日本語Tex体験記

日本語 Tex 体験記

生活環境学部
生活環境学科　米田守宏

1．はじめに

Tex はワークステーション等で広く使われている電子組版システムですが、日本語化され、MS-DOS パソコン上に移植されて、大変使いやすくなっています。最近、筆者は日本語 Tex を手に入れ、使用する機会を得たので、ここでは、その使用体験記といったものを紹介したいと思います。なお、Tex については、以前に野口氏によって紹介されています[1]。

ある、適当に数式の多い論文を書いているとき、数式がきれいに表現できるソフトウェアが欲しいなと思ったのが、Tex と出会ったきっかけです。というのも、通常のワープロは数式記述機能がきわめて貧弱だからです。現在のところ、数式エディタとしてだけ Tex を使っており、ほぼ満足すべき結果を得ています。たしかに、数式記述能力に優れていることは、Tex の特徴のひとつですが、本来、Tex はもっと多くの機能を備えたソフトウェアで、論文や単行本一冊分の高品位な組版ができる能力を持っています。以下に、MS-DOS パソコンへの導入のしかた、筆者による使用例および使用感等について述べたいと思います。

2．日本語 Tex および導入に必要な機器構成

Tex は、スタンフォード大学の Donald E. Knuth 先生が、ご自身の著作を出版するために自ら開発した電子組版システムです。そして、もとの Tex（plain Tex）の上に、マクロをかぶせ、使いやすいコマンド体系に整備したのが LaTex です。この Tex および LaTex を日本語化したものとして、NTT 版 JTex 、アスキー版日本語 Tex 、および日本語 MicroTex がありますが、ここでは、アスキー版日本語 Tex を使用しました[2]。

筆者は日本語 Tex のシステムを NEC PC-9801FX に導入しています。ハードディスクには約 10MB の空きが必要です。さらに、テキストファイルをコンパイルする際に付属のDOSエクステンダを使用するため、CPUは 80386 以上、拡張メモリーは 2 MB以上が必須です。MS-DOS は、ver. 3.1 以上であれば動きます。後で述べるように、本システムではプログラムをテキストファイルとして作成し、これをコンパイルするという方式をとっているため、使い慣れたテキストエディタ（例えば MIFES 等）および日本語FEPを用意します。

3．インストールおよび基本的な操作

マニュアルに従って必要なファイルをハードディスクに導入します。図 1 に本システムのディレクトリ構成および主要なファイルを示します。b:¥tex にシステムを導入し、ここで全ての作業を行うとし

ます。

システムが働くようにするには、あと少しすることがあります。まず、図2に示すように、autoexec.batにコマンド検索用のpathを入れ（図2下線部）、環境変数の設定を行います。Texは大文字、小文字を区別するので注意が必要です。config.sysにはとくに追加することはありませんが、files＝30程度に設定しておきます。ここでシステムにリセットをかけ、再起動します。

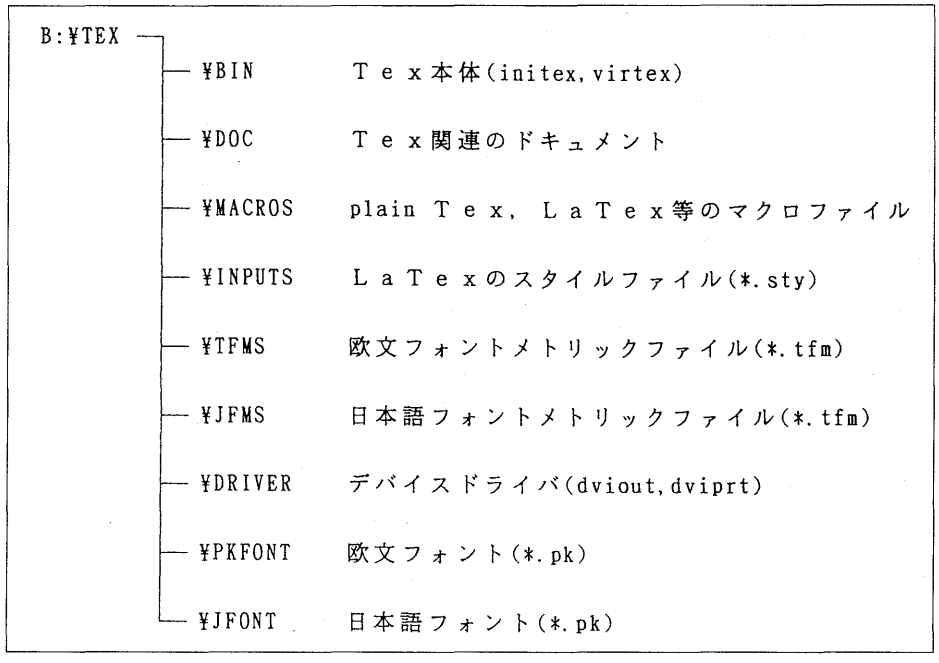

図1　TEXのディレクトリ構成

```
PATH A:¥DOS;A:¥;A:¥UTL;A:¥UTL¥FD;A:¥UTL¥LHA;B:¥MIFES;A:¥QB45¥BIN;
B:¥TEX¥BIN;B:¥TEX¥DRIVER;A:¥VZ;B:¥JUST5


SET texinputs=.;b:/tex/inputs;b:/tex/macros
SET texfonts=b:/tex/tfms;b:/tex/jfms
SET texformats=b:/tex/bin
SET texpool=b:/tex/bin
SET texedit=mifes %%s -%%d
SET texpk=b:¥tex¥pkfont¥^d¥^s.pk;b:¥tex¥jfms¥^s.tfm
SET texknj=b:¥tex¥jfont¥kanji^d.pk
SET texcfg=b:¥tex¥driver
```

図2　AUTOEXEC.BATにおけるPATHと環境変数の設定

次に、b:¥tex¥binに移動し、LaTexのフォーマットファイルを作成します。インストール時にはこのディレクトリにmkfmt.batがコピーされているので次のように実行します。

B:¥TEX¥BIN>mkfmt

これで、フォーマットファイル jlplain. fmt が作成されますが、このファイルの正体が実はマクロパッケージで、Tex 本体（virtex. exe）を起動するときにパラメータ指定してやることで LaTex 形式の処理が可能になります。

LaTex を起動するための次のようなバッチファイル（jlatex. bat）を b:¥tex に作成しておきます。

```
exe386  b:¥tex¥bin¥virtex  &jlplain  %1
```

ここまでの設定にとくに問題がなければ、以後次のように、Tex の文法に沿って作成されたテキストファイル（＊.tex）に対して Tex を実行することができます。

B:¥TEX>jlatex_ファイル名（.tex は省略可）

図3に実行例を示します。実行の結果、DVIファイル（拡張子 .dvi）が作成されるので、これをプレビューワー dviout で画面に呼び出して出来上がりを表示することができます。

B:¥TEX>dviout_ファイル名（.dvi は省略可）

```
B:¥TEX>jlatex center5
B:¥TEX>exe386 b:¥tex¥bin¥virtex&jlplain center5
EXE386 Ver.1.00m (C) 1990 Kyoto Micro Computer Co.LTD. Mem=0015 - 0050
This is TeX, C Version 2.99 - j1.7 (no format preloaded)
(center5.tex
LaTeX Version 2.09 <24 May 1989>
(b:/tex/inputs/jarticle.sty
Document Style `jarticle' <18 Dec 88>.
(b:/tex/inputs/jart10.sty)) (b:/tex/inputs/a4j.sty) (center5.aux) [1]
(center5.aux)
Output written on center5.dvi (1 page, 2544 bytes).
Transcript written on center5.log
```

図3　jlatex の実行例

DVIファイルができなかったとき、あるいは希望した結果で表示されなかった場合は、もとのテキストファイルに戻ってチェックすることになります。もし、出来上がりがこれでよければ、プリンタドライバ dviprt でプリンタに出力します。

B:¥TEX>dviprt_ファイル名（.dvi は省略可）

dviout および dviprt は、それぞれ画面表示およびプリンタ出力のためのデバイスドライバで、b:¥tex¥driver にあります。ただし、これらを動作させるのに必要なパラメータファイル（dviout. cfg, dviprt. cfg）をあらかじめ b:¥tex¥driver¥util から b:¥tex¥driver へコピーしておく必要があります。

以上、b:¥tex で Tex による作業を行う場合の流れ図を図4に示します。前述したような環境下で、マニュアル通りにやれば、あまり問題なくインストールできるはずです。ただし、プリンタドライバ定義ファイル dviprt. cfg を設定するとき、パラメータが多いので、若干の注意が必要です。あと、DVIファイル以外に作成される不要なファイル（＊. log、＊. aux 等）を削除するためのバッチファイルを作っておくと便利でしょう。

4. 使用例

Tex のプログラムの一端にふれていただくため、筆者によるプログラム例を図5に示します。これを、コンパイルし（“Tex にかける”といいます）、プリンタで印字した例を図6に示します（使用し

たプリンタは、キャノン BJ-220JS）。図5と図6の数式番号はそれぞれ対応しています。以下、プログラムについて順を追って簡単に説明していきたいと思います。

第1行の ¥documentstyle というコマンドで文書スタイルの宣言を行っています。{　}の中がスタイルの種類、[　] の中は印刷する用紙の大きさを示しており、システムに用意された情報に従って、日本語の縦書き文書（A4版）として必要な整形を行ってくれます。標準で article、report、book 等が用意されており、大きな文書では、文書スタイルの指定をするだけで、章立て・見出し付け等、自動的に行ってくれます。

第2行 ¥begin{document} は、第33行 ¥end{document} と対応しており、これらで囲まれた範囲が Tex のプログラムであることを宣言しています。原則として、標準テキストファイルを ¥begin{document} と ¥end{document} で囲めば、システムはこれを Tex 文書であるとみなしてくれます。

第3行 ¥begin{large} は、第32行 ¥end{large} に対応しており、これらで囲まれた範囲で、標準で用意されたよりも大きめのフォントを使用することを示しています。

Tex では、コマンドと環境という形で整形を行いますが、これら環境の内容および範囲指定は ¥begin{環境名}と ¥end{環境名}の組み合わせで行います。図5では用いていませんが、主なものとして次のようなものがあります。

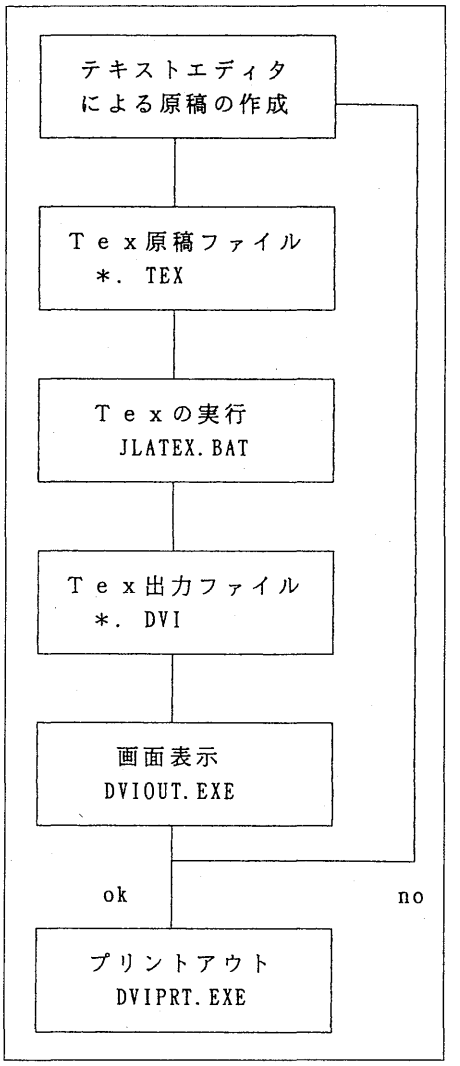

図4　日本語 TEX における作業の手順

quote	: 引　用
center	: センタリング
flushright	: 右寄せ
flushleft	: 左寄せ
itemize	: 箇条書き
enumerate	: 番号付け

環境を指定することで、システムのほうで適当に最適値を判断して、整形を行ってくれます。Tex ではこの他にも各種の環境が用意されており、きめの細かい文書整形を可能としています。

第5行目は、何の制御も行われず、テキストがそのまま出力されている例です。

第8行から第29行までが本文に相当します。ここでは、数式を出力しようとしていますが、Tex では数式出力のため、二通りのモードが用意されています。ひとつは数式モードで、math 環境を使用します。¥begin{math} と ¥end{math} には省略形として“¥(”と“¥)”が用意されています。これは、文中で数式を引用するときに用います。もうひとつは、ディスプレイ数式モードで、displaymath 環境を使用します。これは、文章とは独立して数式を出力するときに用いられますが、省略形として“¥[”と“¥]”が用意されています。第8行目の行頭と行末にあるのがそれで、“[¥”と“¥]”で囲まれた範囲では数式出力に適したコマンドにより制御が行われます。以下、プログラムの入力のしかた、使用されているコマンドについて簡単に説明していきたいと思います。

```
 1: ¥documentstyle[a4j]{jarticle}
 2: ¥begin{document}
 3: ¥begin{large}
 4:
 5: *** 数式の例   [File : center.tex]
 6:
 7:
 8: ¥[ K_V = ¥frac{1}{2} ¥sqrt{¥frac{D_p ¥sigma ¥cos ¥theta_a}{¥mu}} ¥]    %式 1
 9:
10: ¥[ ¥frac{M(t)}{M_{max}} = 1 - ¥frac{8}{¥pi^2} ¥sum_{n=0}^{¥infty}      %式 2
11: ¥frac{¥exp(-D(2n+1)^2 ¥pi^2 t /¥ell^2 )}{(2n+1)^2} ¥]
12: ¥[ ¥hspace{7cm} (n=0,1,2...) ¥]
13:
14: ¥[ ¥frac{¥partial u}{¥partial ¥tau} = ¥frac{¥partial}{¥partial z}      %式 3
15: ¥left¥{ D(u) ¥frac{¥partial u}{¥partial z} ¥right¥} ¥]
16:
17: ¥[ D(u) = D_0 (1 + ¥sigma u) ¥]                                        %式 4
18:
19: ¥[ D = ¥int_0^1 D(u) du = D_0 (1 + ¥sigma/2)  ¥]                       %式 5
20:
21: ¥[ ¥sigma = a_0 + ¥sum_{n=1}^4 a_n ¥tau_{80}^n  ¥]                     %式 6
22:
23: ¥[ B_2(¥tau) = - ¥frac 38 ¥frac{1}{(1 + ¥sigma)} ¥frac{d¥phi(¥tau)}    %式 7
24: {d¥tau} + ¥frac{¥phi(¥tau)}{2} + ¥frac 54 ¥frac{¥sigma}{(1 + ¥sigma)}
25:  ¥phi^2(¥tau) ¥]
26:
27: ¥[ M(¥tau) = ¥ell C_0 ¥left¥{ 1 - ¥frac{1}{32} ¥frac{d¥phi(¥tau)}      %式 8
28: {d¥tau} - ¥frac{17}{24} ¥phi(¥tau) + ¥frac{5}{48} ¥frac{¥sigma}
29: {(1 + ¥sigma)1} ¥phi^2(¥tau) ¥right¥}   ¥]
30:
31:
32: ¥end{large}
33: ¥end{document}
```

図5　TEXのプログラム例（center. tex）

*** 数式の例 [File : center.tex]

$$K_V = \frac{1}{2}\sqrt{\frac{D_p \sigma \cos\theta_a}{\mu}} \tag{1}$$

$$\frac{M(t)}{M_{max}} = 1 - \frac{8}{\pi^2}\sum_{n=0}^{\infty}\frac{\exp(-D(2n+1)^2\pi^2 t/\ell^2)}{(2n+1)^2} \tag{2}$$

$$(n = 0, 1, 2...)$$

$$\frac{\partial u}{\partial \tau} = \frac{\partial}{\partial z}\left\{D(u)\frac{\partial u}{\partial z}\right\} \tag{3}$$

$$D(u) = D_0(1 + \sigma u) \tag{4}$$

$$D = \int_0^1 D(u)du = D_0(1 + \sigma/2) \tag{5}$$

$$\sigma = a_0 + \sum_{n=1}^{4} a_n \tau_{80}^n \tag{6}$$

$$B_2(\tau) = -\frac{3}{8}\frac{1}{(1+\sigma)}\frac{d\phi(\tau)}{d\tau} + \frac{\phi(\tau)}{2} + \frac{5}{4}\frac{\sigma}{(1+\sigma)}\phi^2(\tau) \tag{7}$$

$$M(\tau) = \ell C_0\left\{1 - \frac{1}{32}\frac{d\phi(\tau)}{d\tau} - \frac{17}{24}\phi(\tau) + \frac{5}{48}\frac{\sigma}{(1+\sigma)1}\phi^2(\tau)\right\} \tag{8}$$

図6　dviファイルのプリントアウト例（center. dvi）

はじめに、文字の入力について説明します。アルファベットはそのまま入力します。ギリシア文字は、制御記号¥を用いて例えば次のように入力します。

¥sigma, ¥theta, ¥mu, ¥tau, ¥pi, ¥phi

それぞれ、σ, θ, μ, τ, π, ϕ と出力されます。大文字の場合、一番目の文字を大文字にするという規則があります（例えば、¥Delta: Δ）。下添字、上添字は、それぞれ、_{ }, ^{ }という形で入力します。1文字の場合は{ }は省略可能です。特殊記号も各種用意されており、¥infty（∞）, ¥ast（$*$）, ¥ell（ℓ）等があります。文字の入力に関しては、以上のようにさほど問題がありませんが、Tex の制御に用いられる特殊文字（¥, $, %, #等）の扱いには注意が必要です。

次に数学記号について説明します。

第8行、式1では、分数記号 ¥frac と平方根記号 ¥sqrt が使われています。分数は ¥frac {分子} {分母}という形で使います。式1や式2でわかるように、分子、分母の文字の位置はシステムが自動調節してくれます。平方根 ¥sqrt は、{ }で範囲を指定することにより、式1のように、自動的にその範囲を囲んでくれます。

第10行、式2では、総和記号 ¥sum が使われています。¥sum_{下限} ^ {上限}という形で使います。第19行、式5では、積分記号 ¥int が使われています。¥int_{下限} ^ {上限}という形で使います。このように、数学記号のなかで上限、下限のあるものには一貫した規則があります。第14、15行の ¥partial は、偏微分の記号です。

かっこに関しては、通常はそのまま入力してよいのですが、式にあわせて大きさを自動調節する機能が備わっています。第15行、式3や第27～第29行、式8がその例で、¥left¥{…¥right¥}というように使います。

以上、文書整形、文字入力、数式入力に関する機能のほんの一部を紹介しましたが、Tex の入力法の特徴や書式制御機能の豊富さ、きめ細やかさ等を感じとっていただけたのではないかと思います。

5．日本語 Tex の使い心地について

日本語 Tex のごく一部の機能を使っているだけなので、全体的なことについては、述べることができませんが、使用した範囲内での使い心地について述べてみたいと思います。

言語としての Tex の特徴は、テキストの中に書式制御記号を埋め込んで使う、いわゆるマークアップ型言語（marked-up language）であるということです。この場合、地のテキストの中に制御記号が埋め込まれるため、単なるテキストではなく、テキスト／プログラムの複合体という性格を持つことになります。図5にあるように一見複雑そうに見えますが、これは出来上がったテキスト／プログラムを見ているためであり、見かけほどプログラミングは難しくありません。実際にプログラムを作成する流れに入ってみると、ああしたい、こうしたいという希望（指示）を、文法に従って、次々に入力していくだけの比較的楽な作業であることがわかります。このような、使ってみてわかる作業効率のよさは、製版ないし組版に必要とされる作業を、徹底して論理構造化するという、Tex の基本的設計方針からきていると思われます。テキストファイルをコンパイルする→プレビューワーで見るという作業が容易であり、結果を直ちに確認することができるため、“見たとおり”でないという不安感はそんなにありません。製版に必要なコマンドはいたれりつくせりといえるほど用意されており、プログラムを終えてTex にかけ、結果を得るという一連の作業がちょうど製版の名人に仕事をお願いしているような感じ

になります。つまり、Tex は製版に関するエキスパートシステムといった性格を持つソフトウェアだといえます。

6．Tex の特徴と利用形態について

数式エディタあるいは製版システムとして、Macintosh や Windows で動くソフトウェアがいくつか登場しています（数式エディタとしては Expressionist や Math Type、製版システムでは Aldous Pagemaker 等)。これらのソフトウェアは、“見たとおり”の画面に対して、グラフィカルユーザーインターフェイス（GUI）に従って対話型の作業が可能であり、非常に使いやすいといえます。Tex はこれらと異なり、ユーザーの作業はすべてテキストレベルで行われ（要するに、プログラムを書く)、データ処理はバッチ形式で行われます。従って、GUI ベースのソフトウェアに比べると、とっつきにくい面があることは否定できません。GUI をとるか、テキスト形式による作業形態をとるかについては、好みや慣れあるいはデータの互換性等の問題もあり、議論が分かれることと思いますが、このような比較の上で、Tex を使用することの有利さをまとめてみると次のようになります。

(1) Tex ファイルのデータ交換用標準ファイルとしての、汎用性、流通性のよさ。

(2) マクロによる機能の拡張性のよさ。

(3) 各種スタイルファイルの蓄積とその利用。

(1)は、Tex が主要な大型計算機、各種ワークステーションおよびパソコン等多数の機種に移植されていること、および、Tex 形式に対応している有力なソフトウェア（例えば Mathematica）が多いことなどがその理由です。この流通性のよさは、Tex の現在の利用形態に反映されています。その中でも、重要なもののひとつが、学会誌の論文投稿における Tex の利用です。Tex 方式によれば、著者は本文を書くだけでなく、製版まで指定した形で原稿を作成することになります。受け入れ側に Tex があれば、出来上がった Tex ファイルを電子メールとしてネットワークを通じて投稿し、直ちに製版することができます。このような方法によれば、学会誌出版の大幅な効率化が可能です。実際、アメリカでは Tex 形式での論文投稿を受け付けている例（例えばアメリカ数学会、アメリカ化学会等）は多いようです。この方式を推し進めれば、研究成果の発表形態として紙を用いた論文誌を出版するという従来の方式自体が変わっていく可能性も考えられます。（このあたりの事情は、アメリカにおけるネットワークの普及も関係していると思われますが。）

(2)の例としては、各種用途に特化した Tex の存在があげられます。数式記述能力を向上させた Ams-Tex、文献データベース作成用の Bib-Tex、その他描画用や楽譜記述用の Tex があります。

(3)は、公用・私用にかかわらず各種定型文書の作成に威力を発揮するものと思われます。Tex の場合、¥documentstyle{　}コマンドにより、文書スタイルの詳細な指定が容易であり、さらに既製のスタイルファイルを利用するだけでなく、自分流に改造したり、新しいスタイルファイルを作成し引用することも可能です。特定の学会誌向けのスタイルファイルの作成も考えられます（学会論文も一種の定型文書である）。

仕上がりの美しさ、機能の豊富さ、データ形式の流通性のよさに加えて、以上のような拡張性のよさも Tex の魅力のひとつであり、多くのユーザーのプログラミング意欲をかき立てているようです。

7．おわりに

パソコン上では、ワープロ、製版用ソフトウェアあるいは数式エディタとして、Macintosh や Windows で動く、GUI ベースの優れたソフトウェアがいくつか現れています。それらに対する Tex の特徴は、これまで述べてきたように、プログラムによる機能の拡張性をはじめとする自由度の高さといえるでしょう。ですから、プログラミングに対してさほど抵抗のない人であれば、Tex とテキストエディタの組み合わせというのも文書処理法の選択肢のひとつとしてあり得るのではないかと思い、ここにご紹介した次第です。Tex については、各種解説書が出版されています[3,4,5]ので、興味を持たれた方は参照してください。仕上がりの美しさについては、野口氏も指摘されていましたが、一連のプログラムを書き終えて、Tex にかけ、仕上がりを初めて見る瞬間というのは結構感動するものがあります。これも、プログラミングの醍醐味のひとつといえるかもしれません。

文　献

［1］ 野口誠之、“欧文・数式清書システム Tex で論文を書く方法について”、奈良女子大学情報処理センター広報、創刊号、p.119（1989）

［2］ “パーソナル日本語 Tex”、アスキー書籍編集部編（1992）

［3］ Leslie Lamport、“文書処理システム LaTex”、Edgar Cook、倉沢良一監訳、アスキー出版局（1990）

［4］ 野寺隆志、“楽々 LaTex”、共立出版（1990）

［5］ すずきひろのぶ、“やさしい LaTex のはじめかた”、オーム社（1991）