書きさえすればよい．

まずは，roff のところで使ったのと同じ例をタイプセットするためのソースファイル（**図-6**）およびそれからの出力（**図-7**）を見てみよう**．

図-6 を見てもらえば分かるように Scribe で記述されたソースファイルは大変に読みやすく，分かりやすい．"@" で始まる文字列がコマンド，あるいは宣言で，どれも自然な英語が使ってあるので，意味は一目瞭然だと思う．

最初の行は document type としてはデフォールトの "text" を使えという指示である．2行目の "@style"

Article, Article Form 1	単純な章，節付きのドキュメント．タイトルと目次も付く．
Bibliography	文献用の書式．
Brochure	小冊子用書式．付録も付く．
Guide	ハンドブック用書式．
Letter, LetterHead	手紙用，LetterHead はヘッダー付き．
Manual, Manual Form 1	マニュアル用，タイトル，目次，索引が付く．
MilStd 837 A	軍の標準書式 837 A 用．
ReferenceCard	ポケット・リファレンス・カード用，
Report, Report Form 1	索引なしのマニュアルと同じ．
Slides, Slides Form 1	OHP スライド用．
Text, Text Form 1	デフォールトの単純なテキスト．目次も索引もなし．
Thesis	論文用．

Form 1 はそれぞれの書式の変種

図-5 Scribe の document types

```
@make{text}                    @Comment{document type として 'text' を選択する}
@style{Fontfamily TimesRoman, Spacing 12pt, Spread 0pt, Indent +20pt}
@style{footnotes "@@+[@*] "}
@style{BottomMargin 13cm}
@libraryfile{stable}
```

* Scribe は UNILOGIC 社の登録商標である．

** 著者の手元にすぐ使える Scribe の処理系がなかったので，完璧に図-3 と同じ出力を得るようにはできなかった．Scribe でうまく記述できなかった部分は省略したり代用品で間に合わせたりしてあるが，これは処理系の能力不足というよりは著者が Scribe に不慣れなためにマニュアル[9]だけからは完成できなかったということである．

```
@libraryfile{Mathematics10}
@libraryfile{Specialcharacters}
@Center[@b[Sample Document]]
@define{foo=flushright, rightmargin 0.6in}        @Comment{右詰めのための環境}
@foo[by Yasuki Saito and Hiroshi G. Okuno]
@define{twocolumns=text, Columns 2, LineWidth 192pt, Boxed}
@begin{twocolumns}        @Comment{二段組の開始}
This is a very simple sample document, yet full of commonly used constructs
in everyday typesetting. Some of the exhibited features are listed below:
@begin{Enumerate, numbered <(@1)>}        @Comment{数え上げ環境，括弧付き数字}
narrow paragraph
table
mathematical formulae
@end{Enumerate}
First of all, let's see how to make narrower paragraph:
@modify{Quotation, LeftMargin +20pt, RightMargin +20pt, Spacing 10pt, size 7}
@begin{Quotation}        @Comment{引用の環境を使って狭い段落を実現}
This paragraph is narrower and typeset using smaller font.
Also, this paragraph contains lots of special characters such
as @ovp(a)', @ovp(e)', @ovp(o)^, @ovp(u)'', , , , , , , @y[C] and TeX.
@end{Quotation}
Next is a table of typesetting systems treated in this article:
@standardtable{name smalltable, columns 3, firstcolumn flushleft,
secondcolumn flushleft, third column flushright, boxed}
@begin{smalltable}        @Comment{表の開始}
@tableid{smalltable}
System @\ Author @\ University
RUNOFF @\ J. H. Saltzer @\ MIT
Scribe @\ B. K. Reid @\ CMU
TeX @\ D. E. Knuth @\ Stanford
@end{smalltable}        @Comment{表の終了}
@newcolumn        @Comment{次の段へ行けというコマンド}
And finally comes two mathematical formulae:
@begin{text, indent 0}        @Comment{字下げなしの段落}
The @Math{n}th Fibonacci number @Math{F@down{n}} is given
by@foot{A sample from the "joy of TeX"}:
@end{text}
@begin{MathDisplay}        @Comment{数式モードの開始}
F@down{n}#=#@over[num <(@over[num "1+@sqrt{5}", denom "2"](@up{n}#-#(@over
[num "1-@sqrt{5}", denom "2"])@up{n}>, denom "@sqrt{5}"]
@end{MathDisplay}        @Comment{数式モードの終了}
@begin{text, indent 0}        @Comment{字下げなしの段落}
And this is the an example with integrals@foot{A sample from the "TeX book"}:
@end{text}
@begin{MathDisplay}        @Comment{数式モードの開始}
(@int[from -@infty, to @infty]e@up[-x@up{2}]dx)@up{2} #=# @int[from -@infty,
    to @infty]@int[from -@infty, to @infty] e@up[-(x@up{2}+y@up{2})]dx#dy
#=# @int[from 0, to 2@g(p)]@int[from 0, to @infty] e@up[-r@up{2}]r#dr#d@g(q)
#=# @int[from 0, to 2@g(p)](-@over[num e@up{-r@up{2}},
                    denom 2] @vbar @ss[sub r=0, super r=@infty])#d@g(q)
#=# @g(p)
@end{MathDisplay}        @Comment{数式モードの終了}
@end{twocolumns}        @Comment{二段組の終了}
```

図-6 Scribe のソースファイル

Sample Document

by Yasuki Saito and Hiroshi G. Okuno

This is a very simple sample document, yet full of commonly used constructs in everyday typesetting. Some of the exhibited features are listed below:

(1) narrow paragraph

(2) table

(3) mathematical formulae

First of all, let's see how to make narrower paragraph:

> This paragraph is narrower and typeset using smaller font. Also, this paragraph contains lots of special characters such as à, è, ò, ù, ... , © and TeX.

Next is a table of typesetting systems treated in this article:

System	Author	University
RUNOFF	J.H.Saltzer	MIT
Scribe	B.K.Reid	CMU
TeX	D.E.Knuth	Stanford

And finally comes two mathematical formulae:

The *n*th Fibonacci number F_n is given by*:

$$F_n = \frac{(\frac{1+\sqrt{5}}{2})^n - (\frac{1-\sqrt{5}}{2})^n}{\sqrt{5}}$$

And this is the an example with integrals**:

$$(\int_{-\infty}^{\infty} e^{-x^2} dx)^2 = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-(x^2+y^2)} dx\,dy$$

$$= \int_0^{2\pi}\int_0^{\infty} e^{-r^2} r\,dr\,d\theta$$

$$= \int_0^{2\pi} (-\frac{e^{-r^2}}{2}\Big|_{r=0}^{r=\infty})\,d\theta$$

$$= \pi$$

* A sample from the "joy of TeX"

** A sample from the "TeX book"

図-7 Scribe からの出力
[Postscript で出力を得て，IMAGEN(300 dpi) に出力したもの]

コマンドで使用するフォント (Fontfamily)，行間隔 (Spacing)，段落間の間隔 (Spread)，それに段落の始めの字下げの量 (Indent) を変更している．次の行の "@style" コマンドで，脚注に使う記号を '*'，'**' などに設定している．そのあとに，使用するライブラリのファイル名がいくつか並べてある．

"@Center" はセンタリングをするコマンド，その中に出てくる "@b" はフォントをボールドに切り替えるコマンドである．句切り記号として "{}" が使ってあったり，"[]" が使ってあったりするが，Scribe ではこのほかに "()"，"<>"，"''''"，"""，"''" の中から好きなものを選んで使うことができる．これなどは，Scribe のユーザ・フレンドリな設計の現れの一つである．

Quotation の前の "@modify" コマンドは Scribe のデータベースに書かれているデフォールトのパラメータの値をグローバルに変更するコマンドである．以下で使用する "Quotation" という document type の左右のマージン，行間隔，フォント・サイズなどを変更して，小さい文字で幅の狭い段落を実現している．

このような変更はローカルに行うこともできる．ローカルな変更は，"@begin" と "@end" で囲まれた環境 (environment) の中でのみ有効で，次のようにして指定することができる：

```
@begin [Quotation, Spacing 10pt, size 7]
... Text to be quoted ...
@end [Quotation]
```

表の作成は roff 付属の tbl の方式に似ていて，まず "@standardtable" コマンドで表の書式を指定し，次に表の中味を列記するという形をしている．これは Version 5 から導入されたもので，それまではタブを使ってそれらしく見せることしかできなかった．

数式のシンタックスもあまり説明はいらないだろう．あまり読みやすくはないが，これは roff でも TeX でも同じことがいえる．ただし，roff のプリプロセッサである eqn 同様，数式の出力品質に関しては，はるかに TeX に及ばない．

Scribe の便利さはなんといっても，ユーザに負担をかけずにすべて自動的に決定し，とにかくある程度の品質の文書を作成してしまうという点にある．その

中でも多くのユーザがその便利さを強調するのが文献表の作成機能である．もちろん，現在作成中の論文などとは別に文献情報の入ったファイルがすでに用意されていることが前提になるが，そのファイル中のキーワードだけを文献を引用する箇所に書いてやれば，それでいろいろな書式の文献表を自動的に作成してくれる．

引用をするには，“@Cite（キーワード）”という文字列を引用箇所に書いておけばよい．後はファイルの先頭で使うべき文献情報ファイルを指定し，文献表を出力したい位置で：

@UnNumbered (References)

@Bibliography

と言ってやればよい．文献表の書式は“@Style”コマンドで多くのオプションの中から選ぶことができる．

ここでは詳しくは述べなかったが，“Figure”または“Picture”という環境を使って図形を挿入したり，それにキャンプションを付けたりすることもできる．ただし，挿入する絵自身はたとえば PostScript といった別のシステムで描かなければならない．

以上のように Scribe は与えられた範囲で満足していれば，非常に使いやすく便利なシステムであるが，拡張性に乏しいということが言える．

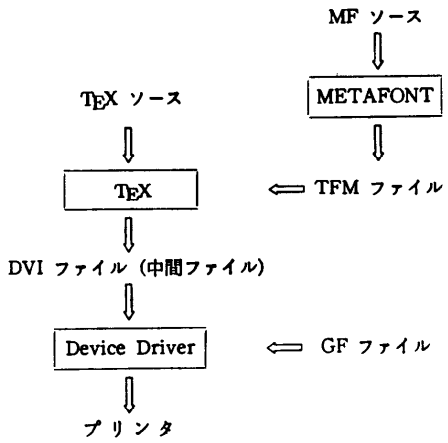

図-8 TEX システムの概要

3.3 TEX

TEX はスタンフォード大学の D. E. Knuth が10年の歳月をかけて開発した文書整形システムである[10),11)]．図-8にシステムの概要を示す．さらに TEX や METAFONT は WEB と呼ばれるシステムで記述されている．

なんといっても TEX システムの最大の特徴は，フ

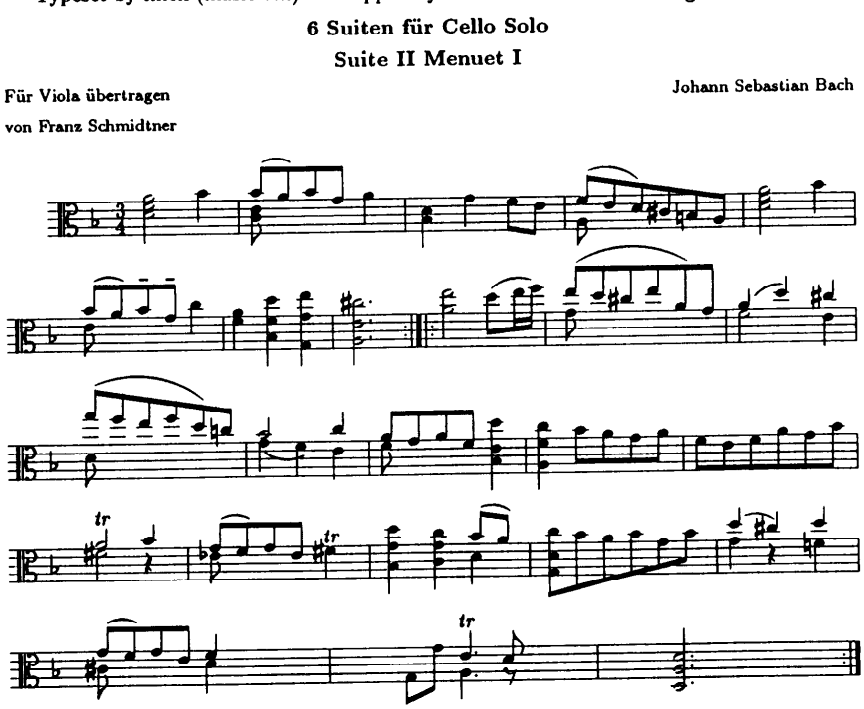

図-9 TEX で出力した楽譜

ォントを作成するためのメタフォント・システム[12),13)]がペアになっている点である．図-8 の右に書いてあるのがメタフォント・システムで，フォントの各文字の記述を含むソースファイルを受け取って，TEX が必要とする TFM ファイル（タイプセットに最低限必要なフォント情報，基本的には各文字の幅，高さ，深さなどのみを含むファイル）と実際の印刷に必要な GF ファイル（各文字のドットパターンを含むファイル，実際にはこれをさらにコンパクトにした PK ファイルがよく使われている．）とを生成する．Knuth はすべてディジタルな手法で作られたある本のゲラ刷りの品質が通常の印刷となんら変わらないものであることを知ったときに，「印刷はすなわちビット操作である」ということを悟り，TEX プロジェクトを始めたという[14)]．つまり，Knuth はページの上のあらゆる点をインクで黒くするかどうかを一つ一つの文字の細部に至るまでコントロールしようと目論み，'TEX と METAFONT という二つのプログラムの組合せで，それを達成してしまったのである．

フォントを自分で作れるというのは roff にも Scribe にも見いだされない新しい次元の自由度である．もちろん Knuth 自身を除けば，本を書くたびに新しいフォントを作ってしまうなどということは普通の人はやりそうにもないが，その可能性が開かれていて，かつ細かなカスタマイズもできるというのは大きな違いである．TEX には Knuth のデザインした Computer Modern というフォント[15)]がついてくるが，そのほかに簡単なものでは会社のロゴから始まって，数学用の特殊なフォント，発音記号のフォント，チェス用のフォント，楽譜用のフォントなど実にさまざまなフォントが作られ利用されている．こうしてフォントを拡張した TEX でタイプセットされた楽譜のサンプルを図-9 に示しておく．TEX の可能性の大きさをまざまざと見せつける例である．

TEX の特徴として多くの人があげるのが「数式のタイプセットに優れている」という点である．これは図-10 を入力として得られた図-11 を見てもらえばはっきりする．もともと数式を美しく出力することは

```
\newdimen\fullhsize \newdimen\fullvsize
\fullhsize=410pt \fullvsize=380pt \hsize=192pt \baselineskip=12pt
\def\fullline{\hbox to\fullhsize}
\def\fullcenterline#1{\hbox to\fullhsize{\hss#1\hss}}
\def\fullrightline#1{\hbox to\fullhsize{\hss#1}}
\newdimen\dimentemp \newbox\partialpage %ここから二段組のマクロ定義開始
\def\begindoublecolumns{\begingroup %二段組開始コマンド
\output={\global\setbox\partialpage=\vbox{\unvbox255\bigskip}}\eject
\output={\doublecolumnout} \hsize=192pt \vsize=910pt}
\def\enddoublecolumns{\output={\balancecolumns}\eject %二段組終了コマンド
\endgroup \pagegoal=\vsize}
%二段組用の出力ルーチン
\def\doublecolumnout{\splittopskip=\topskip \splitmaxdepth=\maxdepth
\dimentemp=\fullvsize \advance\dimentemp by-\ht\partialpage
\setbox0=\vsplit255 to\dimentemp \setbox2=\vsplit255 to\dimentemp
\onepageout\pagesofar \unvbox255 \penalty\outputpenalty}
\def\onepageout#1{\shipout\vbox{\makeheadline %実際の出力を行うマクロ
\vbox to\fullvsize{\boxmaxdepth=\maxdepth #1{makefootline}
\advancepageno}
\def\makefootline{\baselineskip=24pt \fullline{\the\footline}}
\footline={\hss\tenrm\folio\hss}
\def\pagefofar{\unvbox\partialpage %ページボディを組み立てている部分
\wd0=\hsize \wd2=\hsize \hbox to\fullhsize{\box0\hfil\box2}
\ifvoid\footins\else
\vskip\skip\footins \footnoterule \unvbox\footins\fi}
%最後のページの二段の高さをそろえるための出力ルーチン
\def\balancecolumns{\setbox0=\vbox{\unvbox255} \dimentemp=\ht0
\advance\dimentemp by\topskip \advance\dimentemp by-\baselineskip
\divide\dimentemp by2 \splittopskip=\topskip
{\vbadness=10000 \loop \global\setbox3=\copy0
\global\setbox1=\vsplit3 to\dimentemp
\ifdim\ht3>\dimentemp global\advance\dimentemp by1pt \repeat}
\setbox0=\vbox to\dimentemp{\unvbox1} \setbox2=\vbox to\dimentemp{\unvbox3}
\pagesofar}
```

```
\font\titlefont=cmbx12
\overfullrule=0pt %行がはみ出たときに表示される overfullbox を見えなくする
\leavevmode \vskip5mm
\fullcenterline{\titlefont Sample Document} \bigskip %タイトルのセンタリング
\fullrightline{by Yasuki Saito} \bigskip %著者名の右詰め
% ここから本文が始まる
\begindoublecolumns %二段組開始
This is a very simple sample document, yet full of commonly used constructs
in everyday typesetting. Some of the exhibited features are listed below:
\smallskip
\itemitem{(1)}narrow paragraph %数え上げ
\itemitem{(2)}table
\itemitem{(3)}mathematical formulae
\smallskip
\font\smallfont=cmr7
First of all, let's see how to make narrower paragraph:
{\narrower\smallskip\noindent\smallfont\baselineskip=10pt %行間隔も狭くしている
This paragraph is narrower and typeset using smaller font.
Also, this paragraph cbntains lots of special characters such
as \'a, \'e, \^o, \"u, \ae, \AA, \c c, \dag, \S, \P, \copyright\ and \TeX.
More can be found in the mathematical examples below. \smallskip}
Next is a table of typesetting systems treated in this article:
$$ %表を中央に置いて周囲にスペースをあけるために数式表示モードにしている
\vbox{\tabskip=0pt \offinterlineskip
\def\tsumemono{&\omit&height4pt&\omit&&\omit&\cr}
\def\tablerule{\tsumemono\noalign{\hrule}\tsumemono}
\halign{\vrule#\tabskip=0pt plus2em& %以下がひな型を記述した preamble
$\>$#\hfil$\>$&& \vrule\vrule\vrule#&
$\>$#\hfil$\>$&& \vrule#& $\>$#\hfil$\>$&& \vrule#
\tabskip=0pt\cr
\noalign{\hrule}\tsumemono %一番上の横罫線
&\hfil System &&\hfil Author &&\hfil University &\cr
\tsumemono\noalign{\hrule}\noalign{\hrule}\tsumemono
&RUNOFF&&J. ~H. ~Saltzer &&MIT &\cr\tablerule
&Scribe&&B. ~K. ~Reid &&CMU &\cr\tablerule
&\TeX &&D. ~E. ~Knuth &&Stanford &\cr
\tsumemono\noalign{\hrule}}} %最後の横罫線
$$
And finally comes two mathematical formulae:
\noindent %字下げ省略コマンド
The $n$th Fibonacci number $F_n$ is given by\footnote{$^*$}{A sample from the
"joy of\TeX"}:
$$ %数式表示モード開始
\eqalignno{F_n&={{\displaystyle
\left({1+\sqrt5}\over2\right)^n-\left({1-\sqrt5}\over2\right)^n}
\over{\displaystyle\sqrt5}}&(1)\cr}
$$ %数式表示モード終了
\noindent
And this is an example with integrals\footnote{$^{**}$}{A sample from the
"\TeX book"}:
$$ %数式表示モード開始
\eqalignno{\biggl(\int_{-\infty}^\infty e^{-x^2}\, dx\biggr)^2
&=\int_{-\infty}^\infty\int_{-infty}^\infty e^{-(x^2+y^2)}\, dx\, dy\cr
&=\int_0^{2\pi}\int_0^\infty e^{-r^2}r\, dr\, d\theta\cr
&=int_0^{2\pi}\biggl(-{e^{-r^2}\over2} \bigg\vert_{r=0}^{r=\infty}\, \biggr)\, d\theta\cr
&=\pi. &(2)\cr}
$$ %数式表示モード終了
\enddoublecolumns %二段組終了
\vfill\eject\end
```

図-10 TEX のソースファイル

Sample Document

by Yasuki Saito

This is a very simple sample document, yet full of commonly used constructs in everyday typesetting. Some of the exhibited features are listed below:

(1) narrow paragraph
(2) table
(3) mathematical formulae

First of all, let's see how to make narrower paragraph:

> This paragraph is narrower and typeset using smaller font. Also, this paragraph contains lots of special characters such as à, é, ô, ü, æ, Å, ç, †, §, ¶, © and TEX. More can be found in the mathematical examples below.

Next is a table of typesetting systems treated in this article:

System	Author	University
RUNOFF	J. H. Saltzer	MIT
Scribe	B. K. Reid	CMU
TEX	D. E. Knuth	Stanford

And finally comes two mathematical formulae:

The nth Fibonacci number F_n is given by*:

$$F_n = \frac{\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n}{\sqrt{5}} \quad (1)$$

And this is an example with integrals**:

$$\begin{aligned}\left(\int_{-\infty}^{\infty} e^{-x^2}\,dx\right)^2 &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-(x^2+y^2)}\,dx\,dy \\ &= \int_0^{2\pi}\int_0^{\infty} e^{-r^2} r\,dr\,d\theta \\ &= \int_0^{2\pi}\left(-\frac{e^{-r^2}}{2}\Big|_{r=0}^{r=\infty}\right)d\theta \\ &= \pi. \end{aligned} \quad (2)$$

* A sample from the "joy of TEX"
** A sample from the "TEXbook"

図-11 TEX による出力例
[IMAGEN(300 dpi) に出力したもの]

TEX の目標の一つであったから，これは当然のことかもしれないが，TEX では数式の整形に関してかなり細かい配慮が行き届いている．

もう一つの特徴として，マクロ機能の強力さをあげることができるだろう．通常使われている TEX は裸の TEX に plain と呼ばれるマクロ・パッケージを読み込んだもので，ユーザが使うコマンドはこのレベルのマクロで定義されているものが多い．Scribe とほとんど同じ機能を実現している LATEX もすべてがマクロで記述されている．著者自身もマクロの機能だけで，日本語 TEX（jTEX）のプロトタイプを作った経験がある[16]．

TEX は文書整形という仕事に対して，明確なモデルを提供しているという点でも roff や Scribe とは一線を画している．それが，"box and glue" モデルである．TEX の世界では箱 (box) が一番基本的な要素である．箱は幅と高さと深さをもったものとして内部的には表現される（図-12 を見よ）．一つ一つの文字あるいは線分が最も単純な箱で，それらが組み合わさってできる行，段落などページ上に配置すべきものはすべてが箱なのである．箱の中におく各文字のドットパターンは METAFONT で定義するので，文書整形とは一応独立である．この箱のサイズに関する情報が上で述べた TFM ファイルに記述されているもので，この情報さえあれば，TEX は文書整形の仕事を遂行することができる．

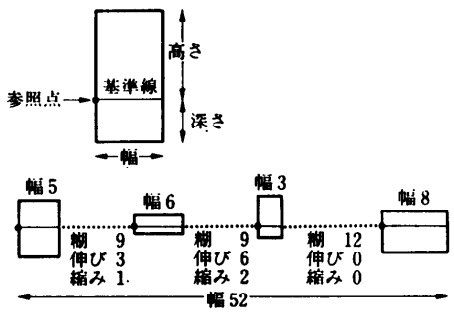

図-12 箱と糊 (box and glue)

箱はすき間なくぎっちりと並べられて単語に対応する箱になり，単語と単語の間の空白をあけるための糊 (glue) が挿入されて，行となっていく．この糊は指定

された範囲で伸びたり縮んだりできるので（図-12 参照），行の右揃えを行うにはこれをうまく調整してやればよい．

実は TEX はもっと凝ったことをしていて，一つのパラグラフを読み込んだあとで，そのパラグラフ全体がバランス良く行に分割されるように糊（glue）の伸び縮みを設定していく．これがいわゆる DP Matching による行分割アルゴリズムである[17]．

こうして作られたパラグラフは行という箱（正確には hbox という）が垂直方向に並べられたもので，これまた別の箱になる（正確には vbox）．こうして，次次に箱の中に箱を並べ，積み上げていくことでページの全体が形作られていく．

この "box and glue" モデルは無限大の長さの（というより無限に伸び縮みできる）glue を導入することによって，中央揃え右揃えなどを統一的に扱えるばかりか，作表，数式のタイプセットもまったく同じ枠組みの中で取り扱えるという意味で，かなり一般的なモデルである．

一般論はこれぐらいにして，図-10 のソースファイルをもう少し詳しく見てみよう．roff や Scribe の例と見比べて一番大きな違いは，ここでは2段組にするためのマクロを陽に書き下していることである．先頭の 40 行ほどが，このマクロであるが，標準の TEX には2段組の機能は含まれていない（LATEX にはこの機能がある．）ので，これを書かないと2段組ができない．といっても，このコードは TEX のマニュアルである TEX book[10] の付録から借用してきたもので，もともとは TEX book を書くために Knuth が用意したマクロである．TEX を使っているときには，このようにして人の書いたマクロを借用することが多い．

TEX では "\" で始まる文字列（何文字でもよい）でコマンド，変数，マクロなどを表す．普通の文字は並べた順番にタイプセットされ，その間にタイプセットの仕方を指示するコマンド，マクロなどが散りばめられる．連続する改行二つ以上で段落の切れ目を表す．最初に出てくる "\fullhsize"，"\fullvsize" などはこの例題用のページの幅や高さを保持するために設定した変数である．

TEX のマクロ定義は一般に

\def\macroname<argument pattern>{<definition>

という形をしていて，"<argument pattern>" のところには定義の中で使うことのできる '#1', '#2', …, '#9' までの9つのパラメータを含む文字列パターンを書く．例えば，"\fullcenterline" の定義は引き数として '#1' 一個を取り，それを "\fullhsize" の中でセンタリングする．ここで使われている "\hss" が上で述べた無限大に伸び縮みできる糊（glue）である．この例で分かるようにパラメータはマクロの定義本体の中に文字列として埋め込まれる．

この2段組マクロは単にテキストを2段に組むだけではなく，最初のページではページの途中から2段組を開始し，最後のページでは2段の高さを揃えるということまでしてくれる．概要だけを説明しておくと，まず2段分に十分なだけの一段の幅の縦に長いページを作っておいて，その先頭からページの高さに相当する部分を二つ切り出してきて 'box0' と 'box2' に入れる（これが，"\doublecolumnout" の中の二つの "\vsplit" コマンドによって実現されている）．それを横に二つ並べ，あいだに無限に伸びる glue ('\hfil') をはさんで一つのページを作る（これは，"\pagesofar" がやっている）．あとは "\onepageout" でそれにヘッドラインとフットラインを付けて出力する．2段組の最後にきて "\enddoublecolumns" が呼ばれると，出力のためのルーチンとして，今までの "\doublecolumnout" ではなく，"\balancecolumns" が呼ばれることになる．このマクロは最後に残っているテキストをだいたい半分に切ってから，半分ずつのおのおのがほぼ同じ長さになるまで，切断する高さ ('\dimentemp') を1ポイントずつ増やしていって最終的な値を決定している．

このマクロ定義の次から本文に対応するコードが始まるが，そこにはそれほどむずかしい部分は出てこない．まず最初に箇条書きをする "\itemitem" というコマンドが使われている．これは段落の字下げの量の2倍の位置から各項目の中味が始まり，番号はその左にはみでる書式である．また "\item" というと，段落の字下げの位置と同じ所から項目が始まり，番号はその左に置かれる．

"\narrower" コマンドはパラグラフの左右に段落の字下げと同じ量の空白を取って，通常の段落より狭い段落を作る．ここでは，この段落をより小さなフォントでタイプセットするために直前で宣言した '\smallfont' というフォントを選択している．この段落に現れる特殊文字は TEX のもっている特殊文字のごく一部である．また，"\TeX" というのは次のように定義されたマクロである．このマクロでは，カー

ニングと呼ばれる文字の微妙なスペースの調整機能と“\lower”という上下方向への移動コマンドが利用されている．ex，em などは長さの単位である：

```
\def\TeX{T\kern-.1667em
        \lower.7ex\hbox{E}\kern-.125emX}
```

次は表である．表の作成は TEX でもユーザにとって難しいものの一つといわれているが，基本的には roff の tbl などとそう違ったものではない．作表には \halign というコマンドを使うのだが，これは一般には

```
\halign <box width spec> { <preamble>\cr
<first row>\cr
<second row>\cr
...
<last row>\cr}
```

というシンタックスをしている．そして，<preamble> には template と呼ばれる‘#’を必ず一つ含む各欄のひな型が‘&’で区切られて並び，各行にはそのひな型の‘#’の位置に埋め込まれるべき各欄のテキストが，これもまた‘&’で区切られて並んでいる．

<box width spec> は表全体の横幅を規定したいときに“to 5cm”のように使う．

TEX は各欄のテキストをひな型 (template) の中に埋め込んで各行を作り，それを垂直方向に重ねて表にする．このときに各欄の幅などは自動的に最大幅の欄に合わせてくれる．もう一つ重要なことは罫線も一つの行，一つの欄として取り扱われるということである．線分も一つの箱 (box) だったのだからこれは当然のことである．

ここまでのことが分かると図-10 の表を作るためのコードがある程度理解できると思う．縦の罫線がくるべき位置には“\vrule#”というひな型が使ってあり，表の各欄の要素に対応するひな型としては“$\>$#\hfil$\>$”が使われている．これは欄の両側に若干のスペース（‘$\>$’）を取った上でテキストを左詰めにするというものである．行と行のあいだの罫線は“\hrule”によって引かれ，その罫線の上下に 4 ポイントの高さのすき間をあけるために，“\tsumemono”というマクロが定義され使われている．

ここで使われている，“\halign”のほかに，タブの設定を基本にした“\settabs”というコマンドを使って作表をする方法も提供されているが“\halign”ほど強力ではない．

数式の部分はあまり説明を要しないだろう．“\eqalignno”というコマンドは数式の最後に番号を付けるとともに，何行かにまたがる数式のたとえば等号の位置を上下にそろえるためのものである．

TEX は Scribe とほぼ同時期に開発されたシステムであるが，その後何回かの改訂を経て現在ではバージョン 3.0 となり，かなり安定したシステムになっている．当初 Knuth は自分のためだけに使うことを考えていたようだが，すぐにユーザが増え，ユーザ・グループ（TEX Users Group, TUG という）もできて，ユーザのいろいろな要求を取り込んでいった．その際に，内部で固定されていたパラメータなどをユーザが変更できるようにするということが何回となく繰り返された[14]．そのおかげで，TEX はユーザによるタイプセットの制御が細かいところまでできるシステムになっている．もちろん，それが容易にできるかどうかは別の問題で，TEX を駆使して自分の思いどおりの出力を得るためには相当の知識を必要とするが，逆にマクロを十分に使いこなせばたいていのことが可能になるといったところがある．

上記の TUG の機関誌である TUGboat には，そのようにして拡張，改良された TEX の話が山のように出てくる．まず各国語への拡張がいくつもある．著者の知るかぎりでも，ドイツ語，フランス語，ロシア語を始めとして，現代ギリシャ語，トルコ語，サンスクリット，タミール語，ヘブライ語，アラビア語そして日本語への拡張が報告されている．アラビア語は当然，右から左へも書けるようになっている．これらの言語への拡張は単にフォントを用意するだけではなく，ハイフネーションのパターンを変えたり，その言語特有の記法を導入したりとかなり工夫が必要だが，TEX の拡張機能（マクロ）はそれを可能にしている．また 4. で触れる図形を書くための PICTEX とか，PASCAL の構文記述に使われた syntax diagram を書くマクロとか，チェスボードやクロスワードパズルをタイプセットするといったものまである．そしてこれらのほとんどが，TEX のマクロ機能によって実現されているのである．

TEX はもともと論文や，科学技術関係の本をタイプセットすることを目的に作られたものであるが，最近は商業用にも使われだしている．2000 万部の発行部数を有するアメリカのテレビ番組情報誌である TV Guide ではその紙面の一部を実験的ではあるが TEX を使ってタイプセットすることを始めている．

3.4 その他

これまで，典型的な文書整形システムを概観してきたが，世の中には他にも多くの文書整形システムが存在する．特に最近はビットマップのディスプレイを利用したものが多い．ここでは，それらのうちのいくつかにごく簡単に触れることにする．

3.4.1 VORTEX

カリフォルニア大学バークレー校で3，4年前から進められているプロジェクトに VORTEX プロジェクト[18]がある．これは数式や図を含む高度な技術的ドキュメントを作成するための環境を作ろうというもので，TEX をベースにしている．このシステムの特徴は，TEX に与えるソースファイルだけでなく，その出力も画面を使って自由に操作しようという点にある．つまり，出力のほうをユーザが画面上で変更すると TEX のソースの対応する部分が自動的に変わってくれるようにしようというわけである．インプリメントには VLisp（VORTEX lisp）と呼ばれる Lisp を使い，出力の表現には Postscript を使っている．最近のリポート[18]によると，ほぼシステムは完成しているようであるが，出力からソースへの逆変換がどれくらいうまくいっているのかは定かではない．それよりも，部分的に TEX にかけて変更のあった部分だけを効率よく処理できる INCTEX（Incremental TEX）というのができていて，それがかなり便利であると報告されている．

3.4.2 Andrew システムの EditText

カーネギー・メロン大学の ITC（Information Technology Center）プロジェクトが開発した，大学全体をカバーするネットワークとワークステーションからなる計算機環境が Andrew システム[19]である．このシステム上で，ドキュメント作成やプログラム開発用に提供されているエディタが EditText である．EditText はビットマップのディスプレイを利用するエディタであるが，ベースには Scribe や troff が控えている．ユーザは画面上でほぼ目に見える形でメニューからコマンドを選びながらテキストを編集していく．画面上に Scribe や troff のコマンドは一切見えないが，実はユーザに見えないようにマスキングされているだけで，ユーザの指示は Scribe や troff のコマンドとして編集しているファイルに挿入される．ユーザがこれらの文書整形用のコマンドを見たい場合には "expose style" というオプションを選べば見ることができるが，そんなことはほとんどせずに，テキストを作成できるのが，この EditText の特徴である．さらに troff の場合にはプレビューアがあって，画面上で出力イメージを確かめることもできる．

これはエディタと文書整形システムが融合したシステムの例だが，このようなシステムはかなり多い．ただし，完全に WYSIWYG 方式にはなっていない点に注意する必要がある．

3.4.3 SGML

SGML というのは "Standard Generalized Markup Language" の略で，ドキュメントの論理的な構造を記述するための記法（notation）である[20]．いままで取り上げてきた文書整形システムとは毛色の変わったものであり，処理系もまだあまり存在しないが，Scribe でみたようにドキュメントの論理的な構造を重視するという方向の延長線上にある話なので，その概要を少し述べる．

そもそも SGML は文書整形システムでもページ記述言語でもなく，しいて言えばドキュメント記述言語である．Markup というのは通常ベタに書かれているテキストにその論理的構造を明確にするマークを付けようという発想からきている．つまり，「これは章のタイトルである」とか，「これは節である」とか，「これは本の名前の参照である」といったテキストの構造に関する情報を陽に記述しようというのである．そのご利益は，こうして付加した情報がいろいろの目的に有効に使えるという点にある．当然，文書整形にも使えるし，索引や文献を作成するときにも使える．また，Markup されたテキストをデータベースにしておき，検索プログラムがこの情報をうまく利用することも考えられる．このようにして，テキストの論理的な構造をかなり一般的に表して，テキストの流通や利用を促進しようというのが SGML のねらいである．

SGML には "markup minimization" という面白い考え方があって，正しく作られたテキストであれば当然満たすべき制約を利用して，必要なマークを最小限にとどめる工夫がなされている．したがって，ユーザはうまく document type を定義してやれば，ほとんど markup なしに，ドキュメントの論理構造を正確に表現することが可能となる．これは，木村泉が文書整形言語の解説[21]の中で「無指令型の整形言語」と言っている話と相通ずるものである．

実際ヒューレット・パッカード社では MARKUP と呼ばれる SGML のパーザを作り，文書整形システム（TEX）へのフロントエンドとして利用する試みがなされている[20]．

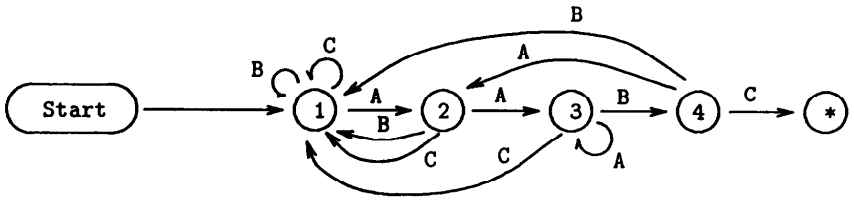

図-13 EEPIC のサンプル出力

4. 図の取り扱い

文書，特に技術的な文書においては，図やグラフは必要不可欠である．図を扱うにはいろいろの方式があるが，そのいくつかを TEX の場合を例に取って考えてみよう．

まず，文書整形システム側では，必要なスペース，枠をあけておいて，あとから図を糊付けするという最も原始的な方法がある．いわゆる「切り貼り」である．しかし，これは計算機屋にはあまり評判が良くない．

次に考えられるのは，絵や図を描くプログラムで生成したファイルをなんらかの形で文書整形システムの出力と合体してプリンタに出力するというものである．たとえば，PostScript で絵を描いておいて，TEX の出力を PostScript プリンタに出すときに，プリンタドライバが絵のファイルを読み込んで，しかるべき位置に挿入するわけである．このとき，どこにどういう大きさの絵を挿入するかを TEX のソースファイルのほうで指定しなければならないが，それには TEX の "\special" というコマンドを使う．たとえば，

```
\special{psfile: picture-file-name}
```

といったぐあいである．"\special" は与えられた文字列をそのままプリンタドライバに渡すので，ドライバの側で，これを正しく解釈して，必要なファイルの読み込みと出力への埋め込みをしなければならない．

Scribe にも同様の機能があって，"@picture" というコマンドでファイルを指定してそれをドキュメントの中に含めることができる．

TEX のマクロの機能を使って，マクロで図形の記述を可能にしてしまうという方法もある．PICTEX[22] がそれである．PICTEX は全体で 3700 行ほどのマクロパッケージだが，特殊なフォントは一切使わない．通常の TEX のフォントのピリオドを並べることによって，全ての図形を書いてしまう．線分，長方形，円弧などの図形や，座標軸，それへの目盛りなどの基本コマンドがそろっていて，これを組み合わせることによってある程度の複雑さの図形を書くことができる．しかし，一番の欠点は効率が悪いことと，フリーハンドに近いような絵がうまく描けないという点にある．

LATEX にはもともと 'Picture Environment' というのが用意されていて（もちろん TEX のマクロで書かれている．）箱や矢印や円弧などを含む簡単な線図形を書くことができるようになっている．これは，いろいろな傾きの線分，いくつかの円弧成分などの図形用のフォントを利用しているので，書ける図形に制限があり，出力もあまり美しくない．

この制限（任意の傾きの線分が書けないなど）を同じマクロのレベルで取り除き，記述法は LATEX の 'Picture Environment' のものを踏襲し，実際の図形の出力には '\special' を使って，プリンタ側の図形描画機能をフルに使うようにしたものが EEPIC である．EEPIC で描いた図形の例を図-13 に示す．

さらに図形をコマンドの列として記述するのはまどろっこしいという人のために，各種のお絵描きプログラムで作成した図形を EEPIC のコマンド列に変換するプログラムなどもできているので，これを利用すれば，フリーハンドの絵なども TEX の中に取り込むことができる．

プロッタ出力を各種の端末へ出すプログラム（gnuplot）を拡張して，LATEX の 'Picture Environment' 用の出力を出すように改良したというものもある（Gnu TEX）．これはいわゆるプリプロセッサ方式で，上で述べたお絵描きプログラムの出力を取り込むのと同じ方式である．

TEX の周辺で図形を扱うために行われているいろいろな試みを概観したが，いくつかの問題点を指摘することができると思う．

1) 図形自身をどのようにして生成するかに関してはいくつかの方式があり，どれも優劣付けがたいのが

現状であり，理想的な図形生成法は，それ自体今後の重要な研究課題である．

2) 文書整形システムとしては，図形をうまく文書の中に配置するという問題も重要である．しかし，roff, Scribe は言うに及ばず，TEX においてもこの図形の配置がうまく扱えているとは思えない．特にその図への文書中からの参照点と図をなるべく近くに配置するとか，見開きなら次のページでもいいというような制約を満たすように自動的に配置することはなかなか難しい．

3) 図形のサイズにはある程度の任意性があり，人間がタイプセットする場合にはテキストの量をにらみながら，適当に図形を拡大縮小して，うまく目的のページに埋め込んでしまうという手法をよく使う．これが自動的にできるようになると大変ありがたい．現状でもサイズを陽に指定すれば，そのサイズに拡大縮小して図を埋め込むということは可能なので，プレビューなどをして，人間が試行錯誤的に指定サイズを決めればいいのだが，これはなかなかやっかいな作業である．

5. コマンド埋め込み方式対 WYSIWYG 方式

3. で概観したシステムのうち，roff, Scribe, TEX などはすべてコマンドがタイプセットしたい文書の中に混在しているという意味でいわゆるマークアップ方式である．それに対して **3.4** で取り上げたシステムのうちのいくつかは画面上に見えるものが，そのまま出力イメージになっているという意味で WYSIWYG 方式と呼ばれる．Andrew システムなどは完全には WYSIWYG ではないがそれに近い方式である．

WYSIWYG という考え方は画面エディタの出現と時を同じくしていわれだしたもので，今では，マン・マシンインタフェース一般に対して使われる概念である．従来ほとんど目に見えない形で進んでいた処理をできるだけ目に見えるようにしようという大きな流れの一つである．文書整形システムの目的は最終的に目に触れる紙の上の文書を作ることにあるのだから，WYSIWYG 的な要素（すなわち目に見えるということ）は必要不可欠である．たとえば，マークアップ方式のシステムでも必ずと言っていいほど，プレビューアのようなユーティリティを利用する．したがって，ここでは文書の配置に関する指示をコマンドの形でファイル中に埋め込むのと，画面を見ながらマウスなどを使って，インタラクティブに指定するのとどちらが良いのかを考えてみよう．

そのように考えると，目的に応じて使い分けるのが一番いいというのが正しい答えのようである．たとえば，図形の配置などは，現時点ではインタラクティブに決めるのが妥当であるが，行を分割したり，ページを決定したりするのまで，ユーザがマウスで場所を指定するなどということは考えられない．これでは，文書整形を計算機ではなく，人間がやっていることになってしまう．

つまり，自動化できるところはできるだけ自動化し，人間の介在を必要とする微調整のときに初めて WYSIWYG 的なマン・マシンインタフェースを利用するのがいいと思われる．

ただし，マークアップ方式で埋め込むコマンド自身がどのようなレベルのコマンドであるかも重要である．roff や TEX はかなり細かいところまでいじれるのでハッカー的なユーザには評判がいいが，一般ユーザにとって使いやすいシステムになっているかというとなかなかそうはなっていない．つまり一般ユーザ向けにはマークアップ言語はテキストの論理的な構造を表現するようなものであるべきで，その意味では Scribe や LATEX，あるいは SGML のような方向が正しい方向であると思われる．

6. 日本語化

最初から日本語用に作られた文書整形システムというのもいくつかあるが，外国で開発されたシステムを拡張して日本語も扱えるようにしたものが圧倒的に多い．外国のほうが，開発が進んでいたこと，日本語にはたいてい欧米語もあちこちに埋め込まれていることなどの原因が考えられる．

ここではいくつかの例を簡単に紹介し，若干の考察を加える．

6.1 JTROFF

上で紹介した troff を日本語用に改良したのが JTROFF[23] である．日本語の一文字一文字を欧文の語 (word) と同等に扱って，和欧混合文書の整形を実現している．基本的にはプリプロセッサによって日本語の文字と文字の間にごくわずかのスペースを入れて欧文と同じアルゴリズムを適用し，行の追込み，右揃えなどを行う．JTROFF の特徴は日本語用のフォントとして，文字幅が一定ではなく文字ごとに文字幅の異なるフォントを採用していることである．これによ

って，可読性が増し，さらに情報の密度が上がると設計者らは主張している．

troff は上でも述べたとおりかなりプリミティブな言語であったから，JTROFF における拡張も，フォントの選択と各種のスペースを指定するコマンドに留まっているが，単純な整形であればこれで十分である．

6.2 KJANUS

JANUS は欧米テキストおよびイメージを含む複合文書を対話的に編集・整形するシステムである．IBM 370 上で稼働している．「タグ」と呼ばれる印を文書に付けることによって各種の書式に整形してくれる．またユーザが新しいタグを定義することもできる．この拡張機能を利用して日本語の文字も扱えるようにしたのが，KJANUS である[24]．

その実現には，下で説明するマクロ・バージョンの jTEX と同じ手法を使っている．

6.3 TEX の日本語化

歴史的には，日本で一番最初に TEX を日本語化したのは三菱電機の藤田氏らであるが[25]，これは古いバージョンの TEX をベースにしていたことや，プリンタドライバを実現するためにハードウェアにまで手を加えたことなどもあってあまり広まらなかった．その後 TEX で日本語を扱えるようにしたものには，著者自身が開発した jTEX[26] とアスキー(株)の日本語 TEX[27] とがある．どちらもほぼ同じような機能を実現しているので，ここでは jTEX を例にして説明を進めることにする．

最初に実現した jTEX は TEX の本体には何も手を加えずに，マクロの機能だけを利用して，日本語を含むファイルのタイプセットを可能にしたものである[16]．jTEX の特徴は日本語のフォントを通常の TEX が扱える一フォント256文字のサイズのサブフォントに分けて，日本語の文字数の多さに対処していることである．日本語が基本的にはどの文字間でも改行できる（禁則処理は除く）こと，および行への追込み，右揃えなどは，日本語の各文字間にごくわずかの糊(glue)を挿入することで実現している．あとは，TEX の行分割アルゴリズムがそのまま適用される．また，禁則処理はこの glue を省略することによって実現している．

このマクロ・バージョンの効率の悪さを克服するために，TEX 本体に手を加え，上と同じことを TEX の中で実行してしまうようにしたのが，現在の jTEX である[26]．

アスキーの日本語 TEX では，フォントをサブフォントに分けないこと，およびペナルティを挿入することによって禁則処理を実現していることなどが，jTEX との主な違いである．jTEX で追加した行数は約3000行であるが，アスキー日本語 TEX の場合には改良を加えた行数はもっと多い．

最近は各種日本語 TEX の出力から写植機出力を得るサービスなども開始されて，日本語 TEX の利用が本格化してきている．

6.4 考 察

日本語のタイプセットはどちらかというと単純なので，英文用の文書整形システムを流用することによって，ほぼ満足のいく結果が得られている．ただし，縦書きとかルビをふるとか，まだいくつも問題が残っている．

特に，日本語フォントの整備という問題は重要である．フォントは文書整形システムそのものの問題ではないという考え方もあるが，TEX のように徹底的に出力の品質を向上させようとすれば，フォントのよしあしは一番大きな要因である．最終的に写植機のレベルまでいけば，各種の日本語フォントがサポートされている場合が多いが，上で概観したような日本語文書整形システムでは，限られたフォントをやりくりして使っているのが現状である．たとえば，jTEX では最初は JIS のプリンタ用 24 ドットフォントというものをベースにいくつかのサイズのフォントを拡大して作って使用していたし，現在でも大日本印刷から提供される各種サイズのドット・フォント（明朝とゴチック）を使用している．

日本語特有のタイプセット法というのは，縦組み以外にはあまり存在しないが，日本語では英語などとの混植が多いことがその特徴の一つであろう．混植をした場合に出来上りが美しく見えるためにはフォント自身の形以外に，日本語と英語の間にどれくらいのスペースをあけるかとか，バランス良く見えるためにはそれぞれのサイズがどれくらいがいいかとか，ベースラインをどこに合わせるかなど整形上の問題点も数多く存在する．印刷業界においてはこのようなノウハウは長年にわたって蓄積されていると思われるが，文書整形システムには必ずしも組み込まれているとはいえないのが現状である．

7. 将来の文書整形システム

現在広く使われている文書整形システムのいくつかを具体的に解説し，さらに図形の取り扱い，日本語化などの問題を考察してきたが，それでは将来の文書整形システムはどのように発達していくのであろうか？

現在でもすでにある程度進みつつあり，近い将来に必ず標準的になるであろうこととして，マン・マシンインタフェースの改良，特にビットマップ・ディスプレイの多用があげられる．最終出力が目的なのだから，それをプレビューするというのもごく自然なことだし，ユーザの要求を一番表現しやすいのは最終出力イメージ上であるから，最終出力イメージを直接使うインタフェースが開発されるのは目に見えている．roff のように暗号か判じ物のようなコマンドや変数名は言語道断だが，Scribe や TEX でもエラー・メッセージなどがユーザに分かりやすいわけではない．この辺の改良も確実に行われるだろう．

さらに，一般ユーザからみると TEX のように細かなところまで制御できるというよりは，もっと高いレベルで整形に関する指示を与えたいという要求が必ず出てくるはずである．たとえば図の配置などに関して，「この文と同じページあるいは見開きの隣のページに配置せよ」などという要求が容易に記述できるようになることが望まれる．

一応 Scribe のような方式では，ユーザは文書の論理的な構造だけを指定して，整形の細かな点はシステムにまかせるという意味で roff や TEX などよりは高度な要求記述を実現していると言うこともできる．ただし，システムが用意した書式では満足しないユーザが必ず出てくるという問題が厳然として存在する．それには，機能拡張やカスタマイズの手段を提供することで対処するのだが，既存のシステムをみるかぎり，機能拡張やカスタマイズはかなり手間のかかるプロセスである．roff のマクロも，TEX のマクロもその内部処理法をかなり理解していないと，ちょっと凝ったことをやろうと思った途端にどうやったらいいのか分からないという状況に陥ることが多い．

これを，どうにかして容易にカスタマイズできるようにするというのが，大きな課題であり，かつ将来の有望な方向であると思われる．

もちろん何の指示をしなくても理想的な形に整形してくれるというのが究極の文書整形システムであり，木村泉の言う「無指令型の整形言語」[21] というアイデアはなかなか魅力的なものではあるが，それの実現はかなり先になるだろう．

カスタマイズを容易にするというのは困難な課題であるが，可能な方法として以下のようなものが考えられる：

（1） ユーザが思いつきそうな変更を前もってなるべくたくさん列挙し，それらをうまくカバーするように拡張用の言語を設計する．

（2） WYSIWYG 方式を徹底的に押し進めて，画面の上の出力イメージ上でマウスなどを使って指示を与えるだけでユーザの望みどおりの出力が得られるようにするという方法も考えられる．ただし，この方法は(1)以上にむずかしい問題を含んでいる．なぜならば，マウスなどで表現できる操作は具体的な出力イメージ上の具体的な操作のみなので，システム側はその情報からユーザが一般的にどうしたいのかを推論しなければならない．そうしないと，大きな文書の場合，ユーザは同じような操作を何回も繰り返さなければならないことになる．

（3） TEX のようにかなり細かいレベルの制御ができ，かつ拡張可能なシステムの上にマクロを積み上げて，見かけ上，ユーザ・カスタマイズが容易なシステムをでっち上げるという方向もある．しかし，そのようなシステムの典型例である LATEX でカスタマイズが容易かどうかは意見の分かれるところである．実際に LATEX が使われている現場を見ていると何人かのエキスパートがいて，一般ユーザはたいていのカスタマイズはその人にお願いするという Scribe の Database Administrator 的な状況がみられる．それもそのはずで，LATEX をカスタマイズするには TEX の知識とその上のマクロで実現された LATEX の両方の知識を必要とするのである．したがって，何層にもマクロを積み上げるのも考えものである．

このように考えてくると，なかなか容易なカスタマイズというのは実現しそうにもない．やはり，Scribe のマニュアルが言うように，ユーザは中味にだけ没頭し，文書の種類を指定するだけで，あとはシステム側がすべて面倒をみるというのが正しい方向なのかもしれない．その際には，文書の種類としてたいていの人の要求を満たせるように数多くの書式を用意する必要がある．TEX や LATEX でも同じようなことが起こりつつあり，世界中のあちこちで開発したいろいろなスタイルをネットワークを利用してみんなで共有するという体制ができつつある．したがって，当面の現実

的な解としては各ユーザは自分ではカスタマイズはせず，世の中にある多くのカスタマイズの中から自分の目的にあったものを見つけてきて利用するということに落ちつくのかもしれない．

8. おわりに

文書整形システムの現状として，代表的な文書整形システムを取り上げ，その概要を解説した．さらに，文書整形システムが現在抱えている問題のいくつかを取り上げて，考察を加えた．最後に，将来の文書整形システムのあるべき姿に関して，私見を述べた．ワープロなども含めると文書処理という分野は計算機の応用分野の中で最も重要なものの一つである．今後もこの分野はますます重要さを増していき，テキストデータベースとか，マルチメディア・データベースなどの分野とも関係しながら，さらに発展していくものと思われる．

謝辞 Scribeの例をデバッグして，出力サンプルを得るのにNTT基礎研究所の奥乃博氏に大変お世話になりました．

参考文献

1) Furuta, R., Scofield, J. and Shaw, A.: *Document Formatting Systems: Survey, Concepts, and Issues*, Computing Surveys, Vol. 14, No. 3, pp. 417-472 (Sep. 1982).
2) Kernighan, B. W., Lesk, M. E. and Ossanna, Jr., J. F.: *UNIX Time-Sharing System: Document Preparation*, The Bell System Technical Journal, Vol. 57, No. 6, pp. 2115-2135 (1978).
3) 長谷部紀元：UNIXの文書処理機能，情報処理，Vol. 27, No. 12, pp. 1374-1382 (Dec. 1986).
4) 和田英一：連載エディタとテキスト処理，bit, Vol. 15, No. 4-6 (1983), Runoff (13回), Scribe (14回), TEX (15回).
5) 杉原厚吉：清書プログラム，情報処理，Vol. 20, No. 11 (1979).
6) Saltzer, J. H.: *Manuscript Typing and Editing, TYPSET, RUNOFF*, CTSS Programmer's guide, 2nd, ed., MIT Press (1965).
7) Using nroff AND troff on the Sun Workstation, User Manual, Sun Microsystems Inc. (1986).
8) Scribe, Document Production System, User Manual, UNILOGIC Ltd. (Apr. 1984).
9) Scribe, Document Production System, Database Administrator's Guide, UNILOGIC Ltd. (Apr. 1985).
10) Knuth, D. E.: *The $T_EX$book*, p. 483, Addison-Wesley (1984).
11) Knuth, D. E.: *T_EX: The Program*, p. 594, Addison-Wesley (1986).
12) Knuth, D. E.: *The* METAFONT*book*, p. 361, Addison-Wesley (1986).
13) Knuth, D. E.: METAFONT: *The Program*, p. 560, Addison-Wesley (1986).
14) Knuth, D. E.: *The Errors of T_EX*, Software-Practice & Experience, Vol. 19, No. 7, pp. 607-685 (July 1989).
15) Knuth, D. E.: *Computer Modern Typefaces*, p. 588, Addison-Wesley (1986).
16) 斉藤康己：日本語TEX，情報処理学会，日本語文書処理研究会報告，10-3 (Jan. 1987).
17) 山内長承，来住伸子：TEX文書清書システム，コンピュータソフトウェア，Vol. 4, No. 1, pp. 44-52 (Jan. 1987).
18) Harrison, M. A.: *News from the $V_{OR}T_EX$ Project*, TUGboat, Vol. 10, No. 1, pp. 11-14 (Apr. 1989).
19) 石田晴久，岸田孝一，斎藤信男編：最新UNIX, bit臨時増刊，p. 334，共立出版 (May 1987).
20) Price, L. A.: *SGML and T_EX*, Tugboat, Vol. 8, No. 2, pp. 221-225 (July 1987).
21) 木村 泉：文書整形言語，情報処理，Vol. 22, No. 6, pp. 559-564 (June 1981).
22) Wichura, M. J.: PICTEX : Macros for Drawing PICtures, Tugboat, Vol. 9, No. 2, pp. 193-197 (Aug. 1988).
23) 長谷部紀元，亀山豊久：和欧混合文書組み版システムの試作と組み版規則の検討，情報処理学会日本語文書処理研究会，21-4 (May 1985).
24) 山内長承：汎用欧文清書系を採用した和欧混植清書系に対する考察，情報処理学会，文書処理とヒューマンインタフェース研究会報告，12-1 (May 1987).
25) 工藤 司，藤田 博：TEX出力システムの開発，情報処理学会第28回全国大会予稿集，6D-3, pp. 243-244 (1984).
26) Saito, Y.: *Report on $_JT_EX$: A Japanese T_EX*, TUGboat, Vol. 8, No. 2, pp. 103-116 (July 1987).
27) 日本語TEXテクニカルブックI，アスキー出版技術部責任編集，アスキー出版局 (May 1990).

(平成2年8月3日受付)