

X 6 8 k

Programming Series

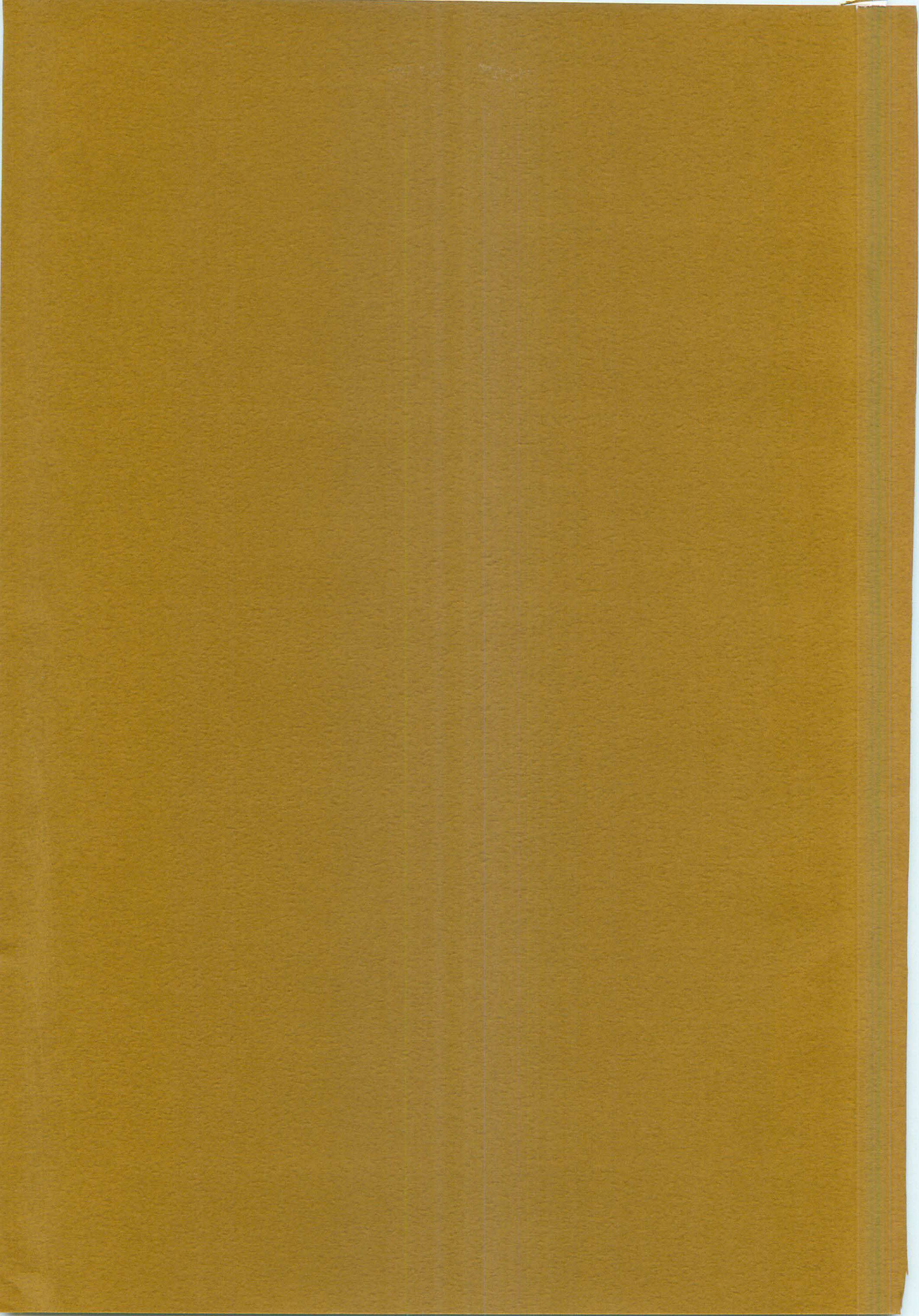
吉野智興+川本琢二+山崎岳志+実森仁志……………共著

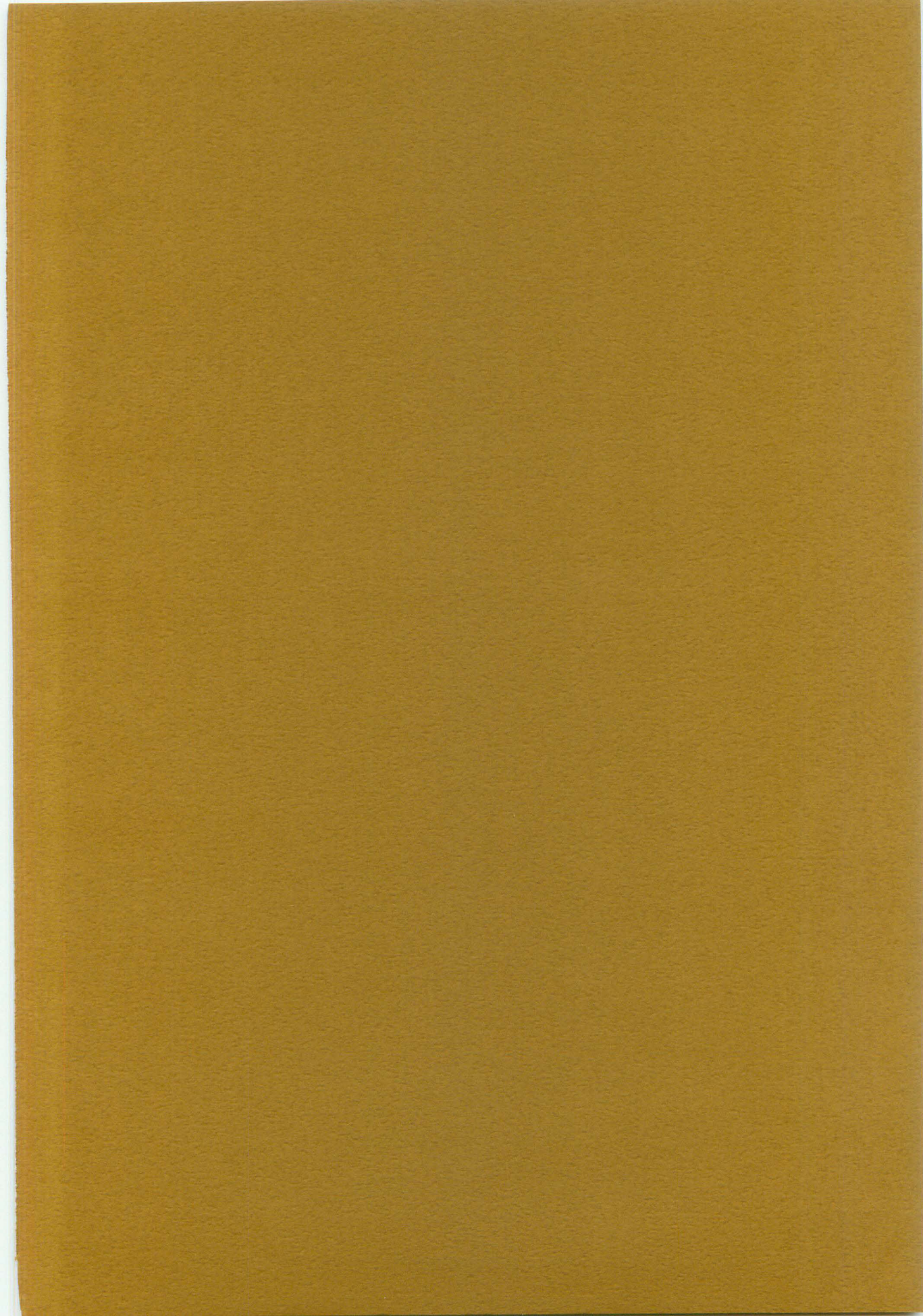
(#3)

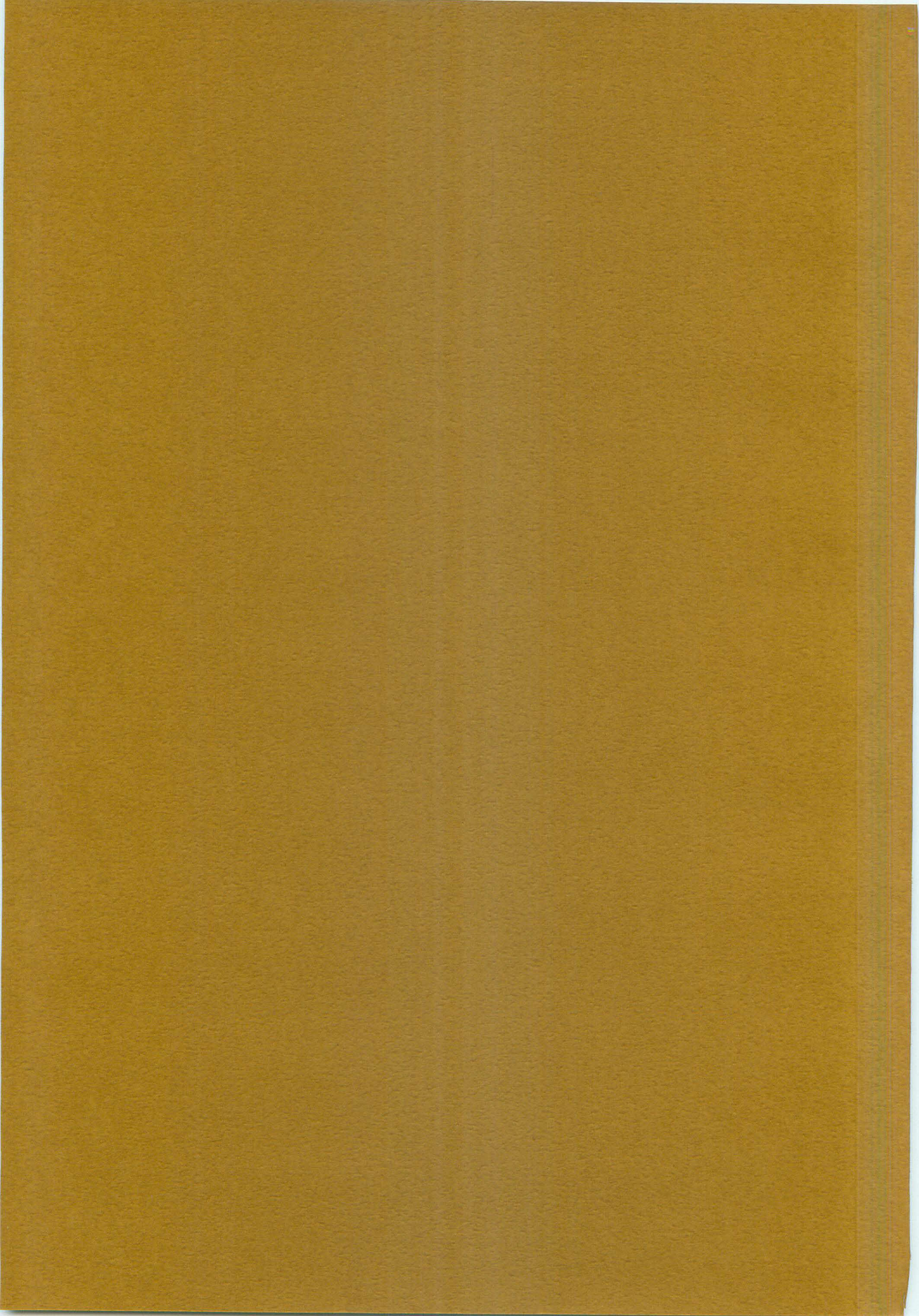
X680x0 TEX

Vol.
1

User's Guide







X 6 8 k

Programming Series

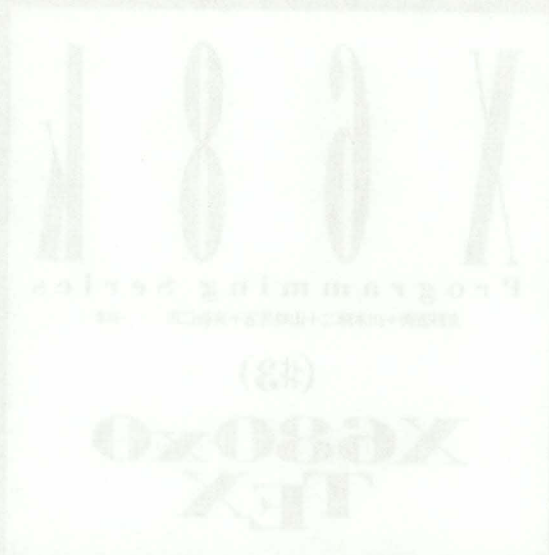
吉野智興・川本琢二・山崎岳志・実森仁志……共著

(#3)

X680x0 TeX

Vol.
1

User's Guide



-
- T_EX は、American Mathematical Society の登録商標です。
 - METAFONT は、Addison-Wesley Publishing Company の登録商標です。
 - UNIX は、UNIX System Laboratories Inc. の登録商標です。
 - その他、本書に登場するシステム名、製品名などは、一般に各社の商標または登録商標です。本文中では、特に TM、® マークは明記していません。
-
- T_EX : © 1982 Donald E. Knuth
 - 日本語 T_EX : © 1992 ASCII Corporation
 - 縦組日本語 T_EX : © 1994 ASCII Corporation, Impress Corporation
-

本書に添付されたディスク内の各アーカイブファイルについては、各アーカイブファイルごとに定められた配布規定に従って再配布することができます。

本書の本体価格には、フリーソフトウェアファウンデーション (FSF) への寄付金が含まれています。

はじめに

この『X680x0 TeX』は、X68000 / X68030 で動く TeX のほぼフルセットの動作環境を、初心者でも比較的簡単に使えるようにインストーラ付きで提供するという目標で作成されました。また、市販の X68000 / X68030 につながるプリンタをほぼすべてサポートしたドライバ類や、TeX での使用範囲にとどまらないフォントマネージャも提供されています。

その一方で、この本では、TeX をできるだけ平易に説明しながら、TeX を使うのに最低限必要な知識を記述してあるつもりです。ですが、TeX は非常に巨大で複雑なシステムなので、一朝一夕で使いこなせるものではないのも事実です。TeX を使っていて「こんなことはできないのだろうか?」とか「思うように作成できない」とかいった壁に突き当たってしまうかもしれません。

しかし今日、幸いなことに TeX についての参考書籍が出版されています。それらに書かれている内容は、ほぼすべて、この X680x0 版 TeX にもあてはまるはずですから、あなたが抱いた疑問点のうちのいくつかを解決してくれるに違いありません。それでも、ときには知らない言葉が出てきたり、X680x0 版 TeX に特有の概念に行き当たったりすることがあるでしょう。そのようなときには、もう一度この本を読み返してみてください。手がかりとなる記述が見つかるはずです。

さらに、添付ディスクには、TeX を使ううえで便利なプログラムもたくさん収録してあります。すべてを使えるようになった頃には、すでに TeX は、あなたが自由に使えるパーソナルパブリッシングシステムとして十分に役に立つ存在になっていることでしょう。

1994 年 6 月吉日

著者を代表して
吉野智興

X68k Programming Series #3

T_EXVol.1
User's Guide

C O N T E N T S

はじめに	iii
------------	-----

Chapter 1	T _E X	1
-----------	------------------	---

1.1	X680x0 での T _E X の歴史	2
	• T _E X とは	2
	• X680x0 版 T _E X の歴史	4
1.2	X680x0 の T _E X システム	6
	• X680x0 版 T _E X の対象機種	6
	• T _E X の処理の流れ	6
	• T _E X の種類と実行ファイル	10
	• より高度な環境構築	14
	• T _E X のコマンドライン書式	15
	• T _E X が参照する環境変数	16
	• T _E X のための X680x0 のセットアップ	18
	• T _E X が使うファイルの拡張子	19

Chapter 2	Install	21
-----------	---------	----

2.1	最低限必要な環境	22
2.2	インストーラによるインストール	25
	• インストールディスクの説明	25
	• インストールの前に準備しておくこと	30
	• 一括インストール	32
	• 分割インストール	37
	• インストールチェック	39
2.3	インストールディレクトリ	43
2.4	インストーラのエラーメッセージ一覧	45

Chapter 3	Device Driver	47
-----------	---------------	----

3.1	デバイスドライバ	48
3.2	フォントマネージャ — 文字の変形	50
	• 概要	50
	• フォントドライバ	51
	• フォントマネージャの常駐	54
3.3	プレビューア — 画面での閲覧	56
	• プレビューアの使用法	56
	• プレビューアの起動方法	60
	• プレビューア画面表示中のキー操作	61
	• エラー対策	63
3.4	プリンタドライバ — プリントアウト	64
	• プリンタドライバの使用法	64

	● 自分のプリンタに対応させる	65
	● 起動方法	66
	● エラー対策	66
3.5	ファックスファイル変換	68
	● ファックス送受信の手順	69
	● \TeX のファイルを Starfax 形式へ変換する	69
	● Starfax 形式ファイルを \TeX へ変換する	73

Chapter 4 Exercise 77

4.1	はじめに	78
4.2	\LaTeX で遊ぶ	79
	● \LaTeX とは	79
	● \LaTeX を使ってみる	80
	● \LaTeX による処理	82
	● dvi ファイルをプレビューする	91
	● プリンタで出力する	93
	● \LaTeX へようこそ	94
4.3	最小の \LaTeX ガイド	96
	● コントロール・シーケンス	97
	● 特殊文字	97
	● \LaTeX ソースの構造	103
	● 最小の「お約束」	110
4.4	論理構造を表す環境	112
	● 章節にかかわる構造	113
	● 引用にかかわる構造	116
	● 箇条書きにかかわる構造	118
	● その他の文章構造	124
	● ソースに忠実な出力をする構造	126
	● 数式にかかわる構造	130
	● “グルーピング”にかかわる構造	132
4.5	視覚構造を制御する	137
	● 文字の大きさを変更する	137
	● フォントを変更する	140
	● フォントテーブルの利用	145
	● その他の視覚構造を制御する	149
4.6	「始まり」の終わりに	157
	● 日常あるいは平穏な日々	157
	● \LaTeX nician への道	159

Chapter 5 Customize 161

5.1	はじめに	162
5.2	フォントマネージャの高度な利用	163
	● フォントマネージャのコンフィギュレーションファイル	163
	● ツァイトフォントとキャッシュフィルタの組み合わせ	165
	● ツァイトフォントとダイエットフィルタの組み合わせ	177
	● FM TOWNS の丸文字フォントを使用する	180
5.3	print.cfg の作り方	186
	● プリンタドライバのオプション	186
	● ドライバ制御系を設定する	188
	● プリンタ制御系を設定する	195
	● テスト出力してみる	204
	● 高速化を目指して	206
	● 再度のテスト出力をしてみる	210
	● コンフィギュレーションファイルの完成	212
5.4	METAFONT	214
	● METAFONT の概要	214
	● makefont によるフォント作成	220

Chapter 6	$\text{T}_{\text{E}}\text{X}$ family	229
6.1	書式説明	230
6.2	plain $\text{T}_{\text{E}}\text{X}$	232
6.3	$\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$	233
6.4	$\text{V}_{\text{F}}\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$	234
6.5	$\text{S}_{\text{L}}\text{T}_{\text{E}}\text{X}$	236
6.6	$\mathcal{A}\mathcal{M}\mathcal{S}-\text{T}_{\text{E}}\text{X}$	238
6.7	$\mathcal{A}\mathcal{M}\mathcal{S}-\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$, $\mathcal{L}\mathcal{A}\mathcal{M}\mathcal{S}-\text{T}_{\text{E}}\text{X}$	240
6.8	Chem- $\text{T}_{\text{E}}\text{X}$, $\text{X}_{\text{M}}\text{T}_{\text{E}}\text{X}$	242
6.9	$\text{M}_{\text{U}}\text{T}_{\text{E}}\text{X}$	244
6.10	Music $\text{T}_{\text{E}}\text{X}$	246
6.11	multicol.sty	248
6.12	Tarticle.sty	250
6.13	$\text{P}_{\text{I}}\text{C}_{\text{T}}\text{E}_{\text{X}}$	252
6.14	epic.sty, eepic.sty, ecleepic.sty	253
6.15	epsf.sty, eclepf.sty	254
6.16	Tpic, Gpic	255
6.17	xfig, transfig	256
6.18	GNUPLOT	258
6.19	GHOSTSCRIPT, dvi2ps	260
6.20	lips3dvi	261
6.21	$\text{B}_{\text{I}}\text{B}_{\text{T}}\text{E}_{\text{X}}$	262
6.22	Makeindex	263
6.23	plain2	264
6.24	dvi2tty	266
6.25	Nemacs or Mule, and Emacs Lisp	268
6.26	μE macs	269
	あとがき	270
	Index	273

COLUMNS

<code>lndrv.x</code>	13
Big $\mathrm{T}_{\mathrm{E}}\mathrm{X}$	29
フォントミキサ	53
ファックスモデムと $\mathrm{T}_{\mathrm{E}}\mathrm{X}$	69
$\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ の出力性能	94
コントロール・シーケンス その1 ～コントロール・ワード～	98
コントロール・シーケンス その2 ～コントロール・シンボル～	99
コントロール・シーケンス その3 ～プリミティブ～	104
日本語 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ における全角文字の扱い	106
$\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ について その1 ～論理構造と視覚構造～	116
$\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ について その2 ～ $\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ の使い方～	128
$\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ について その3 ～コントロール・シーケンスの埋め込み～	136
フォントの大きさとデザイン	139
<code>em</code> と <code>ex</code>	140
New Font Selection Scheme (NFSS)	141
フォント倉庫キャッシュフィルタ	168
ポイント数とドット数	172
<code>newfm_fix.lzh</code> その1 ～フォントマネージャの複数登録について～	180
<code>newfm_fix.lzh</code> その2 ～ $\mathrm{V}^{\mathrm{F}}\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ と <code>newfm_fix.lzh</code> ～	185
$\mathrm{T}_{\mathrm{E}}\mathrm{X}$ が想定するデフォルトサイズ	189

TEX

$\text{T}_{\text{E}}\text{X}$ は、スタンフォード大学の Donald E. Knuth 博士によって作成されたコンピュータによる組版システムです。しばしば誤解されるのですが、 $\text{T}_{\text{E}}\text{X}$ はワープロとはまったく異なる発想のもとに生みだされたシステムで、ワープロとは次元の異なる高性能と高機能とを兼ね備えています。本章では、 $\text{T}_{\text{E}}\text{X}$ とワープロとの違い、X680x0 での $\text{T}_{\text{E}}\text{X}$ の歴史、必要な環境の整備や起動方法などについて説明しています。この章はあくまで $\text{T}_{\text{E}}\text{X}$ の本体を動かす方法について説明してあるだけで、本書の添付ディスクによる $\text{T}_{\text{E}}\text{X}$ のインストール方法や、 $\text{T}_{\text{E}}\text{X}$ のソースの記述方法等については説明していませんのでご注意ください。

1.1 X680x0 での T_EX の歴史

T_EX は、スタンフォード大学の Donald E. Knuth 博士が作成されたコンピュータによる文書作成システムです。ここでは、T_EX の概要と、X680x0 での T_EX の歴史について説明したいと思います。

1.1.1 T_EX とは

しばしば誤解されやすいのですが、T_EX は一般にいう日本語ワードプロセッサとは次元の異なったソフトウェアです。日本語ワードプロセッサは、手書きではちょっと見栄えが悪い場合とか、大量に同じ文書を配布したいとかいう場合に、日常レベルでは問題がない程度のきれいな印刷結果を手軽に得られるため、パーソナルコンピュータの代表的なアプリケーションのひとつになっています。日本語ワードプロセッサは WYSIWYG¹⁾ の代表的ソフトウェアで、作成された文書はディスプレイに表示されたものとはほぼ同じ形で、実際に印刷されます。WYSIWYG とは、画面上に表示されたもの (書体・大きさ・行間・縦横の比率など) をそのまま印刷結果として手に入れられることを意味しますが、その性能はさまざまです。WYSIWYG の代表格を強いて挙げれば、Macintosh (通称 Mac) のソフトウェア群でしょう。PC-9801 でよく使われる「一太郎」や「花子」も WYSIWYG ライクな印刷が可能なソフトウェアですが、出版などの本格的業務では Macintosh のほうがよく使われています。

T_EX は、日本語ワードプロセッサとは異なり、印刷された結果とその元になる文書とがまったく見た目が異なったものになっています。その能力の一端をご覧くださいませう。

T_EX で作成した文書のサンプルです。T_EX では単に文章をきれいに印刷できるだけでなく、参照なども自動で行ってくれます。たとえば、この文章は第 1.1.1 項「T_EX とは」、p.2 に印刷されています。

このサンプル部分の原稿 (ソースファイル) は List 1-1 のような記述がしています。

1) What you see is what you get. の頭文字をとっています。

List 1-1 • \TeX のソースファイル

- 1: \TeX で作成した文書のサンプルです。 \TeX では単に文章をきれいに
- 2: 印刷できるだけでなく、 $\{\text{\Large}$ 参照なども自動 $\}$ で行ってくれます。
- 3: たとえば、この文章は $\text{\ref{\text{\TeX}}}$ とは、 $\text{\p.\pageref{\text{\TeX}}}$ 実例 $\}$ に
- 4: 印刷されています。

\TeX について知らない人が \TeX で作成された List 1-1 のような原稿を見た場合には、日本語に不思議な英語が交じった異世界の文書に見えるでしょう。

\TeX に比べると、日本語ワードプロセッサは、どちらかといえばコンピュータの素人向けに設計されているため、操作方法は簡易になっているのが普通です。それでも巷にワープロ教室が林立するほど、一般の人にとってはコンピュータで文章を作成することは難しいことなのです。その難しいコンピュータでの文書作成を行う \TeX は、もっぱらコンピュータに十分に慣れた人を対象につくられていますので、 \TeX を操作するには、まず、自分のコンピュータを操れるだけの知識が要求されます。これらの大きなハンディがあるにもかかわらず、 \TeX は急速に普及しつつあります。その理由はどこにあるのでしょうか。

まず、ひとつは \TeX が「組版システム」であるという点です。たとえば、皆さんが、今、手にしているこの本も \TeX で作成されています。日本語ワードプロセッサで作成された文書は、通常の方法ではこのような美しく印刷された本にはなりません。最近のアウトラインフォントを用いた高解像のプリンタを用いても、本の印刷に使われるものとは解像度の次元が異なっています。

また、 \TeX は日本語ワードプロセッサが知らないような組版の知識、たとえば英語と日本語が混在しているような場合の文字の詰め方²⁾などについての知識を持っていて、これを使って文書进行处理します。このため、 \TeX で作成されたデータは普通のプリンタで印刷しても、通常の日本語ワードプロセッサの印刷より美しく文字がレイアウトされます。これは D. E. Knuth 博士が印刷所の作成する出版物のレイアウトやフォントがなかなか自分の思うようにならないことに不満を持って \TeX を開発したという経緯を思い起こせば当然のことともいえるでしょう。

もうひとつは、 \TeX には機種依存性がないという点です。X680x0 で処理できる \TeX で作成された文書は、この本で説明されている X680x0 の拡張部分さえ使わなければ³⁾、PC-9800 シリーズ用に配布されている \TeX でも、ワークステーション上で動いている \TeX でも処理できます。この互換性は、 \TeX で作成された文書の原稿だけでなく、 \TeX で処理された結果においても維持されています。たとえば、PC-9801 上で \TeX の処理を行い、X680x0 で印刷をすることも、なんら問題なく行うことができます。このような、同一の文書を異なった環境下で作成することを可能にしてくれる \TeX の特徴は、日本語ワードプロセッサにはまずないものといえます。

このような優れた特徴を備えているために、コンピュータ導入に比較的敷居が低い世界⁴⁾では \TeX による出版物の作成が一般化⁵⁾しつつあります。

2) \TeX では、英文字と日本語文字が混在した場合、英単語をハイフンで分けて改行する処理をできるだけ行わないように日本語部分の文字間隔を調節します。

3) 主にフォントマネージャの柔軟性に頼っている部分のことです。たとえば、第6章「 \TeX family」の p.234 で紹介する VF- \LaTeX などはその典型で、X680x0 だけが実現しているものです。

4) 理系の学術論文の世界やコンピュータ関係の出版の世界。

5) とはいえ、「一太郎」に慣れてしまった上司を持つ人は使いたくても使えないそうですが。

1.1.2 X680x0 版 TeX の歴史

X680x0 は、TeX が主に動作している、CPU に 680x0 を持ったワークステーションのアーキテクチャに近いパーソナルコンピュータです。このため、元来は TeX の動作に最も支障が少ないパーソナルコンピュータのひとつなのです。

しかし、実際に TeX が X680x0 上で動くようになるまでにはたくさんの方々の努力が必要でした。シャープから発売された XC コンパイラ Ver.1.00 は TeX はもとより、UNIX 上で動くプログラムをコンパイルするには信頼性と性能が低すぎました。それでも有志の方々は、だましだまし UNIX 上のプログラムを少しずつ X680x0 に移植してきましたが、本格的な移植が始まったのは GNU C Compiler が X680x0 上に移植されてからのことです。GNU C Compiler (以下、GCC と書く) は、Free Software Foundation (以下、FSF と書く)⁶⁾ が配布している C コンパイラで、XC と比較すると非常に高速で、信頼性の高い C コンパイラです。この GCC を大変な努力の末、移植されたのが近藤真己氏でした。この近藤版 GCC なくしては、現在の X680x0 の TeX、X680x0 の多数のフリーソフトはこの世に存在しなかったでしょう。

この近藤版 GCC のリリースをきっかけに多数の UNIX 上のフリーソフトが X680x0 に移植されました。そのなかのひとつが TeX システムだったのです。最初の X680x0 版の TeX は、前田薫氏⁷⁾によって X680x0 の世界にもたらされました。この TeX は、現在の X680x0 版 TeX とは異なる、英語専用の TeX でした。英語専用の TeX では日本語を直接処理することはできません。

ここで、少し X680x0 版 TeX の話から離れて、TeX の日本語対応の話をしておきましょう。TeX は英語圏で作成されたソフトウェアなので、元来、日本語を認識できませんでした。しかし、TeX の優れた機能をなんとか日本語でも使えないものかという声上がるのは当然のなりゆきでした。

TeX で日本語を扱うようにする試みには 2 つの流れがあります。ひとつは「NTT-JTeX」と呼ばれるもので、もうひとつは「日本語 TeX」と呼ばれるものです。双方とも TeX で日本語を扱えるように拡張したものです。この両者はソースレベル上ではほぼ互換性がありますが、TeX で処理したあとの dvi ファイル(後述)では互換性がありません。日本語フォントを扱う TeX へのインプリメントがまったく違った方法で行われているためです。

パーソナルコンピュータでは、PC-9800 シリーズ上の TeX が、日本語 TeX である点、印刷やディスプレイ上の閲覧フォントの処理をドライバ側に追い出している点などのために一般化しつつありますが、こちらは後者の日本語 TeX です。X680x0 版の TeX もこの日本語 TeX です。現時点では、どちらの TeX が最終的に“日本語を扱える TeX”として主流となるのかは不明です。しかし、アスキー社からパーソナルコンピュータでは圧倒的に多数を占める PC-9801 シリーズで稼働する日本語 TeX を収録した『パーソナル日本語 TeX』が発売されたので、おそらくこれが日本語 TeX の主流になるものと思われます。なお、インプレス社からは Microsoft Windows 3.1 上で動作する TeX を収録した『TeX for Windows』が発売されています。

6) GNU プロダクトと呼ばれる一連のフリーソフトウェアを開発しているアメリカの団体です。主宰者は Richard Stallman という天才プログラマーです。

7) 元東大理論科学グループ(略称、TSG)の前田薫氏です。

X680x0 版 T_EX は当初、日本語を認識できませんでしたが、X68000 に移植した前田氏が T_EX の入手方法を整備してくれたおかげで、徐々に X680x0 ユーザに広まっていきました。この配布版に含まれていたプレビューアは PC-9801 から移植されたもので、X680x0 で使うにはハードウェアの力を生かしきったものではありませんでした。本書に添付されているプレビューアの作者で、本書の著者の 1 人でもある川本琢二氏が、最初にプレビューアを改善・公開したのが、1989 年後半でした。川本氏が公開したプレビューアは、当時 PC-9801 で動いていた日本語マイクロ T_EX⁸⁾ の出力結果も X680x0 で見られるように改良していました。

こうして日本語を表示できるプレビューアができたのに、日本語を処理できる T_EX がないというのは悲しいものです。そこで、筆者 (吉野) はワークステーションで動いているアスキー版日本語 T_EX を X68000 へ移植することを試みました。正直に言えば、この移植は C 言語でプログラムが記述できる人なら誰にでもできるという程度のものです。たいした苦勞もなく、アスキー版日本語 T_EX は X680x0 で動くようになりました。日本語が処理できるようになった X680x0 版の T_EX は、パソコン通信を通じて急速に広まっていきました。その後、何度かアスキー日本語 T_EX はバージョンアップを重ねていますが、X680x0 用 T_EX も世間に遅れない程度のバージョンアップを行えていると思います。現在では、ワークステーション上のアスキー日本語 T_EX と同じ T_EX が PC-9800 シリーズでも動くようになったので、T_EX を主題にした雑誌の特集も組まれるようになりましたが⁹⁾、X680x0 で日本語が扱える T_EX が動きはじめたのは、それより 3 年ほど前¹⁰⁾ のことでした。

X680x0 版の T_EX は、処理速度では現在の PC-9801 上の T_EX に及びませんが、T_EX 専用のモードを備えたエディタや多数のプリンタ、フォントをサポートしたドライバ群に関してはまだ一日の長があるはずです。現在でも、大勢の T_EX ユーザが X680x0 で T_EX での文書作成を行っています。X680x0 での T_EX 環境を、この本と添付ディスクで存分に楽しんでいただけたら幸いです。

8) MS-DOS でも稼働できるように規模を縮小した日本語 T_EX です。

9) 『C magazine』(ソフトバンク) 1993 年 10 月号です。

10) 『C magazine』(ソフトバンク) 1990 年 11 月号で、筆者 (吉野) が紹介記事を書いています。

1.2 X680x0 の TeX システム

対話的に作業を行うインタプリタが日本語ワードプロセッサだとすれば、TeX の処理の流れは、一気に作業を行うバッチ動作のコンパイラに相当するものといえます。日本語ワードプロセッサには明確な処理の流れというものが存在せず、常時対話的に印刷や編集作業を行います。しかし、TeX では編集や印刷といった処理が明確に分離されています¹⁾。また、TeX を使うには、ある程度 Human68k の知識が必要です。Human68k については X680x0 に添付されている Human68k のマニュアルを参照してください。

1) これらの一連の作業を行うためのプログラムは、文書を作成するためのテキストエディタを除いて、すべて本書の添付ディスクに収録されています。

1.2.1 X680x0 版 TeX の対象機種

X680x0 版の TeX は、X680x0 の全シリーズに対応しています。特に機種は限定しませんが、実際に使った場合の効率、マシンの実行速度相当の差があると考えて差し支えないでしょう。ですから、できるだけ高速のマシンで大量のメモリを実装していることが快適な TeX 環境を約束するものと思ってください。ただし、印刷や閲覧ではメモリバッファの関係で、メモリが少ない高速マシン²⁾とメモリがフル実装の X68000 では、メモリが多い低速マシンのほうが快適な場合もあります。CPU 速度かメモリ量かを選ぶなら、メモリを選ぶのもひとつの方法かもしれません。

2) たとえば 4M バイトの X68030。

1.2.2 TeX の処理の流れ

TeX は、ワードプロセッサのようにひとつのプログラムで原稿の作成から印刷までを行うシステムではありません。普通のワードプロセッサでは、メニューのような画面で編集や印刷を選んだり、編集画面に印刷用の機能を動かす操作方法があったりしますが、TeX では「編集」「閲覧」「印刷」といった作業は全部独立していて、個別のプログラムで行うようになっています。このために、パソコン初心者には使いはじめるまでに乗り越えなければならないハードルが多すぎて、使いこなすのが難しくなっています。ここで、TeX での原稿作成の流れを見てみましょう³⁾。

3) とにかく使ってみたい人は第 4 章「Exercise」(p.77)を先にお読みください。

(1) 原稿となるファイルを作成する

原稿となるファイル（ソースファイル）を、テキストエディタで作成します。このとき、執筆者は原稿中に T_EX のコマンド（「コントロール・シーケンス」とも呼ぶ）を埋め込んでおきます。T_EX は、この埋め込まれたコマンドを解釈して組版を行います。通常、ソースファイルの拡張子は “.tex” を使います。List 1-2 は、6 ページの文章のソースです。

(2) T_EX で dvi ファイルに変換する

T_EX を使ってソースファイルを「dvi ファイル」と呼ばれるファイルに変換します。dvi ファイルは、拡張子が .dvi のファイルで、T_EX が出力する文字の位置や大きさ、種類を、機種依存しない形で記録した印刷情報ファイルです。T_EX がこの dvi ファイルを作成する時点で誤った記述を発見してエラーを出力することがあります。C コンパイラがプログラムの文法エラーを指摘するように、ソースファイルに T_EX の書式上の間違いがあった場合には、T_EX はエラーを報告して執筆者に変更を要求します。なお、この処理で T_EX はフォントの大きさ・配置などの情報を、拡張子が .tfm であるフォント情報ファイルから読み取ります。この tfm ファイルが見つからない場合は、エラーを出力します。この場合には METAFONT⁴⁾ を用いて tfm ファイルを作成します。

(3) dvi ファイルをプレビューする

dvi ファイルをプレビューしても、最初からきれいに画面上に出力されることはめったにありません。これは、コンパイルは通ったとしても、プログラムが正しく動くとはかぎらないというのと同じことです。この場合は、指定どおりに文書が作成されているかどうかをディスプレイで確認します。指定どおりになっていなければ、修正箇所を見つけて、気に入った配置になるまでエディタでの編集、T_EX での変換を繰り返します。

(4) プリンタで印刷する

指定どおりの文書が作成できたら、実際にプリンタで印刷します。T_EX の出力は、単にプリンタで印刷するだけでなく、高精度な写植機から印画紙として出力したり、製版フィルムに出力したりして、印刷所で印刷することもできます⁵⁾。

4) METAFONT については第 5.4 節「METAFONT」(p.214) を、METAFONT によってフォント (tfm ファイルを含む) を作成する方法については第 2 章「Install」(p.21) を、それぞれ参照してください。

5) もっとも、何百万円もする印刷コストを個人で負担できるかどうかは知りませんが。

List 1-2 • T_EX のソースの例

```

1: % ※ '%' は '%' から改行まで (改行コードを含む) がコメントであることを表し
2: %   ます。よって、ここはコメントです。
3: % ※なお、このソース中では標準の LaTeX 環境には設定されていない独自の
4: %   コントロール・シーケンスが使用されている部分もありますから、注意
5: %   してください。
6: % ※節見出しを書く
7: \subsection{\TeX の処理の流れ}
8: \label{\TeX の処理の流れ}
9:
10: \TeX は、ワードプロセッサのようにひとつのプログラムで原稿の作成から
11: 印刷までを行うシステムではありません。
12: 普通のワードプロセッサでは、メニューのような画面で編集や印刷を
13: 選んだり、編集画面に印刷用の機能を動かす操作
14: 方法があったりしますが、\TeX では「編集」「閲覧」「印刷」と

```

```

15:   いった作業は全部独立していて、
16:   個別のプログラムで行うようになっています。
17:   このために、パソコン初心者に
18:   は使いはじめるまでに乗り越えなければならないハードルが多すぎて、
19:   使いこなすのが難しくなっています。
20:   ここで、\TeX での原稿作成の流れを見てみましょう%
21:   \footnote{とにかく使ってみたい人は\crefp{chap:TeXex}を
22:   先にお読みください。}%
23:   。
24:
25:   \begin{enumerate}
26:   {\bf {\item 原稿となるファイルを作成する}}\
27:     原稿となるファイル（ソースファイル）を、テキストエディタで
28:     作成します。このとき、
29:     執筆者は原稿中に\TeX のコマンド（「コントロール・シーケンス」と
30:     も呼ぶ）を埋め込ん
31:     でおきます。\TeX は、この埋め込まれたコ
32:     マンドを解釈して組版を行います。
33:     通常、ソースファイルの拡張子は
34:     ‘‘{\tt .tex}’’を使います。 \progref{TeXのソースの例}は、
35:     \pageref{TeXの処理の流れ}ページの文章のソースです。
36:   {\bf {\item \TeX で {\tt dvi}ファイルに変換する}}\
37:     \TeX を使ってソースファイルを {\tt dvi}ファイル\ein*{dvi ファイル}と
38:     呼ばれるファイルに変換します。
39:     {\tt dvi}ファイルは、拡張子が{\tt .dvi}のファイルで、
40:     \TeX が出力する文字の位置や大きさ、種類を、機
41:     種依存しない形で記録した
42:     印刷情報ファイルです。 \TeX がこの{\tt dvi}ファイルを作
43:     成する時点で誤った記述を発見してエラーを出力することがありま
44:     す。C コンパイラがプログラムの文法エラーを
45:     指摘するように、ソースファイルに \TeX の書式上の
46:     間違いがあった場合には、\TeX はエラーを報告し
47:     て執筆者に変更を要求します。
48:     なお、この処理で\TeX はフォントの大きさ・配置など
49:     の情報を、拡張子が {\tt .tfm}
50:     であるフォント情報ファイルから読み取ります。
51:     この{\tt tfm}ファイルが
52:     見つからない場合は、エラーを出力します。
53:     この場合には\META%
54:     \footnote{\META については\crefp{chap:METAFONT}を、
55:     \META によってフォント（{\tt tfm} ファイルを含む）を
56:     作成する方法については\crefp{chap:install}を、それぞれ参照し
57:     てください。}を用いて{\tt tfm}ファイルを作成します。
58:   {\bf {\item {\tt dvi}ファイルをプレビューする}}\
59:     {\tt dvi}ファイルをプレビューしても、最初からきれいに
60:     画面上に出力されることはめった
61:     にありません。これは、コンパイルは
62:     通ったとしても、プログラムが正しく動くとはかぎらないと
63:     いうのと同じことです。この場合は、指定どおりに
64:     文書が作成されているかどうかをディスプレイで
65:     確認します。指定どおりになっていなければ、修正箇所を見つけて、
66:     気に入った配置になるまでエディタでの編集、
67:     \TeX での変換を繰り返します。
68:   {\bf {\item プリンタで印刷する}}\
69:     指定どおりの文書が作成できたら、実際にプリンタで
70:     印刷します。 \TeX の出力は、単にプリンタで印刷
71:     だけでなく、高精度な写植機から印画紙として出力したり、
72:     製版フィルムに出力したりして、
73:     印刷所で印刷することもできます\footnote{もっとも、何百万円もする
74:     印刷コストを個人で負担できるのかどうかは知りませんが。}。
75:   \end{enumerate}

```


このように T_EX での文書処理は、日本語ワードプロセッサでの処理に比べると、手軽とはとてもいえない複雑な工程を経て印刷結果を得るようになっていきます。T_EX の知識がない場合には、List 1-2 のような実際の T_EX のソースを見ても、その処理結果を想像することもできないでしょう。List 1-2 は、本当にこの本の原稿の抜粋なのです。

できあがった文書の美しさは T_EX ならではのものです。さらに T_EX では日本語ワードプロセッサでは“とてつもなく面倒な変更”も簡単に行うことができます。

たとえば、日本語ワードプロセッサで作成した A4 判の大きさの原稿を、配布の都合上、B5 判に直したいといった変更も簡単に行うことができます。このような変更は、日本語ワードプロセッサで行えば、単なる“変更”といったレベルではなく、“作り直し”のレベルになってしまいます。しかし、T_EX を使えば、このような変更は簡単にできてしまいます⁶⁾。

なぜなら、T_EX は日本語ワードプロセッサのように文書の文字の並び方を操作するのではなく、文書の論理構造を操作するものだからです。日本語ワードプロセッサでは、「章」とか「節」といった論理的な概念を認識しません。画面上に指示された位置に文字を並べて、それをそのまま印刷するのが日本語ワードプロセッサの役目です。A4 判を B5 判に見かけ上変換して印刷することができる高機能な日本語ワードプロセッサもあるかもしれませんが、レイアウトは多少なりとも人間が手で変換する必要があるでしょう。

T_EX では「章」とか「節」といった概念を T_EX のコマンドを通して理解します。A4 判から B5 判への変更で今まで 1 ページに収まっていた文章が次のページに移動する場合にも、つねに「章」や「節」を意識してレイアウトを行います。たとえば「章」の見出しは必ず奇数ページから始めるという規則を T_EX に指示しておきます。A4 判ではうまく奇数ページに収まっていた「章」の最初のページの原稿が B5 判への変更で偶数ページに移ってしまった場合は、T_EX は自動的に空白の部分を作成して「章」の頭が奇数ページになるようにページの割り付けを変更します。この場合にはそれまで参照していたページなども移動しますが、T_EX のコマンドで適切に参照の指示がされていれば、すべて自動で参照ページの振りなおしも行ってくれます。

先ほども説明したように、本書は T_EX を使って作成されていますが、原稿を作成するときにこのページに印刷されている文字の位置をいちいち設定したわけではありません。行頭・行末の禁則処理についても意識して原稿を作成したわけではありません。意識したのは、段落と T_EX のコマンドだけです。文字の配置やページ割り付け等は、全部 T_EX が自動的に行ってくれたのです。この本の大きさも最初に T_EX に教えてやるだけで、ページ番号をつけることなども自動的に行ってくれます。これが T_EX の最大の利点です。A4 判の大きさから B5 判の大きさに変更するには、最初に T_EX に指示している紙の大きさを変更するだけでいいのです。サイズ変更によるページの移動や、引用箇所の変換・変更などは全部 T_EX が処理してくれます。T_EX での処理手順を一度理解してしまえば、T_EX が非常に便利で美しい仕上がりをもたらしてくれるでしょう。

この論理構造を理解しながら組版をする T_EX の優れた特徴は、ある場合にお

6) とはいっても、実際には、それなりの知識が必要ですが。

いてはとんでもない欠陥に見えることがあります。それは、T_EX が行うレイアウトを強制的に変更したい場合に顕著に現れます。T_EX にはある程度レイアウトを指定するコマンドはありますが、日本語ワードプロセッサのように「機械が許す範囲で思うがままに配置する」ということはほとんど不可能です。道具には「得手、不得手」があるものです。一般的に「論理的な文章」や「定型文書」を記述する場合には T_EX は非常に便利ですが、一時的にしか使わない短い文書を作成する場合にはたいいてい日本語ワードプロセッサのほうが簡単です。T_EX だけがすべてではないことも事実として認識しておきましょう。

最初に、日本語ワードプロセッサはインタプリタ、T_EX はコンパイラに相当すると述べました。実際の作業も、T_EX で行う場合は、コンパイラを使うときのようにエディタと T_EX での処理を往復することになります。この作業を円滑に行うには、まず、X680x0 のエディタに慣れる、また COMMAND.X のような、コマンド処理中心のユーザインターフェースに慣れる必要があります。残念ながら、まだ X680x0 の T_EX での環境は“誰でも使える”というような環境までは進化していません。エディタで日本語テキストを自由に編集できるようにでなければ、T_EX を使うことはできません。この段階に至っていない方は、まず Human68k のマニュアルから読みはじめてください。

1.2.3 T_EX の種類と実行ファイル

T_EX の基本的な実行形式ファイルは `initex.x` と `virtex.x` です。`initex.x` は、通常の T_EX のソース処理には使わず⁷⁾、基本的なデータベースを作成するときに使用します。このデータベースには英単語のハイフネーション⁸⁾の知識や、出力時に使うフォントの種類、ページのレイアウトの方法などが T_EX の内部形式で収められます。このデータベースを作成するための T_EX の命令、「マクロ」を集めて記述したファイルを普通「マクロファイル」と呼びます。

`virtex.x` は、この `initex.x` が生成した基本的なデータベースを読み込んだあと、ユーザの T_EX ソースファイルを dvi ファイルに変換するプログラムです。`virtex.x` は、必ずひとつの基本データベースを必要とします。この基本データベースで T_EX の種類が決定されます。T_EX にはいくつか種類がありますが、次の 2 つが代表的なものです。

(1) plain T_EX

あらゆる種類の T_EX のうちで最も基本的な T_EX です。単純であるために細かい指示を T_EX に指定するには便利です。

(2) L^AT_EX

最もよく用いられている T_EX でしょう。「手紙」や「本」といった定型文書の処理を特に得意とする T_EX です。

このほかにも第 6 章「T_EXfamily」(p.229) で説明されているたくさんの種類の T_EX が存在します。しかし、実行ファイルがその種類に対応して複数存在す

7) といっても、`initex.x` で通常のソースファイルを `dvi` ファイルに変換できないわけではありません。

8) 英単語を“-”で分けて改行を行う場合の処理方法です。

るわけではありません。L^AT_EX と呼ばれる T_EX の処理を行いたい場合も、実行するのは `virtex.x` です。L^AT_EX 等の種類に対応して存在するのは先ほど書いた `initex.x` が生成するデータベースなのです。

T_EX にさまざまな指示をするためのマクロは、ユーザがソースファイル上で新しく定義して増やすことができます。ですが、この新しく追加したマクロは基本データベースには存在しません。ですから、毎回ユーザがマクロ定義を記述しなければいけません。また L^AT_EX なども、L^AT_EX のマクロファイルを `initex.x` で処理した基本データベースをもとにして動きますから、この元のマクロファイルが改変された場合にはデータベースも改変しないと、改変された L^AT_EX を使うことにはなりません。

ですから、新しい T_EX のマクロファイルを入手して T_EX の種類を増やしたり、新しい L^AT_EX を生成したい場合には、このデータベースの作成方法を知っておかなければいけません。このデータベースは、拡張子に `.fmt` が付加されるファイルなので「`fmt` ファイル」としばしば呼ばれます。

では、実際に `fmt` ファイルを作成してみましょう。`initex.x` に `fmt` ファイルを作成させるのは簡単です。実例として、L^AT_EX の `fmt` ファイルを作成してみましょう。

```
A> initex jlplain
This is pTeX, C Version 2.99 j1.7 p1.0.9F EW (INITEX)
(X:/XXXXX/jlplain.tex Preloading the plain format: codes,
....
....
....
*
```

`initex.x` は、`fmt` ファイルの作成が終わると、通常、最後に “*” を表示してコマンド待ちの状態になって停止します。この状態でキーボードから `\dump` とコマンドを入力します。`\dump` は `initex.x` だけに用意されたコマンドで、T_EX の内部情報を `fmt` ファイルに出力します。それでは `\dump` してみましょう。

```
*\dump
.....
.....
\font\tencirc=lcircle10
\font\tencircw=lcirclew10
23921 words of font info for 90 preloaded fonts
14 hyphenation exceptions
Hyphenation trie of length 5942 has 181 ops
No pages of output.
Transcript written on jlplain.log.
A>
```

9) 他のドライブやディレクトリ上にあるファイルを、シンボリックリンクという特別なファイル構造を通して、あたかもカレントディレクトリに実体があるかのようにアクセスできるものを指します。

10) この例では `jlplain.fmt` です。

11) UNIX では “&” は並列実行を指定します。UNIX はマルチユーザ・マルチタスクの OS なので同時に複数のプログラムを動かすことができます。

12) この場合は、`virtex.x` が存在しているディレクトリ名が `virtex.x` に付加されています。

これで L^AT_EX の `fmt` ファイルが出力されました。この `fmt` ファイルを `virtex.x` に読み込ませることではじめて、`virtex.x` は L^AT_EX として働くようになります。L^AT_EX 独自の実行ファイルが個別に存在しているわけではありません。もしあなたが使っている環境に `latex.x` というような名前の、個別の L^AT_EX の実行ファイルが存在するなら、それは第 1.2.4 項「より高度な環境構築」(p.14) で説明する `undump` でデータベースをすでに読み込んで実行形式に変換された `virtex.x` であるか、別のファイルネームで同じ実行ファイルを実行できるようにシンボリックリンク⁹⁾ された `virtex.x` なのです。

T_EX のデータベースを書き出した `fmt` ファイルを `virtex.x` に読み込ませるには、コマンドラインから `fmt` ファイルを指定して起動します。

```
A> virtex &jlplain
This is pTeX, C Version 2.99 j1.7 p1.0.9F EW (no format preloaded)
...
```

このように、コマンドラインから `fmt` ファイルの拡張子 `.fmt` を削除したファイル名¹⁰⁾ の前に “&” を付加して指定します。ちなみに、“&” は UNIX ではシェルが処理する特殊文字¹¹⁾ なので、そのまま “&” と記述することはできません。UNIX で T_EX を使う機会がある人は注意してください。なお、Human68k の `COMMAND.X` では、“&” は普通の文字として扱われますので、そのまま `virtex.x` に渡すことができます。

この指定された `fmt` ファイルは、カレントディレクトリか、第 1.2.6 項「T_EX が参照する環境変数」(p.16) で説明する環境変数 `TEXTFORMATS` が示すディレクトリに存在していなければなりません。`virtex.x` は、この `fmt` ファイルの指定がない場合には、まず、起動された実行ファイルのフルパスネーム¹²⁾ に `.fmt` を付加したファイルを探します。

たとえば、`A:\BIN\virtex.x` がフルパスネームだとすれば、`fmt` ファイルの指定がない場合、`A:\BIN\virtex.x.fmt` を検索します。これが見つからなければ、`virtex.x` は `plain.fmt` を読もうとします。`plain.fmt` は plain T_EX という最も基本的な T_EX のデータベースです。これが見つからない場合は、`virtex.x` はエラーを出力して終了します。

Human68k に UNIX のようなリンクファイルを実現するフリーソフトウェアを使えば、実行された `virtex.x` のフルパスに `.fmt` を付加する、このファイル検索規則を利用して、あたかも L^AT_EX の実行ファイルが存在するかのように見せることも可能です。つまり、たとえば、`virtex.x` の別名として `latex.x` をつくっておきます (シンボリックリンク)。この `latex.x` の実体は `virtex.x` ですが、起動時には `latex.x` という名前で起動されます。このため、まず `latex.x.fmt` を検索しますので、これをあらかじめ作成しておき、`latex.x` と同じディレクトリに入れておけば、コマンドラインから `fmt` ファイルを指定する必要がなくなります。実際の設定方法は次の第 1.2.4 項「より高度な環境構築」で説明しますので、ここでは省略します。

◎ `lndrv.x`

本文で登場している `lndrv.x` について簡単に説明しておきましょう。

`lndrv.x` は UNIX でのシンボリックリンクを Human68k 上で実現する常駐プログラムです。

シンボリックリンクとは、あるファイルに、別の新しいファイル名をつけるようなものです。ただし、リネームの場合と違い、元のファイルもそのまま残ります。そして、元のファイル名でも、新しくつけられたファイル名でも、同じファイル（これを「実体」といいます）がアクセスされます。もちろん、元のファイルとは違うディレクトリやドライブに、このシンボリックリンクのファイルをつくることもできます。

これは、一見、元のファイルを別のファイル名でコピーしたかのようにも見えますが、コピーと違って、元のファイルに修正をほどこしたため、コピーしてつくったファイルと食い違いが起こってしまうということはありません。シンボリックリンクファイルでつくった別のファイル名は、ディレクトリをとってみれば、そこに、そのファイル名のファイルが存在するように見えます（シンボリックリンクに対応していない `COMMAND.X` では動作がおかしくなります）が、あくまで別の名前がついているだけで、実体は元のファイルなのです。

本文中での例では、`virtex.x` を `latex.x` にリンクして使っています。この場合、`latex.x` を起動したときに `lndrv.x` が“すり替え”を行って、実体である `virtex.x` を `latex.x` のファイル名で起動したかのように動作します。

`lndrv.x` は、“すり替え”機能を実現するために、Human68k で通常は使われていないファイルの属性情報を使用しています。

ファイルへのアクセスに“`lndrv`”が割り込んで、この属性情報を調べ、もしシンボリックリンクファイルだったら、リンク先の実体のファイルをアクセスするように“すり替え”るのです。

`lndrv.x` では UNIX 上のシンボリックリンクと同様、ファイルに対してだけではなく、ディレクトリに対しても“すり替え”処理を行うことができます。

つまり、あるディレクトリにアクセスすると、別のディレクトリにアクセスしたのと同じことになります。ドライブレベルであれば Human68k の `SUBST.X` も同じような機能を実現していますが、`lndrv.x` ではディレクトリレベルでもこれを行うことができるわけです。この機能を利用すれば、たとえば、「 \TeX の実行ファイルがあるディレクトリにパスを通したいけれど、パスが長すぎて（Human68k のパス指定は 1 行 255 文字以内という制限のために）指定しきれない」場合などにも、短いパスにディレクトリごとリンクして対処することができます。

このように、`lndrv.x` は非常に便利ですが、ファイルの属性情報を勝手に（あくまで、Human68k のお約束になってないというだけのことで）に拡張して使っているため、`lndrv` に対応しない一部のプログラムでは、リンクが壊れて思いもかけない動作をする場合がありますから注意してください。

13) もちろん、その内容が多
くの人役に立つような
ものなら、進んで公開す
べきでしょう。

14) Human68k は 21 文字
のファイルネームが使える
ことになっていますが、
実際にはファイル名には
8 文字 + 拡張子 3 文字し
か認識していません。

15) lndrv.x は Human68k
上でシンボリックリンク
を実現するためのドライ
バで、作者は沖勝氏です。
添付ディスク 8 に収録さ
れています。

16) ln.x は実際にファイルを
シンボリックリンクする
ためのユーティリティで
す。既述の lndrv.x 用
に、いくつかの同名かつ
同機能のプログラムが、
異なる作者の手によって
発表されています。本書
の添付ディスク 8 には、
lndrv.x の作者である沖
勝氏の手によって、“コ
ンパイルされた”（「移
植された」ではないそう
です。アーカイブ内のド
キュメント参照）GNU
の ln.x が収録されてい
ます。

17) 11 ページ参照。

これで T_EX の種類を決定する fmt ファイルについては理解していただけたか
と思います。L^AT_EX にユーザ独自の T_EX の処理を加えた fmt ファイルを作成し
ておけば、毎回、\input コマンドで読ませる手間を減らすことも可能です。し
かし、このような fmt ファイルを使った T_EX は、他の場所で動いている T_EX と
は当然互換性がないので、そのような独自の fmt ファイルでしか処理できないよ
うな T_EX のソースをプライベートではない環境に流通させるのは問題があると
いえるでしょう¹³⁾。

1.2.4 より高度な環境構築

第 1.2.3 項「T_EX の種類と実行ファイル」で、T_EX の種類は virtex.x に読ま
せる fmt ファイルの種類で決定されることを説明しました。通常は、いくつか
の種類の T_EX の fmt ファイルを用意しておき、用途に応じて使い分けるのでは
ないかと思います。たとえば、List 1-3 のようなバッチファイルを作成しておく
のもひとつの手です。このバッチファイルは、ひとつの L^AT_EX ソースファイルを
L^AT_EX で処理するためのものです。また、いくつかの種類の T_EX を使う場合に
は、List 1-3 のようなバッチファイルを複数用意すればいいでしょう。ただし、
バッチファイルからバッチを呼び出すには工夫が必要になる等の不満が少々残る
かもしれません。

List 1-3 ● 実行バッチの例

```
1: rem LaTeX 実行バッチ
2: virtex &jlplain %1
```

これを解決する方法のひとつに、第 1.2.3 項「T_EX の種類と実行ファイル」で
説明した、virtex.x が自分の起動したパスを参照するという性質を利用します。
この性質と、Human68k のファイルネーム制限¹⁴⁾を解除するフリーソフトウェ
ア TwentyOne.x、および Human68k のファイルに UNIX のシンボリックリン
クの概念を導入する lndrv.x¹⁵⁾と ln.x¹⁶⁾を用いることで、ほぼ UNIX と同様な
T_EX の利用環境を実現することが可能です。L^AT_EX を作成する場合の具体的な手
順を書いてみます。

(1) L^AT_EX の fmt ファイルを作成する

virtex.x が存在するディレクトリで、initex.x を用いて L^AT_EX 用の fmt
ファイルを作成します¹⁷⁾。

(2) fmt ファイルをリネームする

fmt ファイルの名称を、自分が実行したい T_EX の名称にリネームします。こ
のとき、ファイル名の最後に拡張子.fmt を付加します。たとえば、latex と
いうコマンド名で起動したい場合には latex.x.fmt とリネームします。

(3) シンボリックリンクを張る

latex.x が virtex.x を示すようにシンボリックリンクします。これで、

latex とタイプすると、latex.x という名前で実体である virtex.x が起動されます。この結果、latex.x.fmt が読み込まれて L^AT_EX が実行されます。

これで latex とタイプすれば、L^AT_EX が起動するようになります。UNIX でもこのような使い方をするのが普通ですが、より日常的に L^AT_EX を使う場合や、L^AT_EX を起動したときに表示されるメッセージ、

```
This is pTeX, C Version 2.99 j1.7 p1.0.9F EW (no format preloaded)
LaTeX Version 2.09 <4 Aug 1988>
```

```
.....
.....
```

のなかの (no format preloaded) と出力されるのが気に入らない一部のユーザは、undump¹⁸⁾ という方法を用います¹⁹⁾。

これは、これまで説明したバッチやシンボリックリンクといった方法に比べると、荒っぽい、機種依存性のある方法です。どういうことをするかといえば、fmt ファイルを読み込んだ状態の T_EX をそのままディスク上に出力して、それを再び実行形式のファイルにするといった方法をとっています。undump するには、この本に添付されたディスクのほかに GNU make や他のツール類が必要です。また、トラブルが起こった場合には自分で対処できるだけの深い知識も必要になります。

一方、X680x0 でも最近ではハードディスクがある程度高速になっていますし、現在ではこの方法はそれほど有効なものとは思えません。やはり、シンボリックリンクによる起動が最も簡便で、ディスクの消費量も少ない方法だと思われます。

18) undump された T_EX では undump された日時などの情報が出力されます。

19) ただ単に表示がカッコイイという理由くらいしかなさそうですが。

1.2.5 T_EX のコマンドライン書式

T_EX の実行形式ファイルである initex.x および virtex.x による処理では、コマンドラインから動作を変更するようなオプションを指定することはありません。コマンドラインで指示されるのは、T_EX のコマンドかソースファイルの指定だけです。

○ initex.x のコマンドラインの書式

```
initex <ソースファイル>
initex <コマンド>
```

<ソースファイル> は T_EX のソースファイル名で、拡張子 .tex は省略可能です。<コマンド> は T_EX のコマンドです。

```
A> initex mylplain
This is pTeX, C Version 2.99 j1.7 p1.0.9F EW (INITEX)
....
....
```

○ virtex.x のコマンドラインの書式

```
virtex [<fmt ファイル>] <ソースファイル>
virtex [<fmt ファイル>] <コマンド>
```

[<fmt ファイル>] は省略可能です。<ソースファイル> は TeX のソースファイル名で、拡張子 .tex は省略可能です。<コマンド> は TeX のコマンドです。<fmt ファイル> については第 1.2.3 項「TeX の種類と実行ファイル」(p.10) を参照してください。

```
A> virtex &mylplain mysrc.tex
This is pTeX, C Version 2.99 j1.7 p1.0.9F EW (INITEX)
....
....
```

このようにコマンドライン上から指定するような TeX のオプション類はありません。TeX では、コマンドラインよりも第 1.2.6 項「TeX が参照する環境変数」で説明する環境変数や、第 1.2.3 項「TeX の種類と実行ファイル」で説明した実行ファイルの実際の名称や起動環境のほうが、その動作に大きな影響を与えます。

1.2.6 TeX が参照する環境変数

TeX の実行プログラムは、環境変数を参照して、さまざまなファイルを読み込んでから動作します。この環境変数にはファイルを検索するディレクトリを指定します。ディレクトリの指定方法は、Human68k でのフルパスを記述します。また、エラーがあった場合に起動するエディタを指定する TEXEDIT 環境変数を除いて、複数のパスを指定することができます。複数のパスを指定する場合は“;” (セミコロン) で区切ります。元来、UNIX ではパスの区切りに“:” (コロン) を使いますが、Human68k ではドライブ名の区切りに“:” が使われているので、かわりに“;” を使っているのです。なお、Human68k の仕様上、パスの区切りに“\” (バックスラッシュ) と“/” (スラッシュ) を使うことができます²⁰⁾ が、TeX では“\” (¥) という記号に特殊な働きが割り当てられている都合上、パスの区切りに“/”を使用しなければならないことがほとんどです。また、ディレクトリ名に全角文字を使った場合には、TeX の動作は保証されませんので注意してください。List 1-4 に環境変数の設定例を示します。

20) ただし、/ をパスの区切りとして認識する機能は不完全です。

List 1-4 • T_EX 環境変数の設定例

```

1: REM pool ファイルは C:\TEX\TEXPOOL にある
2: SET TEXPOOL=C:\TEX\TEXPOOL
3: REM TeX のマクロファイルは C:\TEX\TEXMAC および C:\TEX\MYMACRO にある
4: SET TEXINPUTS=C:\TEX\TEXMAC;C:\TEX\MYMACRO
5: REM fmt ファイルは C:\TET にある
6: SET TEXFORMATS=C:\TEX\TEXFMT;C:\TEX\MYFORMAT
7: REM tfm ファイルは C:\TEX\TEFonts にある
8: SET TEXFONTS=C:\TEX\TEFonts
9: REM エディタは em, 引き数はファイルネームと行番号
10: SET TEXEDIT=em %s -g%d

```

T_EX は、以下のような環境変数を参照します。これらの環境変数が設定されていなかった場合には、ファイルの検索はすべてカレントディレクトリが対象になります。

(1) TEXFORMATS

第 1.2.3 項「T_EX の種類と実行ファイル」で説明した `fmt` ファイルがカレントディレクトリで発見できなかった場合に検索するディレクトリを指定します。

(2) TEXINPUTS

コマンドラインや `\input` コマンドで指定されたファイルを検索するディレクトリを指定します。

(3) TEXFONTS

フォントの情報を収めている `tfm` ファイル群を検索するディレクトリを指定します。

(4) TEXFMT

コマンドラインから “&” で指定された `fmt` ファイルを検索するディレクトリを指定します。

(5) TEXPOOL

`ptex.pool`²¹⁾を検索するディレクトリを指定します。この指定を必要とするのは `initex.x` のみで、頻繁に `fmt` ファイルをつくる方以外は必要とときだけ指定しても実用上は問題ありません。

(6) TEXEDIT

T_EX は、処理するソースにエラー箇所を発見すると、作成者の指示をおおぎます。そのとき、エディタ起動を指示して T_EX を終了させることができます。この環境変数で使用するエディタを指定します。その書式は、

エディタ名 %s 行指定オプション %d

です。%s の部分にはエラーを起こしたソースファイル名、%d の部分にはエラーが起こった行番号が入ります。この 2 つの引数を「エディタ名」で指定したエディタに渡します。この環境変数は、T_EX 内部で C 言語の `sprintf` 関数にそのまま渡されます。誤った指定を行った場合には、バスエラーやアドレスエラーを引き起こすので注意してください。T_EX では `sprintf` 関数

21) T_EX を作成する際に Pascal から C への変換を行います。そのときに生成される内部情報ファイルです。

が環境変数で指定されたフォーマット文字で安全に実行できるかどうかをチェックしていません。

1.2.7 T_EX のための X680x0 のセットアップ

T_EX を動かすための X680x0 の動作環境について少し説明しておきます。T_EX を動かす場合には、T_EX 自体には直接的な原因のないエラーが発生することがあります。そのようなエラーは、たいてい X680x0 の動作環境である Human68k のセットアップの不備が原因です。Human68k の不備で発生するエラーにはどのようなものがあるかを見てみましょう。

○ 存在しているファイルが見つからないエラー

このエラーは、CONFIG.SYS で指定する FILES の数が不足しているときに発生します。T_EX はたくさんのファイルを扱いますので、CONFIG.SYS での FILES は多めに設定²²⁾ しておいてください。

また、Human68k の Ver.2.00 では “/” をパスの区切りとして認識しませんので、T_EX を使うことはできません。Ver.2.01 以上の Human68k をお使いください。

○ ファイルが実行できません

フリーメモリが少ない場合に T_EX を実行しようとする、このエラーが発生します。特に Big T_EX (29ページのコラム参照) を動かす場合にはフリーメモリが 4M バイト程度必要になります。

○ UNIX ではエラーにならないファイルがエラーになる

Human68k は、MS-DOS のようにファイルネームに制限があります。X680x0 の T_EX は、Human68k で許されている 21 文字までのファイルネームすべてを区別でき、かつ通常はひとつしか使えない “.” (ピリオド) が複数使える環境で動作するようにつくられています。つまり、TwentyOne.x の常駐を前提にした環境ということです²³⁾。

しかし、UNIX では、この TwentyOne.x によって拡張された Human68k の制限を越えてしまうようなファイル名がしばしば使われます。このため、UNIX では処理できるファイルが Human68k では処理できないことがあります。

このほか、X680x0 で T_EX を使う場合に注意しておいたほうがよい点を挙げておきましょう。X680x0 では、メモリは金銭的に許されるかぎり増設しておいたほうがよりよい T_EX 環境を構築できます。また、T_EX のソースファイルを編集する場合には X680x0 では “≡” ではなく、“\” を使うように X680x0 本体を設定しておきましょう。設定は、Human68k のコマンドである SWITCH.X で行います。“\” を使ったほうが、ソースファイルの見かけがきれいに见えます。

22) できれば最大数の 93 を指定してください。同じ理由から、TwentyOne.x の “-B” オプションの引数も多めにしておいたほうがよいでしょう。

23) TwentyOne.x は、ファイル名に複数のピリオドを使うことができます。

1.2.8 T_EX が使うファイルの拡張子

T_EX は、最終的な印刷結果を得るまでにたくさんのファイルを参照します。これらのファイルには、普通、そのファイルの役割を表す拡張子が使われます。今まで拡張子については適宜説明を加えてきましたが、ここで整理をしておきましょう。T_EX 全般で用いられるファイルを一括して説明していますので、この節以外で該当ファイルの説明がある場合は、参照すべきページを付記しておきます。

なお、拡張子が 4 文字以上のファイルは、TwentyOne.x を常駐させていない、オリジナルのままの Human68k では使用することができませんので、十分に注意してください。

(1) .pool 拡張子のファイル

T_EX の実行プログラムは、エラーメッセージやその他の詳細情報を出力するために pool ファイルを必要とします。pool ファイルの内容を手で勝手に変更してはいけません。

(2) .tex 拡張子のファイル

T_EX のソースファイルや、T_EX のマクロの集合ファイルは、通常、.tex という拡張子を持っています。第 1.2.2 項「T_EX の処理の流れ」(p.6)を参照してください。

(3) .sty 拡張子のファイル

sty は、L^AT_EX でよく用いられる拡張子で、作成される文書のスタイル²⁴⁾を定義する T_EX のマクロが記述されています。「style_option」(p.105)および「stylefile」(p.108)を参照してください。

24) 文書の大きさ (たとえば A4 サイズ) や書体、文章のレイアウトのしかた等のこと。

(4) .fmt 拡張子のファイル

T_EX の種類を決定する内部情報ファイルです。第 1.2.3 項「T_EX の種類と実行ファイル」(p.10)を参照してください。

(5) .tfm 拡張子のファイル

T_EX がフォントの情報を得るためのファイルです。METAFONT が作成しますが、METAFONT では日本語のフォントは作成できません。詳しくは「METAFONT の処理の流れ」(p.215)を参照してください。

(6) .mf 拡張子のファイル

METAFONT がフォントを作成するために使う、フォントの作成方法を記述したソースファイルです。詳しくは「METAFONT の重要な注意事項」(p.214)を参照してください。

(7) .base 拡張子のファイル

METAFONT が使用するファイルで、T_EX の fmt ファイルに相当するファイルです。「METAFONT のセットアップ」(p.218)を参照してください。base ファイルは、TwentyOne.x を常駐させていないオリジナルのままの Human68k では使えない 4 文字以上の拡張子を持っています。十分注意してください。

(8) .numgf 拡張子のファイル

METAFONT が出力するフォントのイメージファイルです。X680x0 のプレ

ビューア `preview.x` やプリンタドライバ `print.x` では、通常は `numgf` ファイルを圧縮した `.pk` という拡張子を持ったファイルを使います。詳しくは「METAFONT の処理の流れ」(p.215)を参照してください。

(9) `.pk` 拡張子のファイル

フォントイメージを圧縮した形式のファイルです。プレビューアやプリンタドライバは、`pk` ファイルからフォントのイメージを得ます。詳しくは「METAFONT の処理の流れ」(p.215)を参照してください。

25) `texinfo` 形式は、ひとつのドキュメントソースを、Emacs のドキュメント処理システム “info” でも TeX でも閲覧することができるようにしてあるドキュメント形態です。

このほかにも、GNU で使われている `texinfo` 形式のファイル²⁵⁾に用いられる `.texinfo` といった拡張子などがあります。また、TeX はソースファイルの拡張子が `.tex` であることを要求していません。ですから、普通のテキストファイルの拡張子 `.doc` の中身が、実は TeX 形式のドキュメントであることもあります。

Install

本書で説明する $\text{T}_{\text{E}}\text{X}$ のインストール方法は、(1) 同一パーティションにインストールする、(2) 3 つのディレクトリに分割してインストールする、(3) より高度なカスタムインストールをする、という 3 つに分類することができます。本章では、 $\text{T}_{\text{E}}\text{X}$ 初心者向けのインストール方法として、本書の添付ディスクに付属しているインストーラによって可能な、(1) と (2) を中心にして説明を行います。読者の環境にあわせた、より高度なカスタムインストールについては第 5 章「Customize」で説明していますので、そちらを参照してください。

2.1最低限必要な環境

TeX システムを動かすためには、X680x0 本体のほかにハードディスク、プリンタ、メモリ等の環境を整備する必要があります。ここでは、添付ディスク付属のインストーラ `install.x` でインストールするために必要な環境について説明します。

(1) 必要なハードディスク容量

1)MO にインストールすることも可能です。

本章でのインストールの説明は、基本的に ハードディスク (以下、HD と略す) にインストールすることを前提にして進めます¹⁾。添付ディスクに収められているインストーラ `install.x` を使ってインストールした場合、最低限必要なディスク容量は合計で 10M バイトです。これは、フロッピーディスク (以下、FD と略す) にインストールして使うことはとうてい不可能な容量です。ですから、以下の説明は HD にインストールすることを前提に進めます。HD のついたマシンをお持ちでない方は、添付ディスク付属のインストーラでのインストールは不可能です。

TeX システムの動作環境を把握して必要ファイルのみを FD にインストールして実行すれば FD 上での運用も不可能ではありません。しかし、その場合は FD を何度も交換しながら使わなければなりませんし、ファイルの読み込みにも時間がかかり、とても実用的に使える環境とはいええないでしょう。最近では HD も安くなっていますので TeX システムを運用する場合は HD を購入してください。HD の空き容量が 8.5M バイト以上あれば、なんの問題もなくインストールできます。

また、HD を分割して使っている場合は、空き容量が 4.6M バイト以上の領域、2.1M バイト以上の領域、1.6M バイト以上の領域がそれぞれあれば、インストーラにより分割インストールすることが可能です。どちらの条件も満たせない方は、本書添付のインストーラでのインストールはできません。TeX システムのディレクトリ構成を詳しく把握されている方なら自力でインストールすることも可能でしょうが、そうでない方は HD 上にインストールが可能な空き容量をとってください。

なお、内蔵 ROM 以外の日本語フォントを使用する場合には、別に日本語フォントのインストール用のディスク容量が必要になります。たとえば、日本語フォントとして本書添付のフリーのビットマップフォントを使用される方は、全体の空き容量が 10.5M バイト以上、あるいは、分割先領域としてさらに 2M バイト必要になります。

(2) 必要メモリ

T_EX システムは、大量のメモリを必要とします。特に印刷時にはさまざまなフォントをメモリ上に読み込むので大量のメモリが必要になります。300dpi 以上のプリンタを使用される方なら、4M バイト以上のメモリを用意してください。環境によっては 4M バイトのメモリがあってもメモリ不足になる可能性があります。メモリ不足でエラーになったら float?.x 以外のデバイスドライバをすべて外し²⁾、常駐ソフトはいっさい使用しない等の工夫をしてフリーエリアを増やしてください。フリーエリアが 3M バイト以上あれば、問題なく印刷できます。

2)float?.x は浮動小数点演算のために必要です。

(3) 必要プリンタ

T_EX はプリンタで紙に打ち出してこそ、その美しさを発揮することができますし、内容を確認することもできます。ですから、プレビューアを使ってディスプレイ上で表示するだけでよいという方以外は、プリンタを用意する必要があります。対応プリンタは以下のリストのとおりです。

X680x0 版の T_EX は、現在市販されている多くのプリンタに対応しています。リストのなかになくても同一系列のプリンタは使用できます。たとえば、シャープの CZ-8PC5 は CZ-8PC4 の上位コンパチなので T_EX での使用が可能です。現在市販されている一般的なパソコン用プリンタは、たいていリストのプリンタのいずれかにあたるはずです。プリンタ付属のマニュアル等を参考にして、あなたのプリンタがどれに相当するのかを調べてください。もし、残念ながら対応プリンタがリストにない場合には、そのプリンタ用のコンフィギュレーションファイルを作成しなければなりません。第 5.3 節「print.cfg の作り方」にキヤノンの BJ-10 プリンタを例にしてコンフィギュレーションファイルの作成方法を説明してありますので、そちらを参考にしてください。

メーカー	名称	dpi 値	メーカー	名称	dpi 値
エプソン	AP-550	180	シャープ	CZ-8PC3	180
	AP-850	360		CZ-8PC4	360
	AP-900	360		CZ-8PG1	180
	HG800	180		CZ-8PK7	180
	HG2500	180		CZ-8PK8	180
	LP-3000	300		IO-735X	180
	MJ-500	360	富士通	FM-PR451	180
	VP800	180	YHP	HP-DJ505J	300
	VP950	180		HP-LaserJet3	300
	VP2000	180		HP-LaserJet4	600
キヤノン	BJ-10v	360	NEC	PC-PR101	360
	BJ-130J	360		PC-PR406H	180
	BJ-330J	360		PC-PR2000/14	400
	LaserShot	180		NM9400S	180
	LaserShot	240	スター精密	TR24CL	180
ブラザー	M-1024	180			

(4) 日本語フォント

T_EX で美しい日本語表示を行うためには日本語のフォントを用意する必要があります。X680x0 シリーズ用の T_EX は以下のフォントに対応しています。

○ Z's STAFF および Z's WORD JG のアウトラインフォント

ツアイトの Z's STAFF に、そして Z's WORD JG の Ver.1 と Ver.2 に、それぞれ付属するアウトラインフォントです。アウトラインフォントですので、任意のサイズに拡大してもきれいな出力が可能です。

○ 書体倶楽部のアウトラインフォント

Z's STAFF および Z's WORD JG 用として発売されているフォント集です。基本的に Z's WORD 付属のフォントと同じものです。

○ Z's WORD JG Ver.3 のベジェフォント

Z's WORD JG Ver.3 用のフォントです。これもアウトラインフォントですので、任意のサイズに拡大してもきれいに出力することができます。

○ アスキーの パーソナル日本語 T_EX フォントライブラリ

アスキーが PC-9801 シリーズ用の『パーソナル日本語 T_EX』のために発売している JXL4 形式の日本語フォントです。フォントを拡大、縮小しても大変美しい書体で出力できますが、360dpi プリント用のフォントしか用意されていません。また、アウトラインフォントではありませんので L^AT_EX で定められているコントロール・シーケンスに対応した文字サイズ³⁾以外は出力できません。

○ 本書添付のフリーのビットマップフォント

フリーデータとして公開されているビットマップの日本語フォントです。基本サイズ (24, 32, 48 ドットのいずれか) はきれいに出力されますが、基本サイズ以外を用意されていないため、拡大、縮小にはあまり向いていません。

○ 本体内蔵 ROM フォント

本体内に内蔵されている漢字 ROM のフォントを使用します。ディスク容量に余裕がない方はこれを選ぶしかありませんが、出力結果は決してきれいなものとはいえません。T_EX を本格的に活用するつもりなら、他の日本語フォントを使用すべきでしょう。

3) コントロール・シーケンス `\tiny`, `\scriptsize`, `\footnotesize`, `\small`, `\normalsize`, `\large`, `\Large`, `\LARGE`, `\huge`, `\Huge` を使用して出力される文字の大きさだけでしか利用できないということです。各コントロール・シーケンスによって変化する文字の大きさについては、『L^AT_EX における文字の大きさ』(p.138) および『Vol.2 — Reference 編』の第 4.2 節「ポイント別文字サイズ比較」(p.140) を参照してください。

2.2インストーラによるインストール

本節では、添付ディスクに用意されているインストーラを用いてインストールする方法について説明します。インストールする前にあなたの環境が前節で説明した環境・条件を満たしているかどうかを確認してください。前述のとおり、インストール先の空き容量によって一括インストールと分割インストールを選ぶことができます。あなたの環境にあわせてどちらかの方法でインストールしてください。なお、本書では一括インストールによって作成される環境を標準環境としていますので、ディスク容量に余裕のある方はできるかぎり一括インストールすることをお勧めします。

2.2.1 インストールディスクの説明

本書に添付されているディスクは以下のような構成になっています。

○ DISK 1

<code>install.x</code>	: インストーラ本体です。
<code>TeXmain.lzh</code>	: $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ を動作させるために必要なファイル群です。
<code>TwOnes.Lzh</code>	: <code>TwentyOne.x</code> の Human68k Ver.2.02 用 (<code>TwOne202.x</code>), Ver.2.03 用 (<code>TwOne203.x</code>), Ver.3.0x 用 (<code>TwOne30x.x</code>) です。

○ DISK 2

<code>driver.lzh</code>	: フォントマネージャ、プレビューア等の実行ファイルです。
<code>smtexbin.lzh</code>	: X680x0 において標準の $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ 本体および <code>fnt</code> ファイルです。
<code>mf.lzh</code>	: METAFONT を作成する場合に必要なファイルのすべてがまとめられています。

○ DISK 3

<code>bigtexbin.lzh</code>	: Big $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ 本体および <code>fnt</code> ファイルです。
<code>pk360.lzh</code>	: プリンタ用の 360dpi <code>pk</code> フォントです。

○ DISK 4

pk180.lzh : プリンタ用の 180dpi pk フォントです。
 pk400.lzh : プリンタ用の 400dpi pk フォントです。

○ DISK 5

pk118.lzh : 画面表示用の 118dpi pk フォントです。
 pk300.lzh : プリンタ用の 300dpi pk フォントです。

○ DISK 6

pk240.lzh : プリンタ用の 240dpi pk フォントです。
 fr_font1.lzh : フリーの 24 ドット、および 32 ドットのビットマップ日本語明朝体フォントです。日本 PC ユーザーズ連盟が配布している、フリーの科学技術 DTP 数式 CAD ワープロ『LABO System 123』で使用するフォントをコンバートしたもので、NIFTY-Serve の FLABO に KNJ24.LZH および KNJ32.LZH のファイル名で登録されています¹⁾。本書の添付ディスクに付属のインストーラでは、プリンタの解像度にあわせてインストールするビットマップフォントを自動で選択します。このアーカイブに含まれる 24 ドットフォントは使用するプリンタの解像度が 180dpi のときに、32 ドットフォントは使用するプリンタの解像度が 240dpi および 300dpi のときに、それぞれ選択されます。なお、32 ドットフォントは 180dpi, 240dpi, 300dpi, 360dpi, 400dpi 以外の解像度を持つプリンタを使用する場合にも選択されます。

1) 便宜のため、フォントファイル名を mincho24.bm1 および mincho32.bm1 に変更しています。

○ DISK 7

YAFF48.lzh : フリーの 48 (実質は 46) ドットのビットマップ日本語明朝体フォントです。『LABO System 123』の 32 ドットフォントをもとに、融氏によって作成されました。
 このフォントには第二水準の漢字データが用意されていませんので、本書の添付ディスクに付属するインストーラでこのフォントを選択した場合、第二水準の漢字には ROM フォントが使用されることになります。
 なお、アーカイブ内には、現在フォントファイルに含まれているひらがなとは異なる書体のひらがなデータ、このひらがなデータに変更するツール (48 ドット専用, X680x0 および PC-9800 用)、実

質デザインサイズである 46 ドットにコンバートするツール (X680x0 用) などが含まれています。NIFTY-Serve の FSHARP および FLABO に、YAFF48.LZH のファイル名で登録されています。本書の添付ディスクに付属のインストーラでは、プリンタの解像度にあわせてインストールするビットマップフォントを自動的に選択します。このアーカイブに含まれる 48 ドットフォントは使用するプリンタの解像度が 400dpi のときに、48 ドットフォントから生成される²⁾ 46 ドットフォントは使用するプリンタの解像度が 360dpi のときに選択されます³⁾。

- KAWASAKI.LZH : 奈須野陽氏によって作成されたシェアウェアのゴシック体フォント「川崎 32」を、福田雅史 (べんぜん) 氏がコンバートしたものです⁴⁾。1 週間以上継続して使用する場合は使用料の 1500 円を作者に支払う義務があります。詳細はアーカイブ内のドキュメントに記されていますので、使用される方は必ず参照してください。
- LHA21246.x : 拡張子が.lzh の圧縮ファイルを解凍するためのプログラム LHA.x の自己解凍型アーカイブです。
- sample : 配下に第 6 章「TeXfamily」(p.229) で使用したサンプルソースを収めたディレクトリです。

○ DISK 8

- LATEXGUD.LZH : L^AT_EX の使い方に関する L^AT_EX のソースです。『やさしい L^AT_EX のはじめかた』(すずきひろのぶ, オーム社刊, 1991) のもとになりました。詳細は、第 6 章「TeXfamily」(p.233) に示します。
- fax2ttl.lzh : ファックスファイルをタイトル形式に変更するツールです。
- ttl2fax.lzh : タイトルファイルをファックス形式に変更するツールです。
- tex2ttl.lzh : 本書添付のプリンタドライバのダンプ出力をタイトル形式に変更するツールです。以上の 3 つのファイルについての詳細は、第 3.5 節「ファックスファイル変換」(p.68) に示します。
- gzp124x3.Lzh : GNU プロダクツの圧縮ツールです。lips3dvi2.09a.tar.Z の解凍に使用します。
- tar112b.lzh : GNU プロダクツのアーカイバです。lips3dvi2.09a.tar.Z の展開に使用します。

2) プリンタの解像度の入力を求められたときに 360dpi を選択すると、自動的に 48to46.x を使用して、46 ドットフォントを作成します。

3) 便宜のため、インストールの際にフォントファイル名を mincho48.bm1, mincho46.bm1 に変更します。

4) 便宜のため、インストールの際にフォントファイル名を gothic32.bm1 に変更します。

5) いうまでもありませんが、`TwentyOne.x` を +P オプションで常駐させておかなければ、ファイルにアクセスすらできません。

`lips3dvi2.09a.tar.Z` : LIPS3 や ESC/Page プリンタ用の出力ツールのソースです。

上の 2 つのアーカイブを `LHA.x` を使って解凍し、それぞれに含まれる実行ファイルを使用可能にした状態で、コマンドラインから以下のように入力して解凍してください⁵⁾。

```
A> gzip -dvc lips3dvi2.09a.tar.Z | tar
tvhk
```

`LIP3_68.LZH` : `lips3dvi2.09a.tar.Z` のパッケージを X680x0 に対応させるための差分です。

以上の 2 つのファイルについての詳細は、第 6 章「`TeXfamily`」(p.261) に示します。

`LNDRV126.LZH` : シンボリックリンクを使うためのツールです。

`ln.lzh` : 実際にシンボリックリンクをはるためのツールです。

以上の 2 つのファイルについての詳細は、第 1.2.4 項「より高度な環境構築」(p.14)、および p.13 のコラムで説明しました。

`MakeIndex.Lzh` : `LATEX` で索引をつくるためのツールです。

詳細は、第 6 章「`TeXfamily`」(p.263) で示します。

`PLAIN2.LZH` : プレインテキストから `LATEX` ソースを作成するためのツールです。

詳細は、第 6 章「`TeXfamily`」(p.264) で示します。

`pchX6_2full.Lzh` : 差分を適用するためのツールです。`VFLATEX` のマクロ作成に使用します。

`vfkit_last.lzh` : “`LATEX` の多書体化への試み” で使用する `tfm` ファイルを作成するためのツール集です。

`vflatex.lzh` : “`LATEX` の多書体化への試み” のためのキットです。

`VFjfm.Lzh` : `VFkit_last.lzh` 付属のシェルスクリプトによって作成した `tfm` ファイルです⁶⁾。

以上の 3 つのファイルについての詳細は、第 6 章「`TeXfamily`」(p.234) で示します。

`newfm_fix.lzh` : “`LATEX` の多書体化への試み” のためのフォントマネージャ用のコンフィギュレーションファイル集です。

詳細は、p.180, 185 のコラムで示します。

`ZsTrash.Lzh` : 融氏が作成した、ツァイトの『書体倶楽部』の毛筆体と教科書体についているゴミデータを削除するプログラム「点取虫」です。この処理をしてい

6) `VFLATEX` 用の `tfm` ファイルは非常に数が多く、今回添付した以外にも作成することができます。そのため、専用のディレクトリにまとめて管理したほうが、結果的にファイル検索の際の無駄が少なくなります。

◎ Big T_EX

本書の添付ディスクに付属するインストーラによって T_EX をインストールする場合、T_EX の実行ファイルとして、X680x0 において“標準”とされる T_EX、あるいは“Big T_EX”と呼ばれる特殊な T_EX、このどちらかを選択することができます。

Big T_EX とは、通常の T_EX に比べて、設定された作業用のメモリ上限を非常に大きく設定してある T_EX のことです。

普通 UNIX では、Big T_EX が「標準の T_EX」とされていますが、メモリ環境が貧弱な X680x0 の場合、UNIX でいう「標準の T_EX」がメモリ不足のために動かない場合がほとんどです。そのため、X680x0 版では Big T_EX が「オプション」の扱いになっています。

さて、この Big T_EX ですが、どのような場面で使用すればよいのでしょうか。

X680x0 で標準とされる T_EX では、設定された作業用のメモリ上限が小さく設定されているために複雑なマクロを使った（ある意味では、本当に行いたい）処理ができないことがあります。この場合には Big T_EX を使って作業を行うことになり

ますが、この Big T_EX は文字どおり本当に“Big”であるために、X68030 の標準メモリ環境である 4M バイトでも、多くのデバイスドライバを外さなければ動作しません。もっとはっきりいえば、コマンドラインから Human68k のコマンド MEMFREE を入力したとき、4M バイト以上の空きがなければ Big T_EX を使用することはできません。ですから、理想的な Big T_EX の環境として、ここでは「X68030 にメモリを 12M バイトフルに実装したもの」を推奨しておきます。

なお、Big T_EX と通常の T_EX では動作がまったく同じですので区別がつかない場合があります。

T_EX の実行ファイルを起動したときに、「This is P_TE_X, C Version 2.99...」と大文字の“P”で「P_TE_X」というメッセージを出力するのが Big T_EX で、「This is p_TE_X, C Version 2.99...」と小文字の“p”で「p_TE_X」というメッセージを出力するのが X680x0 標準の小さな T_EX です。

ない当該フォントを使用すると、デバイスドライバの実行時にはエラーが起こります。

ZeitUtil.lzh : ツァイトのベクトルフォントを加工するためのプログラム集です。ベクトルフォント同士をマージしたり、ベクトルフォントを構成する情報を細密化したりという処理が可能になります。

Storage.Lzh : フォントマネージャ用のフォントキャッシュフィルタのひとつです。絶大な偉力を発揮してくれます。

詳細は p.168 のコラムに示します。

以上、8 枚のディスクが揃っていることを確認しておいてください。

このうち、付属のインストーラによって行うインストールの対象になるのは、1～7 です。8 はある意味で“おまけディスク”で、基本的には第 6 章「T_EXfamily」(p.229)で紹介するマクロパッケージやアドオンツールの類を収めています。したがって、このディスクの中身は、読者の皆さんが必要に応じて、1～7 のインストールディスクによって構築した T_EX システムにインストールしなければなりま

せん。インストールの方法、および使用法については、各アーカイブ内のファイルに従ってください。また、簡単な使用法などについては、第6章「 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ family」(p.229)で触れているものもあります。

2.2.2 インストールの前に準備しておくこと

HD の容量、メモリなどの確認をしたら、インストールの前に以下の準備をしてください。

(1) LHA.x のインストールチェック

本書の添付ディスク 7 に収録されている LHA21246.x をハードディスクなど、空き容量のある作業領域へとコピーし、同ファイルを実行してください⁷⁾。このファイルは自己解凍型の圧縮ファイルですので、実行した結果、いくつかのファイルが解凍されるはずで、そのうちの実行ファイル LHA.x をパスの通ったディレクトリにムーブしてください。

なお、LHA.x は Human68k Ver.2 以上でなければ動作しません。Ver.1 しか持っていない人は、 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ の環境構築以前に、まず Human68k Ver.2 以上の環境構築から始めてください。

(2) 同時にオープンできるファイル数の設定のチェック

第 1.2.7 項「 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ のための X680x0 のセットアップ」(p.18)で述べたように、 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ がたくさんのファイルを扱う都合上、同時にオープンできるファイル数を多めに設定しておかなければなりません。CONFIG.SYS の FILES 行の設定を大きめに (可能なら最大値の 93 を) 指定してください⁸⁾。

```
files = 30
```

(3) 環境変数設定領域のチェック

$\mathrm{T}_{\mathrm{E}}\mathrm{X}$ システムはかなり多くの環境変数を参照します。そのため、環境変数の設定領域を大きめに確保しておかなければなりません。CONFIG.SYS の SHELL 行に記述された COMMAND.X⁹⁾の “/E” オプションの設定を (最低でも) 以下のようにしてください。

```
shell = COMMAND.X /P /E:50
```

(4) TwentyOne.x の常駐チェック

本書でインストールする $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ システムは、4 文字以上の拡張子を持つファイルを扱うため、それらのファイルの認識を可能にするフリーソフトの TwentyOne.x が常駐していなければなりません。

添付ディスク 1 に収録されている TwOnes.lzh には、Human68k の各バージョン用の TwentyOne.x が収録されています。まず、LHA.x を使用して TwOnes.lzh を展開し¹⁰⁾、あなたが使っている Human68k¹¹⁾が Ver.2.02 なら TwOne202.x を、Ver.2.03 なら TwOne203.x を、Ver.3.0x なら TwOne30x.x を、それぞれ選び、TwentyOne.x にリネームしたうえで、あなたの起動ディスクの適当なディレクトリにコピーしてください。

7) カレントディレクトリを当該作業領域に移したあと、コマンドラインから「LHA21246.x」と入力し、リターンキーを押してください。

8) 第6章「 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ family」(p.229)の筆者は、通常 30 を指定していますが、問題が発生したことは一度もないようです。もっとも、同著者の環境が特殊なのかもしれませんので明言は避けますが。

9) すでに Human68k から離脱しつつある人々がいる今、COMMAND.X から離れているユーザははるかに多いと思いますが、ここでは「初心」に帰ってください。

10) TwOnes.lzh を空き容量のある作業領域にコピーし、そのディレクトリにカレントを移したあと、コマンドラインから「lha x TwOnes」と入力し、リターンキーを押してください。

11) Human68k のバージョンチェックは、Human68k のコマンド VER で行うことができます。

次に、CONFIG.SYS に、

```
device = TwentyOne.x +P      .
```

の 1 行を追加します。いうまでもありませんが、このとき、あなたが TwentyOne.x をコピーしたディレクトリ名も含めて、フルパスで指定してください。

以上が終了したら、システムを再起動 (リセットボタンを押す) します。

(5) 使用プリンタの確認

あなたの使用するプリンタが前述のプリンタのどれに属するかを確認してください。インストール中に指定します。プリンタを使用しない場合には指定は不要です。

(6) 日本語フォントの準備

あなたの使用する日本語フォントを決めてください。本インストーラは、添付ディスク収録のフリーのビットマップフォント、およびアスキーの『パーソナル日本語 TeX フォントライブラリ』を除いて、日本語フォントがすでにインストールされていることを前提にしています。以下の要領で日本語フォントをインストールしておいてください。

○ ツアイトの Z's STAFF および Z's WORD JG、書体倶楽部のアウトラインフォント

すべてのファイルを同一ディレクトリに入れておいてください。インストール時にそのディレクトリを指定します。

○ ツアイトの Z's WORD JG Ver.3 のベジェフォント

明朝体、ゴシック体それぞれの第一水準フォント "*.FN1"、第二水準フォント "*.FN2"、記号等のフォント "*.FNO" を同一ディレクトリに入れてください。インストール時にそのディレクトリを指定します。

○ アスキーの『パーソナル日本語 TeX フォントライブラリ』のフォント

『パーソナル日本語 TeX フォントライブラリ』のオリジナルディスクを用意してください。インストールはインストーラが行います。なお、その際にディスクの空き容量が 20M バイト以上必要ですので注意してください。

○ 本書に添付されているビットマップの日本語フォント

インストーラがインストールを行いますから、今の時点での処理は不要です。

○ 本体内蔵の ROM フォント

はじめからインストールされているので、何の処理も必要ありません (当たり前ですね)。

以上の準備ができれば、いよいよインストールを開始します。

2.2.3 一括インストール

前述のとおり、一括インストールには 8.5M バイト以上の空き容量が必要です。空き容量が足りないと、指定ドライブの空き容量不足を指摘してインストールを中止するので注意してください。それでは、いよいよインストールを開始します。まず、X680x0 の FDD 0 に添付ディスク 1 を入れ、カレントドライブを FDD 0 に変えます。ここでは、FDD 0 を A ドライブとして説明を進めます。

A>install

12) インストーラは、キー入力待ち状態なら、いつでも ESC キーを押すことによって中断することができます。

と入力すると、インストーラが起動され、次のような画面が表示されます¹²⁾。

T e X I N S T A L L E R f o r X 6 8 0 x 0 V e r s i o n 1.00

インストール方法を選んでください。

(1) ディレクトリ固定インストール (初心者用インストール)

(2) ディレクトリ分割インストール

push 1 or 2

インストーラが起動したら、まず、インストール方法を決定します。ここでは一括インストールを行うので (1) を選んでください。(1) を選ぶと、次に Big T_EX を使用するかどうかを聞いてきます。

B i g - T e X を使用しますか？

(多量のメモリを消費します。使用しない人は No として下さい)

push Y or N

前述のとおり Big T_EX は大量のメモリを必要としますので通常は “N” キーを押してください。ここで、“Y” キーか “N” キーを押すと、次に本書付属のビットマップフォントを使用するかどうかを聞いてきます。

本書付属のビットマップフォントを使用しますか？

(標準容量 + 3.2 M b y t e の空き容量が必要です)

push Y or N

ビットマップフォントを使用するなら “Y” キーを、他の日本語フォントを使用するなら “N” キーを押してください。ここで “Y” を選択すると次のような画面になります。

ゴシック体フォント gothic32.bm1 はシェアウェアです。継続使用される方は disk7 の KAWASAKI.LZH 内のドキュメントの規定に従って送金してください。
ゴシック体フォントを使用しますか？ (push Y or N):

ゴシック体のビットマップフォントはシェアウェアですので、1週間以上継続して使用する場合は使用料の1500円を作者に支払う義務があります。詳細は添付ディスク7のKAWASAKI.LZH内のドキュメントに記されていますので、使用される方は必ず参照してください。

ゴシック体を使用されない方は、ここで“N”を選択してください。ゴシック体のかわりにROMフォントが使用されるように設定されます。

ビットマップフォントについての指定が終わると、次にT_EXをインストールするディレクトリを聞いてきます。

T_EXをインストールするドライブ、ディレクトリ名を指定してください。(ex.c:\tex)
(ディレクトリ無指定時は、指定ドライブ:\usr\local\texとします)
input dir name:

ここで、T_EX環境を作成するドライブ名およびディレクトリ名を入力してください。たとえば、Cドライブのtexというディレクトリにインストールしたいのならば、

c:\tex

と入力してください¹³⁾。なお、ドライブ名のみを指定した場合は、当該ドライブの

\usr\local\tex

というディレクトリとなります¹⁴⁾。あらかじめ指定したディレクトリが存在しなければ、ディレクトリを作成してインストールします。

次に、使用する日本語フォントを聞いてきますので、あなたの使用する日本語フォントを入力してください。

あなたの日本語環境を選んでください。

- (1) Z's STAFF又はZ's WORDのアウトラインフォント
- (2) 書体倶楽部のアウトラインフォント
- (3) Z's WORD JG IIIのベジェフォント
- (4) アスキーの98用T_EXのフォント
- (5) 本キット付属のビットマップフォント
- (6) 内蔵ROMフォント

push 1-6

ここで、内蔵ROMフォント以外のフォントを選択すると、日本語フォントの詳細を聞いてきます。以下に示すように、あなたの使用する日本語フォントにあわせて入力してください。

○ Z's STAFF または Z's WORD JG のアウトラインフォント

第二水準のフォントを持っていない場合には、本書付属のビットマップフォントでこれを代用することができます。そこでまず、この代用を行うかどうかの確認が求められます¹⁵⁾。

13) これ以降ディレクトリの指定は、すべてドライブ名まで含めたフルパスで指定してください。

14) このディレクトリ構成は、UNIXシステムでのディレクトリ構成に従ったものです。

15) ここでビットマップフォントによる代用を選択した場合には、明朝体・ゴシック体ともにビットマップフォントによって代用されることとなります。「明朝体はビットマップフォントで代用するが、ゴシック体はROMフォントで代用する」といった選択肢は用意していません。

第2水準フォントをお持ちで無い方は、本書付属のビットマップフォントで代用できます。

ただしゴシック体フォント gothic32.bm1 はシェアウェアですので継続使用される方は disk7 の KAWASAKI.LZH 内のドキュメントの規定に従って送金して下さい。

ビットマップフォントを使用しますか？ (push Y or N)

ここで“Y”を選択した場合、インストールの際にディレクトリ %TEXHOME%\freefonts が作成され、ビットマップフォント mincho32.bm1 および gothic32.bm1 がインストールされます。

次に、フォントファイルのあるディレクトリを聞いてきますので、ドライブ名を含むフルパス指定で入力してください。

アウトラインフォントのあるディレクトリを入力してください。
(ex.c:\zeit_fonts)

すると、フォントファイルの有無およびファイル名を、明朝体第一、第二水準、ゴシック体第一、第二水準のそれぞれについて聞いてきますので、入力してください。ただし、先に第二水準をビットマップフォントで代用するという選択をしている場合には、明朝体・ゴシック体ともに、第二水準のフォントファイル名の入力を求めることはありません。

明朝体第一水準フォントのファイル名を入力してください。(無指定時:mincho.vf1)

16) 特に変更していないかぎり
ファイル名は元のファイル名と同一です。

ただし、無指定時のフォント名と同一ファイル名¹⁶⁾ならリターンキーを押してください。もし明朝体第一水準フォントのファイル名が mincho.fon だとしたら、

mincho.fon

と入力してください。このとき、本インストーラは指定フォントファイルの有無を判定しますので、あらかじめフォントのインストールをすませておかなければなりません。

○ 書体倶楽部のアウトラインフォント

Z's STAFF のアウトラインフォントと同様です。

○ Z's WORD JG Ver.3 のベジェフォント

フォントファイルのあるディレクトリを聞いてきますので、ドライブ名を含むフルパス指定で入力してください。

ベジェフォントのあるディレクトリを入力して下さい。(ex.c:\jg_fonts)

次に、明朝体、ゴシック体それぞれについて、拡張子 *.FN? を除いたファイル名を指定してください。

明朝体フォントの主ファイル名（拡張子を除いたもの）を入力して下さい。
(ex. ZMIN1BN)

たとえば、明朝体のフォントファイルとして、ZMINOBY.FNO, ZMINOBY.FN1, ZMINOBY.FN2 があるとすれば、

ZMINOBY

と入力してください。このとき、本インストーラは指定フォントファイルの記号フォントファイル *.FNO、第一水準フォントファイル *.FN1、第二水準フォントファイル *.FN2 の有無を判定しますので、あらかじめフォントファイルのインストールをすませておかなければなりません。

○ アスキー社の PC-9801 シリーズ『パーソナル日本語 \TeX 』用の日本語フォント

まず、アスキー日本語フォントのインストールを行うかを決定します。

アスキー日本語フォントのインストールを行ないますか? (push Y or N):

すでにインストールされている場合や、あとから手動でインストールする場合は“N”を選択するとフォントインストール画面を終了します。インストーラを使用してインストールする場合は“Y”を選択してください。すると、次のような画面になります。

アスキー日本語フォントをインストールするドライブを指定して下さい。
(ex. c:)

アスキー日本語フォントをインストールするドライブ名を入力してください。このフォントは 20M バイト以上の空き容量のあるドライブにしかインストールできないので注意してください。なお、フォントがインストールされるディレクトリは指定ドライブの “\asc_font\jxl4” ディレクトリになります。以上の指定が終わると、インストールを開始します。画面に次のようにディスク挿入指示が出ます。

FD 0 にアスキー日本語フォントディスク 1 を挿入して下さい。

画面の指示に従って 18 枚すべてのディスクをインストールしてください。

○ 本書付属の日本語ビットマップフォント

フォントをインストールするディレクトリを聞いてきますので、ドライブ名を含むフルパスで指定してください。

ビットマップフォントを入れるディレクトリを入力してください。
(ex.c:\tex\freefonts):

ディレクトリを指定すると、添付ディスクの 6 および 7 を挿入するように求めてきますので、指示どおりにディスクを入れてください。フォントのインストールを行います。なお、ビットマップフォントを使用する際は HD の空き容量がさらに 3.2M バイト必要になりますので注意してください。

日本語環境の指定が終わると、プリンタ出力用フォントの dpi 値を求めてきます。

使用プリンタの DPI サイズを選んで下さい。
1)180 2)240 3)300 4)360 5)400 6)その他

もし、本書添付のビットマップフォントをインストールする場合には、ここで選択した値によってインストールするビットマップフォントが変更されます。選択した解像度が「180dpi」のときには 24 ドットフォントが、「240dpi」および「300dpi」のときには 32 ドットフォントが、「360dpi」のときには 46 ドットフォントが、「400dpi」のときには 48 ドットフォントが、「その他」のときには 32 ドットフォントが、それぞれインストールされることになります。

プリンタの dpi 値の入力が終わると、X680x0 版 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ システム対応プリンタの一覧が表示されます。

使用するプリンタを選択してください。

- | | | | | |
|-------------|--------------|-------------|-------------|-------------|
| 1)AP550 | 2)AP850 | 3)AP900 | 4)BJ10V | 5)BJ10V_B5 |
| 6)BJ130J_1 | 7)BJ130J_2 | 8)BJ330J | 9)CZ8PC3 | 10)CZ8PC4 |
| 11)CZ8PG1 | 12)CZ8PK7 | 13)CZ8PK8 | 14)FM_PR451 | 15)FUJITSU |
| 16)HG2500 | 17)HG800 | 18)HPDJ505J | 19)HPLJ3 | 20)HPLJ4 |
| 21)I0735X_E | 22)I0735X_N | 23)LBP180L | 24)LBP180RL | 25)LIPS2 |
| 26)LIPS3A4 | 27)LIPS3B4 | 28)LP3000 | 29)LP3000B5 | 30)LP3000SL |
| 31)LPTest | 32)LaserShot | 33)M1024 | 34)MJ500 | 35)NM9400S |
| 36)PCPR406H | 37)PR101E2 | 38)PR2K4A4 | 39)PR2K4B4L | 40)PR2K4B5 |
| 41)PR2KA4 | 42)TR24CL | 43)VP2000 | 44)VP800 | 45)VP950 |
| 46)その他 | | | | |

input no:

このなかからあなたの使用するプリンタの番号を選んで入力してください。画面に表示されるプリンタのうち、注意すべきものについて以下でまとめます。

- 4)BJ10V, 5)BJ10V_B5
4) は A4 判用、5) は B5 判用です。
- 6)BJ130J_1, 7)BJ130J_2
6) はコントローラボード Ver.3 用です。
- 15)FUJITSU
富士通の FM-漢字プリンタ用です。
- 19)HPLJ3, 20)HPLJ4
YHP の Laser Jet の、前者は III 用、後者は IV 用です。
- 23)LBP180L, 24)LBP180RL
キヤノンの LBP-180 用です。前者はすべてのページを単一のスタイルで出力する場合に、後者は左右のページで出力のスタイルを変更する¹⁷⁾ 場合に使用します。
- 28)LP3000, 29)LP3000B5, 30)LP3000SL
エプソンの LP-3000 用で、前から順に、A4 判出力、B5 判出力、SLiT_EX のスライド出力の際に使用します。
- 38)PR2K4A4, 39)PR2K4B4L, 40)PR2K4B5, 41)PR2KA4
NEC の PC-PR2000/4 用で、前から順に、A4 判の縦置き出力、B4 判の横置き出力、B5 判の縦置き出力、A4 判の縦置き出力 (高速版) の際に使用します。

もし対応するプリンタがこの一覧に存在しない場合や、そもそもプリンタを使用しない場合には、「その他」を選択してください¹⁸⁾。

以上、すべての指定が終わると、インストールを開始します¹⁹⁾。途中で数回ディスクの入れ替えを要求されますので、画面の指示に従ってディスクを交換してください。数十分後、全インストールが終了します。

2.2.4 分割インストール

ここまでで説明してきた一括インストールは、T_EX システムの関連ファイルをすべて同一ディレクトリに収めることができるためにシステム構造が簡素となり、したがってインストール後の管理がしやすいという点で、T_EX の初心者には特に勧めることができました。しかし、場合によっては 8M バイトを超える空き容量をつくりだすことができない場合もあると思います。あるいは、T_EX と METAFONT とで極端に使用頻度が異なる場合には、T_EX は高速な HD へ、METAFONT はバックアップがわりの MO へ (またはその逆) インストールしたほうが、貴重な高速メディアを無駄にしないで済むでしょう。そういう意味では、分割インストールは一定程度に T_EX のシステムに理解のある人に安心して勧められるインストール方法だといえるでしょう²⁰⁾。

分割インストールは、T_EX メイン部、T_EXpk フォント部、METAFONT 部の 3 つに分けてインストールします。それぞれの内容は、以下のとおりです。

17) 左マージンの大きさを変えるなど。

18) ここに対応するプリンタがない場合は、インストール終了後、第 5.3 節「print.cfg の作り方」を参考にしてコンフィギュレーションファイルを作成しなければなりません。

19) 画面上では LHA.x によるファイルの解凍作業が表示されています。

20) だからといって、初心者がこのインストール方法を選択しても、困るようなことはまずありませんから安心してください。

21) これらは単に解像度が違うだけで、実質的には同じものです。

○ T_EX メイン部

後述する pk フォントを除く T_EX に必要なファイルが入ります。

○ T_EX pk フォント部

T_EX が参照するフォントイメージファイルである pk フォントが入ります。画面表示用と印刷用の 2 種類があります²¹⁾。

○ METAFONT 部

フォントを作成するために必要なシステム METAFONT が入ります。META FONT の詳細は、第 5.4 節「METAFONT」(p.214)を参照してください。

このように、分割インストールを選んだ場合は 3 つのディレクトリに分けてインストールされますが、それ以外は一括インストールと変わりません。ですから、ここでは一括インストールと違う部分だけを記述します。

まず、一括インストールと同様にインストーラを起動します。すると、インストール方法を聞いてきます。

```
T e X  I N S T A L L E R   f o r   X 6 8 0 x 0   V e r s i o n   1 . 0 0
```

インストール方法を選んでください。

(1) ディレクトリ固定インストール (初心者用インストール)

(2) ディレクトリ分割インストール

push 1 or 2

ここでは、(2) ディレクトリ分割インストールを選んでください。すると、T_EX メイン部、pk フォント部、METAFONT 部の 3 つのインストールディレクトリを順番に聞いてきますので、ドライブ名からフルパス指定でそれぞれ入力してください。

— T_EX メイン部の指定 —

T e X をインストールするドライブ、ディレクトリ名を指定してください。(ex.c:\tex)

(ディレクトリ無指定時は、指定ドライブ:\usr\local\tex とします)

input dir name:

— pk フォント部の指定 —

P K フォントをインストールするドライブ、ディレクトリ名を指定してください。

(ex.c:\tex\fonts)

(無指定時は、T e X インストールディレクトリ\fonts とします)

input dir name:

— METAFONT 部の指定 —

MetaFontをインストールするドライブ、ディレクトリ名を指定してください。
 (ex.c:\tex\mf)
 (無指定時は、TeXインストールディレクトリ\mf とします)
 input dir name:

その後の日本語環境の指定以降は一括インストールと同様ですので、p.33以降を参照してください。

2.2.5 インストールチェック

本節では、インストーラによるインストールが無事終了しているかどうかをチェックします。まず、カレントディレクトリをインストール時に指定した TeX インストールディレクトリ²²⁾に移動してください。以降、このディレクトリを「TeX の HOME ディレクトリ」と呼ぶことにして、ディレクトリとして表記する場合には、%TEXHOME% で表すことにします²³⁾。そのディレクトリの下に TeXenv.bat というファイルができています。TeXenv.bat は、パスの追加および環境変数の設定を行います。コマンドライン上で、

C>TeXenv

と入力してください²⁴⁾。パスおよび環境変数が設定されます。設定される環境変数の詳細は『Vol.2 — Reference 編』の第 1.1 節「TeX の環境変数」(p.2) を参照してください。

なお、この TeXenv.bat の内容を AUTOEXEC.BAT の後半で指定しておくこと、起動時に自動的に TeX 環境を設定することができます。

また、第 2.2.2 項「インストールの前に準備しておくこと」(p.30) で述べたように、環境設定の際に環境エリアサイズが十分ないとエラーになることがあります。その場合は、コマンドラインから、

C>COMMAND /E:50

として環境エリアサイズを増やしてください²⁵⁾。環境エリアサイズの拡大は CONFIG.SYS で指定しておくこともできます²⁶⁾。その場合は CONFIG.SYS に次の 1 行を追加してください。

SHELL = COMMAND.X /P /E:50

環境変数の設定が終わったら、TeX の HOME ディレクトリ配下の doc 内にある jtexdoc.tex というファイル²⁷⁾を、画面閲覧用のプログラム preview.x で見てみましょう。

その前に、まず、このファイルを LaTeX で処理して dvi ファイル²⁸⁾を作成します。コマンドライン上で、

22) 分割インストール時は TeX メイン部です。

23) 実際、このディレクトリは、次の操作で環境変数 %TEXHOME% に設定されることになります。

24) これ以降、カレントドライブを C ドライブとして説明を進めます。

25) チャイルドプロセスを起動することになりますから、この方法はメモリが少ない人にはお勧めできません。

26) 詳しくは Human68k のユーザーズマニュアルを参照してください。

27) 日本語 TeX を使う際の注意等が記述されています。

28) dvi ファイルについては第 1.2.2 項「TeX の処理の流れ」(p.6) を参照してください。

```
C>latex %TEXHOME%/doc/jtexdoc.tex
```

と入力してください。このとき、パスの区切りには必ず“/”を使用しなければなりません。詳しい理由は、第 1.2.6 項「 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ が参照する環境変数」(p.16)および第 4.3.1 項「コントロール・シーケンス」(p.97)を参照してください。

無事に処理が終了すると、カレントディレクトリに `jtexdoc.dvi` というファイルが作成されます。`jtexdoc.dvi` が作成できなければ、インストールが正しく行われていないことになります。もう一度最初からやりなおしてみてください。`dvi` ファイルが作成されたら、いよいよ画面上で閲覧してみることにしましょう。

まず、「フォントマネージャ」と呼ばれる常駐プログラム `fontman.x` を登録します。フォントマネージャは、その名のとおり日本語フォントの生成管理をつかさどるプログラムで、本書の添付ディスクが構築する $\mathrm{X}680\times 0$ 版 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ の出力を、画面で表示あるいはプリンタで出力する際には必ず常駐させなければなりません。

```
C>FMset
```

と入力してください。画面上に次のページに示すようなメッセージが出力され、フォントマネージャが登録されます。

X68k Font Manager Ver 3.00c

Copyright (C) 1989,90,91,92 by E x t (T.Kawamoto)

フォントマネージャが常駐しました。

コンフィギュレーションを読み込みます。

ドライバを登録します。

ジェネレータ : f:/tex/fontman/bm1.sys f:/tex/ffonts/gothic32.bm1

ビットマップファイル : gothic32.bm1

常駐しました。(font ID = \$0001)

ジェネレータ : f:/tex/fontman/bm1.sys f:/tex/ffonts/mincho46.bm1

ビットマップファイル : mincho46.bm1

常駐しました。(font ID = \$0002)

フィルタ : f:/tex/fontman/smooth.sys -s 256 -p 32

常駐しました。(font ID = \$0003 , 親 font ID = \$0001)

フィルタ : f:/tex/fontman/smooth.sys -s 256 -p 46

常駐しました。(font ID = \$0004 , 親 font ID = \$0002)

フィルタ : f:/tex/fontman/totex.sys -j

f:/tex/fonts/min10.tfm -e f:/tex/dump

常駐しました。(font ID = \$0005 , 親 font ID = \$0004)

フィルタ : f:/tex/fontman/smooth.sys -s 256 -p 16

常駐しました。(font ID = \$0006 , 親 font ID = \$0000)

ミキサ : f:/tex/fontman/mixJIS2.sys

常駐しました。(font ID = \$0007)

フィルタ : f:/tex/fontman/totex.sys -j

f:/tex/fonts/goth10.tfm -e f:/tex/dump

常駐しました。(font ID = \$0008 , 親 font ID = \$0007)

f:/tex/fontman/totex.sys -j

f:/tex/fonts/goth10.tfm -e f:/tex/dump

常駐しました。(font ID = \$0009 , 親 font ID = \$0006)

フィルタ : f:/tex/fontman/smooth.sys -s 256 -p 24

常駐しました。(font ID = \$000A , 親 font ID = \$0000)

フィルタ : f:/tex/fontman/totex.sys -j

f:/tex/fonts/min10.tfm -e f:/tex/dump

常駐しました。(font ID = \$000B , 親 font ID = \$000A)

登録は正常終了しました。

ただし、画面に出力されるメッセージは、各個人のインストール環境によって異なりますから、つねにこのように表示されるわけではありません。ここでは、画面表示の最後に「登録は正常終了しました。」という 1 行があることを確認してください。この 1 行が出力されずに常駐作業が終わってしまっていたら、残念ながらインストールに失敗したか、あるいはメモリが足りないことを示します。本章のはじめから、もう一度、自分の環境を確認してください。

登録が終了したら、

C>preview jtexpdoc.dvi

と入力してください。

どうでしょうか？ 無事、画面に `jtexdoc.dvi` の出力結果が表示されましたか？ 画面に日本語 \TeX という表題が表示され、その下に 1 章として日本語 \TeX の概要が書かれていれば、インストールは成功です。もしうまく表示できていなければ、もう一度最初からやりなおしてみてください。指示どおりに行えば必ずインストールできるはずですので、頑張ってください。なお、今、インストールした \TeX システムの実際の使用法は、第 4 章「Exercise」(p.77) で詳しく説明してありますので、そちらを参照してください。

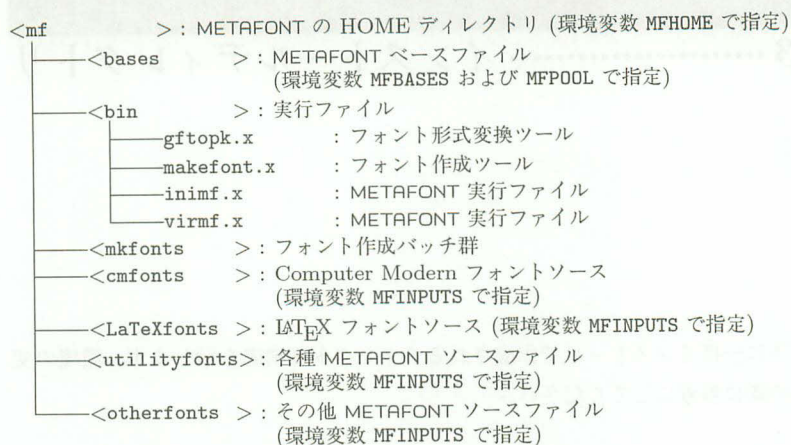
2.3 インストールディレクトリ

以下に一括インストールで作成されるディレクトリ構成を示します。環境の変更等の際に参考にしてください。

```

<tex          >: TEX の HOME ディレクトリ (環境変数 TEXHOME で指定)
├── <batch     >: 各種バッチファイル
│   ├── FMset.bat      : フォントマネージャの設定
│   ├── FMfree.bat     : フォントマネージャの解除
│   ├── LaTeX.bat      : LATEX の実行バッチファイル
│   └── TeX.bat         : plain TEX の実行バッチファイル
├── <bin       >: 実行ファイル
│   ├── initex.x       : TEX 実行ファイル
│   │                   ( X680x0 標準の TEX あるいは Big TEX )
│   ├── virtex.x       : TEX 実行ファイル
│   │                   ( X680x0 標準の TEX あるいは Big TEX )
│   ├── jplain.fmt      : plain TEX 用フォーマットファイル
│   │                   (環境変数 TEXFORMATS で指定)
│   ├── jlplain.fmt     : LATEX 用フォーマットファイル
│   │                   (環境変数 TEXFORMATS で指定)
│   ├── fontman.x       : フォントマネージャ
│   ├── preview.x       : プレビューア
│   └── print.x         : プリンタドライバ
├── <doc       >: TEX ドキュメント類
├── <drivers    >: TEX ドライバ関連ファイル
├── <dump       >: フォントマネージャワーク
├── <fontman    >: フォントマネージャ
├── <fonts      >: フォント (環境変数 TEXPK および TEXFONTS で指定)
├── <jmacros    >: 日本語マクロ集 (環境変数 TEXINPUTS で指定)
├── <macros     >: マクロ集 (環境変数 TEXINPUTS で指定)
├── <mf         >: METAFONT の HOME ディレクトリ
│               (環境変数 MFHOME で指定)
├── <pool       >: TEX 用 pool ファイル (環境変数 TEXPOOL で指定)
├── myfonts.fm   : フォントマネージャ用のコンフィギュレーションファイル
├── preview.cfg  : プレビューア設定 (環境変数 PREVIEW.CFG で指定)
├── print.cfg    : プリンタドライバ設定 (環境変数 PRINT.CFG で指定)
├── preview.p3m  : プレビューア設定 (環境変数 PREVIEW.P3M で指定)
├── print.p3m    : プリンタドライバ設定 (環境変数 PRINT.P3M で指定)
└── TeXenv.bat   : TEX 環境変数設定

```



なお、添付ディスク 8 に収録されているファイルは、必要に応じて、皆さん自身がこの環境内にインストールしなければなりません。どのディレクトリに何をインストールすればよいかは、それぞれのアーカイブに含まれているマニュアルを参照し、よくわからなければインストールしたいファイルと同じ拡張子のファイルが集まっているディレクトリにインストールしてください (とというのは、後者はあまり勧められるような方法ではありません)。

2.4 …… インストーラのエラーメッセージ一覧

- **TwentyOne.x が組み込まれていません。処理を中止します。**
インストールするには TwentyOne.x が組み込まれていなければなりません。第 2.2.2 項「インストールの前に準備しておくこと」(p.30) で説明したように、TwentyOne.x を常駐させてください。
- **ESC キーが押されましたので処理を中止します。**
インストーラは、キー入力待ち状態で ESC キーを押すことにより処理を中止することができます。
- **正しいドライブ名 (ドライブ名:) を指定してください。**
ディレクトリ名入力時にはドライブ名も入力する必要があります。このメッセージは、ドライブ名が正しく入力されていないときに表示されます。ドライブ名も指定してください。
- **ディレクトリ名も指定してください。**
ディレクトリ名入力時に、ドライブ名しか入力されていないときに表示されるメッセージです。ディレクトリ名も指定してください。
- **ディレクトリはルートディレクトリから指定してください。**
ルートディレクトリからディレクトリを指定していないときに表示されるメッセージです。ルートディレクトリから指定してください。
- **指定ドライブの空き容量が不足しています。?M Byte 以上空けてからインストールしてください。**
インストール先のディスク容量が不足しているときに表示されるメッセージです。指定されたディスク容量を空けてから再度実行してください。
- **ディレクトリ作成は行いません。作業を中止します。**
インストール先に指定したディレクトリが存在しない場合、ディレクトリの作成を行うかどうかを確認してきます。作成を行わないと、インストール作業を続行することが不可能になるので処理を中断します。
- **指定ファイルは存在しません。**
日本語フォントファイルの指定時に、指定したフォントファイルが存在しないときに表示されるメッセージです。
- **ファイルが見つかりません。作業を中止します。**
これらのメッセージは本来出力されることはありません。これらのメッセージが出力されるということは、インストール環境そのものに問題があります。再度インストールしてください。

Device Driver

\TeX の処理結果を目で見るためには、専用のプログラムが必要です。本書の添付ディスクが構築する \TeX システムでは、X680x0 の画面で見るためのツールとして“プレビューア”を、プリンタに打ち出すためのツールとして“プリンタドライバ”を、そしてこれらのプログラムから呼び出される常駐型のプログラムとして“フォントマネージャ”を利用することができます。また、添付ディスクには、インストーラではインストールされませんが、 \TeX ファイルをファックスで送るためのツール“tex2ttl”、“ttl2fax”、“fax2ttl”も含まれています。本章では、これらのプログラムについて簡単に解説します。

3.1 デバイスドライバ

T_EX が出力する dvi ファイル (Device Independent File) は、その名のとおり特定の機種に依存しない記述形式になっています。少しだけ具体的にいえば、dvi ファイルは、「dvi 命令」と呼ばれる、仮想のページ記述命令群がコード化されたものによって記述されているのです。

このこと自体は、ユーザが T_EX を使用するために特に意識する必要はないので説明を省きますが、T_EX の処理結果を目で見える形にするためには、この dvi 命令を解釈し、使用する機種に応じた命令に置き換えるプログラムが必要です。このようなプログラムのことを、T_EX の世界では「デバイスドライバ」といいます。ここで、「デバイスドライバ」というと、たとえば OPMDRV.X や PRNDRV.SYS のように、OS に組み込まれてデバイス (周辺装置) の処理を行うプログラムを思い浮かべる人もいるかもしれませんが、しかし、T_EX の世界でいうところの「デバイスドライバ」は、「出力装置固有の処理を行う」という意味を転じてこう呼ばれるのであって、その性質からすると、むしろ「dvi 命令のインタプリタ処理プログラム」といったほうがより正確でしょう。

さて、世の中には非常にたくさんの T_EX のデバイスドライバが存在していますが、そのうち日本語の扱える X680x0 用のデバイスドライバには次のものがあります。

○ preview.x

Human68k 用プレビューアです。T_EX の処理結果を画面で確かめるためのツールで、作者は Ext (川本琢二。筆者)¹⁾です。preview.x は本書に添付されています。最新版のサポートは NIFTY-Serve で行われています。

○ SXpreview.x

SX-WINDOW 用プレビューアです。作者は SHUNA 氏です。梁山泊ネットあるいは NIFTY-Serve で手に入れることができます。NIFTY-Serve での所在は次のとおりです。

```
nifty / fsharp3 / lib 1 / 243 : SXpre110.Lzh
```

拡張子が .tex であるファイルのアイコンをドラッグしてきて SXpreview.x のあるフォルダーでマウスボタンを離すと、そのファイルを読み込んで T_EX を起動してくれます。

SX-WINDOW 上で T_EX のすべての操作を行うことができます。

1) NIFTY-Serve での ID は NAH00720 です。

○ `kodvi.win`

Ko-Window 用プレビューアです。作者は RAO²⁾氏です。NIFTY-Serve で手に入れることができます。

NIFTY-Serve での所在は次のとおりです。

```
nifty / fsharp3 / lib 2 / 796 : kodvi110.Lzh
```

2)NIFTY-ServeでのID
は NAA01617 です。

○ `print.x`

ラインプリンタ用のプリンタドライバです。T_EX の処理結果をラインプリンタに出力するためのツールで、幅広い汎用性を備えています。作者は Ext です。`print.x` は本書に添付されています。最新版のサポートは NIFTY-Serve 上で行われます。

○ `dvi2ps.x`

ポストスクリプトプリンタ用のプリンタドライバです。もともと、UNIX 用のツールを移植する形で作成されました。移植者は K-ras³⁾氏です。`dvi2ps.x` 1-7j (X6.00) が NIFTY-Serve で手に入ります。

NIFTY-Serve での所在は次のとおりです。

```
nifty / fsharp3 / lib 9 / 65 : dv2ps17.Lzh
```

3)NIFTY-ServeでのID
は HFG02505 です。

本章では、このうち `preview.x` と `print.x` (それぞれ「プレビューア」、「プリンタドライバ」と呼びます) について簡単に使用方法を説明します。オプション一覧等の詳しい説明は『Vol.2 — Reference 編』を参照してください。

なお、本書添付のディスクにも、この 2 つの実行プログラムを収めてあります。第 2 章「Install」(p.21) で説明するように、添付ディスク 1 の `install.x` で T_EX をインストールすれば使えるようになっていますので、ぜひ活用してください。

また、X680x0 用 T_EX には日本語フォントを管理するフォントマネージャが備わっています。これは、他の T_EX システムにはない特徴です。フォントマネージャは、プレビューアやプリンタドライバで使用する日本語フォントの生成関係の処理を担当する常駐ソフトです。各種のフォントドライバから選んでフォントを登録する形態をとっているため、ユーザの各種ニーズに応えることができます。フォントマネージャを熟知したユーザなら、後述のコンフィギュレーションファイルを書き換えることによって常駐サイズを小さくするように工夫することもできます。

本章では、フォントマネージャの基本的な使い方と常駐方法についても説明します。

3.2……………フォントマネージャ — 文字の変形

3.2.1 概要

先に簡単に説明しましたが、フォントマネージャとは、プレビューアやプリンタドライバで使用する日本語フォントの生成関係の処理を担当する常駐ソフトのことです。フォントマネージャ自体の使用について説明する前に、まず、なぜこのようなプログラムを用意することになったかを説明することにします。

そもそも、大文字と小文字をあわせて 52 文字しかない英語フォントならともかく、ひらがな・カタカナ・漢字と、日本語フォントはデータ量が膨大です。この、ただでさえデータ量が膨大な日本語フォントにおいて、あらゆるサイズの、あらゆる変形フォントのデータを、あるいは異なるフォントのデータを、すべてファイルとして持たなければならないとしたら、おそらく (それはそれで優れたシステムではありますが) とんでもないことになるでしょう。

そこで、X680x0 版のデバイスドライバは、日本語フォントの生成・管理を行う部分を、フォントマネージャとして独立させたのです。このことは、単にフォントデータそれ自体のディスク占有領域を小さくするというだけではなく、フォントデータとフォントデータを操作するプログラム (フォントドライバ) とを分けることによって、メモリ消費の点でも有利になったことを意味しています。

しかも、フォントデータを操作するプログラムは、たとえ異なるフォントを操作する場合にも、同一の操作を行うかぎり共有することができますから、その意味でもメモリをより節約することができるのです。

フォントマネージャは、もともと X680x0 版 $\text{T}_{\text{E}}\text{X}$ 用に開発されたものですが、このような利点のために、各種のフリーソフトウェアでも利用されています。

たとえば、X680x0 版の $\text{GHOSTSCRIPT}^{1)}$ や、 $\text{mfged}^{2)}$ で利用することができますが、これはまさにフォントマネージャの適応能力の高さを物語っているといえるでしょう。

1) GNU のポストスクリプトエンジンです。NIFTY-Serve で手に入ります。

2) フリーのグラフィックエディタです。これも NIFTY-Serve で手に入ります。

3.2.2 フォントドライバ

ところで、フォントマネージャを独立したプログラムにしたことによって生じる利点は、前項に示しただけではありません。

フォントマネージャは、新たにフォントデータやフォントデータの操作を追加したくなった場合に簡単に対処できるようになっているために、便利かつ効率的なものになっているのです³⁾。具体的にどうなっているかというと、フォントマネージャは、3 系統の「フォントドライバ」と呼ばれる小プログラムを組み合わせる形態をとって、日本語フォントの生成処理を管理しています。したがって、新たにフォントデータやフォントデータの操作を追加したくなった場合、なにもフォントマネージャ全体を書き換える必要はなく、このような小プログラムを作成するだけですむのです。

以上のような意味からいえば、フォントマネージャの機能は、フォントドライバの機能を組み合わせることで実現するのですから、“フォントマネージャとフォントドライバの関係”は“OS およびシェル (Human68k および COMMAND.X) とそこで動くツール (実行ファイル) の関係”にあたるといえば、おそらく直感的な理解はしやすいでしょう。

本項では、フォントマネージャが実現する「機能」をつかさどる、それぞれのフォントドライバについて簡単に説明することにしましょう。

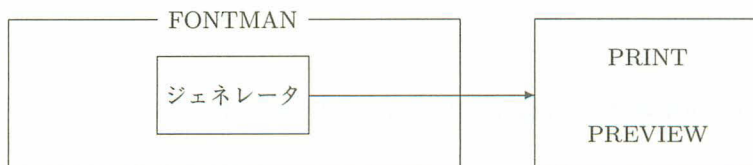
3) 実際、フォントマネージャは、さまざまなフォントに迅速に対応してきました。

◆ フォントジェネレータ

ひとつめのフォントドライバを「フォントジェネレータ」と呼びます。これは、フォントデータ (たとえばフォントファイルなど) に対応する情報を持っていて、その情報をもとに実際のフォントデータを呼び出し、ビットマップに描画する機能⁴⁾を持っています。

フォントジェネレータは複数あってもかまいません。フォントジェネレータの数は、ユーザの所有するフォントファイルの種類に応じて変わります。

4) 一般的なウィンドウシステムに付属するフォントマネージメントシステムは、たいてい、この機能しか持っていません。

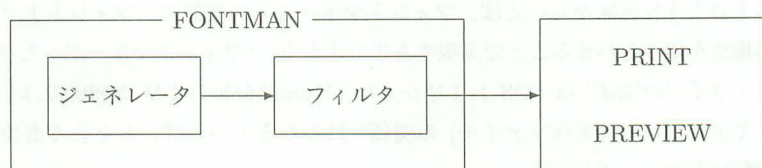


◆ フォントフィルタ

2 つめのフォントドライバは「フォントフィルタ」です。

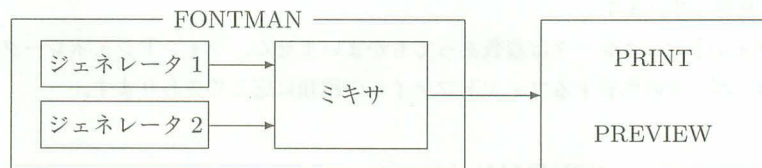
フォントマネージャでは、フォントデータは、たとえば Human68k の標準入力と同じようなイメージで扱われます。先ほど説明したフォントジェネレータも、作成したフォントデータを標準出力に吐き出します⁵⁾。

フォントフィルタは、フォントジェネレータが標準出力に吐き出したフォントデータを標準入力して、フォントデータに変形操作をほどこします。ちょうど Human68k のコマンドラインで使用するフィルタコマンドと同じイメージです。基本となるフォントデータをフォントフィルタにかけることで新たなフォントデータが作成されます。



◆ フォントミキサ

最後のフォントドライバは、「フォントミキサ」です。フォントジェネレータが出力した複数のフォントデータを、フォントミキサによって交ぜることで、新しいフォントデータを作成することができます。



Human68k のコマンドには、フォントミキサに該当するものではありません。

◆ フォントドライバの機能

これらのフォントドライバを組み合わせることによって、次に挙げるようなさまざまな機能を実現することができます。

5) 実際には標準出力ではなく、専用のインターフェースがあります。

○ フォントミキサ

フォントミキサは、もともと、X680x0 のアウトラインフォントがツァイト社のもの、それも第一水準漢字のフォントしかなかった頃、どうしてもツァイトのフォントで第二水準漢字もサポートしたいという要求から生まれたものです。

その頃のフォントマネージャには当然のことではありますが、フォントミキサというアイデアはありませんでした。第二水準

漢字のフォントがないと、第二水準漢字を表示する場合、四角の黒い箱 (T_EX ではボックスと呼んでいます) で代用せざるを得ませんでした。しかし、文章中に四角の黒い箱が出てくるのでは見栄えが悪いということから、第二水準漢字 だけ ROM フォントで代用できないかということになりました。ここからフォントミキサというアイデアが導入されたのです。

- (1) ユーザのフォント環境にあわせて利用フォントを選択する機能
- (2) 複数のフォントを組み合わせる機能
- (3) フォントの太さを調整する機能
- (4) フォントサイズ別の分担機能
- (5) 任意のフォントを傾ける機能
- (6) いわゆる変形フォントのサポート機能

(1) のフォント環境にあわせて利用フォントを選択するというのは、利用するフォントにあわせたフォントジェネレータを用意するということの意味します。フォントジェネレータの設定のしかたについてはここでは述べませんが、本書添付のインストーラ⁶⁾を使えば、インストールの過程でユーザの環境について対話的に入力を求めることによって自動的に行ってくれます。

(2) のフォントの組み合わせとは、次のようなことです。たとえば、ツァイトの Z's STAFF Ver.2 に付属しているフォントしか持っていないと第二水準漢字のフォントがないことになります。このとき、代案として、ROM のフォントを拡大・縮小したものを使うことが考えられました。フォントミキサを利用すれば、このようなフォント同士の組み合わせを実現することができます。フォントミキサの設定のしかたについてはここでは述べませんが、本書添付のインストーラを使えば、インストールの過程でユーザの環境について対話的に入力を求めることによって自動的に行ってくれます。

(3) のフォントの太さについてですが、たとえば、書体倶楽部のフォントはフォントを描く線の太さが全体的に太めですが、この機能を使えば、ダイエット (細く) させることができます。ダイエットさせる場合は、フォントフィルタとしてダイエット (diet.sys) を実装し使用します。ダイエットの設定についてはここでは述べませんが、本書添付のインストーラを使えば、インストールの過程でユーザの環境について対話的に入力を求めることによって自動的に行ってくれます。また、より高度な利用例を第 5.2.3 項「ツァイトフォントとダイエットフィルタの組み合わせ」(p.177) に挙げておきました。

(4) のフォントサイズ別の分担機能とは、たとえば、60 ドット以上の大きなフォントはアウトラインフォントで、それより小さなフォントはビットマップフォントで表示するといったように、フォントのサイズにあわせて表示用のフォ

6) 第 2 章「Install」(p.21) 参照。

ントを使い分ける機能です。専用のフォントミキサをつくることによって使うことができます。フォントミキサの作り方は、フォントマネージャの配付アーカイブ内のドキュメントを参照してください（本書が構築する \TeX システムの $\% \text{\TeXHOME} \% \backslash \text{fontman} \backslash \text{doc}$ 配下のファイルです）。

(5) で挙げたフォントを傾ける機能を、「スラント処理」といいます。スラント処理を行う場合は、斜体フィルタを使います。

(6) のフォントの変形とは、フォントを縦に引き伸ばしたり（長体）、上から押しつぶして平たくしたり（平体）するような機能です。(5) で示したフォントを傾ける機能とも組み合わせることができます。平体、長体用のフォントフィルタをつくることによって対処できます。フォントジェネレータのなかには、これらの機能を持っているもの（ツァイトの『書体倶楽部』のフォント用ジェネレータ）もあります。このフォントジェネレータを使用したサンプルについては、第6章「 \TeX family」の \VfIATeX (p.234) を参照してください。

また、フォントフィルタの例として特筆しておかなければならないものとして、キャッシュフィルタがあります。一度作成したフォントをメモリに蓄えておくことで速度的には有利になりますが、反面、膨大なメモリ容量を使ってしまうので、メモリに余裕がないような場合には十分気をつける必要があります。

3.2.3 フォントマネージャの常駐

X680x0 版 \TeX では、プレビューアおよびプリンタドライバを使用するときには、フォントマネージャを常駐させておかなければなりません。 \TeX 本体 (virtex.x) の処理をしている間は、フォントマネージャを常駐していなくても問題ありません。メモリが少ない人は、 \TeX で処理を行う際（特に Big \TeX を使用する場合）にはフォントマネージャを常駐させないほうがよいでしょう。

本書の添付ディスクと付属のインストーラによって構築した \TeX システムの場合、フォントマネージャの常駐方法は、次のとおりです。

```
A> cd %\TeXHOME%
A> fontman myfonts.fm
X68k Font Manager Ver 3.00c Copyright (C) 1989,90,91,92
by E x t (T.Kawamoto)
フォントマネージャが常駐しました。
コンフィギュレーションを読み込みます。
ドライバを登録します。
ジェネレータ : A:/usr/local/TeX/fontman/zs.sys -g -s 256
:
:
登録は正常終了しました。
```

このようにフォントマネージャの常駐はいたって簡単です。常駐を解除すると

きには次のようにします。

```
A> fontman -r
X68k Font Manager Ver 3.00c Copyright (C) 1989,90,91,92
by E x t (T.Kawamoto)
    フォントマネージャは解除されました。
```

この例のように、「フォントマネージャは解除されました。」というメッセージが出力されてはじめて、メモリ上に常駐しているすべてのフォントマネージャの常駐が解除されたことを示します。

この表現で察することができると思いますが、フォントマネージャは異なる設定⁷⁾で複数回数(これを「depth = 深さ」ということにします)常駐することができますが、1回の常駐解除の命令で解除されるのは最後に常駐させた設定ひとつ(1depth)だけです。したがって、「1 depth 分のフォントドライバが解除されました。」というメッセージしか出ない間は、まだメモリ上にフォントマネージャが常駐していることになります。

7) 別に同じ設定でもかまいませんが、メモリを浪費するだけです。

3.3プレビューア — 画面での閲覧

1) 一定の文法にしたがって、ソースファイルにコマンドなどを埋め込んでいく方法のことです。

第1章「T_EX」(p.1)でも触れたように、Macintosh の DTP ソフトのような WYSIWYG 方式を採用しているソフトとは違い、個々の単語や行ごとにコマンドを埋め込むマークアップ方式¹⁾の T_EX システムでは、T_EX のソースファイルを書いている間は仕上りのイメージをコンピュータ上で視覚的に確認することはできません。しかし、ある程度原稿を書き進むにつれ、今、書いている原稿がどのような仕上がりになるのかを画面上で確かめたくなるものです。

X680x0 版の T_EX には `preview.x` という画面閲覧のためのデバイスドライバがあります。通常、これは「プレビューア」と呼ばれています。「プレ」は日本語だと“前”のことであり、また、「ビューア」は“閲覧する道具”のことですから、「プレビューア」は、“プリントアウトする前に仕上がりイメージを画面上で閲覧するための道具”というわけです。

3.3.1 プレビューアの使用法

さて、プレビューアの実行方法を説明するためにサンプルのドキュメントを用意します。ここでは、プレビューアの作者がプレビューア用に用意したドキュメント²⁾がありますので、これを利用しましょう。

まず、作業を行うためのディレクトリを確保します。自分のハードディスクのなかから高速で、空き容量が 1M バイト以上あるパーティションを選び、そこにディレクトリをつくります。たとえば、ディレクトリを `C:\WORK` とします。

ディレクトリをつくったら、インストール時に作成した T_EX の HOME ディレクトリをのぞいてみてください。ここに `drivers` というサブディレクトリがあります。さらに、そのなかの `doc` ディレクトリには、デバイスドライバのドキュメント類が入っています。

`preview.tex`, `print.tex`, `common.tex` を探してください。

2) L^AT_EX で書かれています。


```

A> cd %TEXHOME%\drivers\doc
A> dir
ボリュームがありません a:\usr\local\TeX\drivers\doc
    17 ファイル          4677K Byte 使用中          3466K Byte 使用可能
ファイル使用量          145K Byte 使用
FastRaster.tex          13226  93-04-03  14:01:54
QandA.doc                22722  93-03-14  17:51:56
common.tex               9872   92-11-17  23:13:36
cutbox.sty               2771   92-11-17  23:13:50
verup.doc                6055   93-11-03  22:26:12
preview.p2m              715    92-11-17  23:14:14
preview.p3m              1498   92-11-17  23:14:26
readfirst.doc            4246   93-10-24  19:10:02
print.p2m                838    92-11-17  23:15:18
print.p3m                1462   92-11-17  23:15:28
test.tex                 15298  93-11-03  22:03:06
readme.doc               4314   93-06-26  14:11:48
sources.doc              1713   92-11-17  23:21:00
tpicdoc.tex              15270  92-11-17  23:21:16
preview.tex              17181  93-10-11  15:44:20
print.tex                19288  93-10-11  15:44:26

```

preview.tex は、プレビューアを操作する人のためのガイドを書いたドキュメントです。print.tex は、 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ でつくったファイルの仕上がりをプリンタに出力する際にプリンタドライバを操作するためのガイドと、各種プリンタに対応するためのオプションについての説明がなされているファイルです。

また、common.tex は、プレビューアとプリンタドライバの共通事項の説明がなされています。たとえば、Tpic special という、 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ 文章中に絵を埋め込むための標準的なコマンド群についての説明や、X680x0 標準のビットイメージ (title.sys 形式) を $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ 文章中に埋め込むための独自の拡張命令についての説明などです。common.tex は、preview.tex や print.tex から呼び出す形で利用します。したがって、common.tex 自身を $\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ で処理することはありません。

では、先ほど確保した作業用ディレクトリに、この 3 つのファイルをコピーします。

```

A> copy preview.tex C:\WORK
preview.tex
    1 個のファイルをコピーしました
A> copy print.tex C:\WORK
print.tex
    1 個のファイルをコピーしました
A> copy common.tex C:\WORK
common.tex
    1 個のファイルをコピーしました

```

そして、作業用ディレクトリに移ってください。

```
A> C:
C> cd C:\WORK
```

先ほどコピーしてきたファイルを \LaTeX で処理しましょう。

```
C> latex preview
C> latex preview
```

2 度 \LaTeX にかけてください。print.tex のほうも同様です。dvi ファイル preview.dvi ができたら、それを画面で見てください。

その前に、フォントマネージャはちゃんと常駐していますか？ 確認してみましょう。

次のようにコマンドを打ち込んでみてください。

```
C> A:
A> cd %TEXHOME%
A> fontman -a
```

そして、次のような画面になれば、フォントマネージャが常駐していることがわかります³⁾。

```
A> cd %TEXHOME%
A> fontman -a
X68k Font Manager Ver 3.00c Copyright (C) 1989,90,91,92
by E x t (T.Kawamoto)
-tex-高速-明朝 = sharp-rom-平滑-tex[min]
-通常 = sharp-rom-平滑
-tex-高速-ゴシック = sharp-rom-平滑 (16)-tex[goth]
-強調 = sharp-rom-平滑 (16)
-tex-明朝 = zeit-明朝-tex[min]
-tex-ゴシック = zeit-ゴシック-tex[goth]
-明朝 = zeit-明朝
-ゴシック = zeit-ゴシック
```

もし次のように表示されたら、フォントマネージャは常駐していません。

3) 表示画面の細部については、ここに表示したものと異なる場合があります。

```
A> cd %TEXHOME%
A> fontman -a
X68k Font Manager Ver 3.00c Copyright (C) 1989,90,91,92
by E x t (T.Kawamoto)
    フォントマネージャは登録されていません。
```

第 3.2 節「フォントマネージャ — 文字の変形」(p.50)に戻って、フォントマネージャを常駐させてください。

次に、T_EX の HOME ディレクトリに以下のファイルがあることを確認してください。

○ preview.cfg

よく使うオプションを入れておくためのコンフィギュレーションファイルです。あなたの好みに応じて書き換えてもかまいません。オプションの一覧と説明は、『Vol.2 — Reference 編』の p.39 を参照してください。

○ preview.p3m

pk ファイル⁴⁾名を FONTMAN Ver.3 でのフォントネームに変換する際の規則を入れておくファイルです。ほとんど変更することのないファイルです。

いよいよ dvi ファイルを画面で閲覧してみます。

```
A> C:
C> preview preview
```

4) 第 1.2.8 項「T_EX が使うファイルの拡張子」(p.19)を参照してください。

と入力してみてください。画面がそのうち (あなたのマシンが X68030 なら即座に) 真っ白になります。しばらくすると、画面に黒い字でプレビューアの画面イメージが表示されます。

カーソルキーを使えば、ページのスクロールができます。ページをめくるのは、ROLL UP キーや ROLL DOWN キーで行います。これらのキーを使ってプレビューアのドキュメントをご覧になってください。

プレビューアの操作がひととおりわかったら、プレビューアを終了しましょう。ESC キーを押してください。軽くキーに触れる程度で十分です。これでプレビューアが終了します。

プレビューアを使えば、仕上がりを実際印刷するまで何度でも画面上で編集結果を確認することができます。

ここでちょっと寄り道をしてみます。

プレビューアを使ってページをめくる際、イライラするくらい遅いときと、ほんの一瞬でめくれるときがあると思います。特に、前のページに戻るときは画面表示が速いことがわかります。プレビューアでは、X680x0 のハードウェア的な遅さをカバーするべく、次に挙げるような工夫をして見かけ上の速さを確保しています。

(1) 画面表示の裏で次のページの描画を行う

X680x0 以外の機種対応のプレビューアは、ユーザがページをめくった瞬間に次のページの描画を開始して表示する仕様のものがほとんどです。他の機種の場合はハードウェア的に十分速いので、このような方法をとっても実用に堪えられます。しかし、X680x0 版のプレビューアでこの方法をとると、遅くて使いものになりません。そこで、X680x0 版のプレビューアは、現在のページ表示中に裏で次のページの描画を行うようにつくられています。おかげでページをめくったときには、すでにそのページの描画は終わっており、画面にそのページを転送するだけでよいのです。プレビューアは、このような仕様になっているため、たとえハードウェアが遅くても、それなりの速度でプレビューできるようになっています。

もっとも、X68030 の場合はハードウェア的に十分速いのですが、それでもこの工夫によって、見かけの速度がより一層速くなります。ページをめくったときの速度は他機種に比べても遜色がないくらい高速です。

(2) ページバッファ

プレビューアは、ページバッファを複数持てるようになっており、すでに描画の終わったページをメモリにストックする方式を採用しています。そのバッファ数は、T_EX の HOME ディレクトリ内のファイル `preview.cfg` に書き込まれています。標準では 5 ページ分です。

List 3-1 • `preview.cfg`

```
1: -buf=5
2: -GRAM
```

1 行目の `-buf=` のあとに書かれた数字を変更すれば、ページバッファを 7 ページに増やしたり、逆に 3 ページに減らしたりすることができます。

また、グラフィック RAM を、すでになんらかの目的 (たとえばグラフィック RAM ディスク) で使用している場合には、2 行目の `-GRAM` を削除してください。

以上、プレビューアの簡単な使い方を説明しました。拡大表示や、代用表示といった高度な利用をしたい場合は、オプションをつけて起動することでこれを制御します。詳しくは、『Vol.2 — Reference 編』の第 2.3 節「PREVIEW オプション」(p.39) のほか、今、画面上に表示されているマニュアル (`preview.dvi`) にも書いてありますので、活用してください。

なお、プレビューアはいくつかの環境変数といくつかのファイルを参照します。この点についての詳しい説明も、『Vol.2 — Reference 編』の第 1.3 節「PREVIEW 環境変数」(p.7) にありますので参照してください。

3.3.2 プレビューアの起動方法

プレビューアの起動方法について、正確に記述しておきます。コマンドライン

からの書式は、以下のようになっています。

```
preview [オプション] foo [オプション] [ページ] [オプション]
```

[オプション] には、『Vol.2 — Reference 編』の第2.3節「PREVIEW オプション」で説明しているオプションを書きます。書く場所は、上の書式のように“preview”の後ろならコマンドラインのどこに書いてもかまいません。

foo には、画面で閲覧したい dvi ファイル名を書きます。なお、拡張子の .dvi はつけなくてもかまいません。

[ページ] は、画面で閲覧したいページ範囲を指定するもので、ページ番号の範囲を書きます。また、不要なら省略することもできます。このページ番号は、プレビューしたときに画面下部のフッタに表示されているページ番号とは違うことがあります。あくまで、先頭ページを 1 ページ目とした通し番号ですので、注意してください。

[ページ指定] のサンプルをいくつか挙げて説明しましょう。

- preview foo
画面表示したい dvi ファイルのすべてのページを表示できます。
- preview foo 3-10
3 ページから 10 ページまでを表示できます。
- preview foo 1-4 16-20 30-
1 ページから 4 ページまでと、16 ページから 20 ページまで、そして 30 ページから最後までを表示できます。
- preview foo 1--9
1 ページから 9 ページまでを、1 ページおきに表示できます。つまり、1, 3, 5, 7, 9 ページだけが表示の対象となります。ページ指定が “--” となっていることに注意してください。

3.3.3 プレビュー画面表示中のキー操作

次に、プレビューを起動したあと、ドキュメントの表示中に使えるキー操作について説明します。

この節では、ページ指定がない場合のキー操作について説明しています⁵⁾。もし表示ページの指定をしてプレビューを起動した場合は、指定内容に応じた読み替えが必要です。たとえば、上述の例のうち、最後に示した場合を見てください。

5) スペース, N, ROLL UP, ROLL DOWN 等のキーの項です。

List 3-2 ● 1 ページおきに表示する

```
1: preview foo 1--9
```

すでに述べたとおり、この例の場合は、1, 3, 5, 7, 9 ページが対象ですから、たとえば、今、5 ページ目を表示しているなら、次のページといえば、当然 7 ページ目のことを指します。

プレビューアを起動して、画面下部のファンクションキー表示が消えた時点から以下のキーが使えます。

←, H, CTRL + B, CTRL + S キー

画面上に表示されていない、紙面の左側を見るために、画面を左方向にスクロールさせます。左端まで画面をスクロールさせると止まります。SHIFT キーを併用すると、低速でスクロールします。

→, L, CTRL + F, CTRL + D キー

画面上では見えない、紙面の右側を見るために、画面を右方向にスクロールさせます。右端まで画面をスクロールさせると止まります。SHIFT キーを併用すると、低速でスクロールします。

↑, K, CTRL + P, CTRL + E キー

画面上では見えない、紙面の上側を見るために、画面を上方向にスクロールさせます。上端まで画面をスクロールさせると止まります。SHIFT キーを併用すると、低速でスクロールします。

↓, J, CTRL + N, CTRL + X キー

画面上では見えない、紙面の下側を見るために、画面を下方向にスクロールさせます。下端まで画面をスクロールさせると止まります。SHIFT キーを併用すると、低速でスクロールします。

スペース, N, ROLL UP, CTRL + V, CTRL + C キー

画面に次ページ上部を表示します。SHIFT キーを併用すると、次ページの、今、見ているページの場所と同じ位置に移ります。バッファを多くとっていて次のページの描画がすでに完了していれば、すぐにページが切り替わります。まだ完了していない場合は、ページ切り替えの要求があったことを記憶しておくだけです。

プレビューアは、何回、ページ切り替えの要求があったかを覚えています。キーを 2 回押せば即座に 2 ページ先の書き込みを始め、書き込みを終了すると、そのページの画面を表示します。キー入力の前に数字を指定すると、そのページ数分先の画面に移ります。たとえば、3 に続けて N キーを押すと、3 ページ先の画面に移ります。

B, ROLL DOWN, XF1 + V, XF2 + V, CTRL + R キー

画面に前ページ下部を表示します。SHIFT キーを併用すると、前ページの、今、見ているページの場所と同じ位置に移ります。バッファを多くとっている場合は、前に書き込んだ画面イメージがバッファに残っているため、すぐに表示することができます。もしバッファに残っていない場合は、即座にそのページの書き込みを始めます。書き込みを終了すると、そのページを表示します。この場合も、プレビューアは何回キーを押したかを覚えています。キー入力の前に数字を指定すると、そのページ数分前に移ります。

数字列 + リターン, 数字列 + P キー

直接閲覧したいページ数を指定することもできます。ページ数を入力し、最後にリターンキーか P キーを押します。この場合、指定した

ページのデータがバッファに残っていることは少ないので、プレビューアは、即座に現在の書き込みを中止し、指定されたページの書き込みを開始します。

W キー

ノーマルレゾリューション⁶⁾時には、より広範囲の画面イメージを見渡せるようにハイレゾリューションに切り替えることができます。ノーマルレゾリューションとハイレゾリューションはトグルになっていて、再度 W キーを押すと元のレゾリューションに戻ります。オプション“-highReso”を指定した場合との違いは、以下のとおりです。

方法	バッファサイズ	セル領域	リアルタイム切り替え
-highReso	可変	増える	不可
W キー	固定	同じ	可能

6) オプション“-highReso”を指定していない場合のこと。

ESC, Q キー

プレビューアでの表示を中止してコマンドモードに戻ります。

3.3.4 エラー対策

最後に、プレビューア実行中に起こり得るエラーについて、その対処法を簡単に説明します。

○ No Memory:

メモリが足りません。-GRAM⁷⁾ を使用してください。

それでも駄目な場合は、バッファの数を減らしてください。

○ Illegal Args:

コマンドラインでの引数の指定に間違いがあります。

○ No File:, No Font:

そのファイルを収めてあるディスクを正しいドライブにセットして、Y キーを押してください。

○ File Fault:, Font Mismatch:, Command Error:

dvi ファイル関係のエラーです。めったに出ることはないと思われます。dvi ファイルが壊れている可能性があります。もう一度 T_EX にかけて dvi ファイルをつくりなおしてください。

○ Unpack Error:

フォントファイルが pk フォーマットでない場合、このエラーが出ます。pk ファイルが壊れている可能性があります。METAFONT あるいは makefont.x を使って pk ファイルをつくりなおしてください。

○ Program Stop:

重大なエラーです。使用を即座に中止し、著者まで症状を報告してください。ただし、もう一度インストールしなおすと直ることがあります。

7) 『Vol.2 — Reference 編』の第 2.4 節「PRINT オプション」(p.54)を参照してください。

3.4……プリンタドライバ — プリントアウト

3.4.1 プリンタドライバの使用方法

いよいよプリントアウトです。まずは、第2章「Install」(p.21)に従って、あなたの使用するプリンタを指定して、そのプリンタ専用のシステムを構築してください。これを実行しておけば、プリンタの機種に依存するような設定は、今後行う必要はありません。

T_EX の HOME ディレクトリに、以下のファイルがあることを確認してください。

○ `print.cfg`

プリンタの出力シーケンスを記述するファイルです。ほかに、よく使うオプションを入れておくこともできます。

○ `print.p3m`

pk ファイル名を FONTMAN Ver.3 でのフォント名に変換する際の規則を入れておくファイルです。ほとんど変更する必要のないファイルです。

次に、プリンタが X680x0 本体とケーブルできちんとつながっていること、プリンタの電源が入っていること、そして、紙が正しくセットされていることを確認してください。

先ほど表示させた `preview.dvi` を印刷してみましょう。

```
A> print preview
```

`preview.tex`, `print.tex` は、それぞれ、その `dvi` ファイルをプリントアウトして保存しておくのがいいでしょう。

プリンタドライバは、プレビューア同様、オプションを指定して起動することができます。詳しくは、『Vol.2 — Reference 編』の第2.4節「PRINT オプション」(p.54)を参照してください。

プリンタドライバはいくつかの環境変数と、いくつかのファイルを参照します。詳しい説明は『Vol.2 — Reference 編』の第1.4節「PRINT 環境変数」(p.12)にあります。

なお、プリントアウトを中止してコマンドモードに戻るときには ESC キーを押してください。軽くキーに触れる程度で、プリンタドライバが終了します。

3.4.2 自分のプリンタに対応させる

もし、あなたのお持ちのプリンタに本 T_EX システムが対応していない場合は、オプションをいくつかセットすることによって、あなたのプリンタに対応させることができます。オプションセットのサンプルは、T_EX の HOME ディレクトリのサブディレクトリ `drivers\configs` 内に何種類か入っています。

```
A> cd %TEXHOME%\drivers\configs
A> dir
ボリュームがありません a:\usr\local\TeX\drivers\configs
      42 ファイル      4693K Byte 使用中      3450K Byte 使用可能
ファイル使用量      42K Byte 使用
AP550.cfg              175  91-08-12  22:14:22
AP850.cfg              203  91-08-12  22:40:28
AP900.cfg              200  93-10-24  19:21:32
BJ10V.cfg              230  91-08-12  22:33:24
BJ130J_1.cfg           323  90-11-11  15:00:00
      :
      :
PR2KA4.cfg             421  93-03-05  14:02:04
TR24CL.cfg             215  91-08-17  12:03:50
VP2000.cfg             223  90-11-09   8:21:38
VP800.cfg              177  90-01-02  15:00:00
VP950.cfg              240  90-01-02  15:00:00
```

この中に、あなたがお持ちのプリンタの型番と類似した名前のファイル¹⁾が見つかれば、それを T_EX の HOME ディレクトリ内に `print.cfg` という名前でコピーします。たとえば、M1024 が類似している場合は、以下のように指定します。

```
A> copy M1024.cfg ..\..\print.cfg
```

もし T_EX の HOME ディレクトリのサブディレクトリ `drivers\configs` 内に類似プリンタがない場合は、コンフィギュレーションファイル `print.cfg` を自分でつくらなければなりません。この作り方については第 5.3 節「`print.cfg` の作り方」(p.186)、および『Vol.2 — Reference 編』の第 3.1 節「コンフィギュレーションファイル文法」(p.100)を参照してください。

1) 本書の添付ディスクに付属のインストーラを使用した場合に、対応プリンタの一覧として表示されるのが、このディレクトリ内のファイル名です (p.36 参照)。

3.4.3 起動方法

プリンタドライバの起動方法は、以下のとおりです。

```
print [オプション] foo [オプション] [ページ] [オプション]
```

[オプション] には、『Vol.2 — Reference 編』の第 2.4 節「PRINT オプション」(p.54)で説明しているオプションを書きます。書く場所は、コマンドラインの前のほうでも後ろのほうでもかまいません。

foo には、プリントアウトしたい dvi ファイル名を書きます。なお、拡張子の .dvi はつけなくてもかまいません。

最後の [ページ] は、プリントアウトしたいページ範囲を指定するものです。また、不要なら省略することもできます。ページ番号の範囲を指定しますが、これはフッタに表示されているページ番号とは違うことがあります。virtex.x によってソースを処理する際に画面に表示されたページ番号です²⁾。

ページ指定のサンプルをいくつか挙げて説明しましょう。

- print foo
プリントアウトしたい dvi ファイル foo のすべてのページを出力します。
- print foo 3-10
3 ページから 10 ページまでを出力します。
- print foo 1-4 16-20 30-
1 ページから 4 ページまでと、16 ページから 20 ページまで、そして 30 ページから最後までを出力します。
- print foo 1--9
1 ページから 9 ページまでを、1 ページおきに出力します。つまり、1, 3, 5, 7, 9 ページが対象となります。ページ指定が "--" となっていることに注意してください。

これらの指定はプレビューアのページ指定方法と同じです。

3.4.4 エラー対策

最後に、プリンタドライバ実行中に起こり得るエラーについて、その対処法を簡単に説明します。

- No Memory:
メモリが足りません。-GRAM, -TRAM を指定してください³⁾。
Human68k のデバイスドライバで使用頻度の低いものを外して X680x0 を起動しなると少し状況がよくなります。それでもメモリが足りない場合は、メモリボードを購入し、メモリを増設してください。

2) たとえば本書の形式上の第 1 ページの場合、このページはまえがきや目次のあとにありますから、実質的な第 1 ページではありません。virtex.x がソースを処理する際に画面に出力し、デバイスドライバが「ページ番号」として採用しているのは、ここでいう「実質的な」ページ番号です。

3) 『Vol.2 — Reference 編』の第 2.4 節「PRINT オプション」(p.54)を参照してください。

- **Illegal Args:**
コマンドラインでの引数の指定に間違いがあります。マニュアルをよく読んで、正しく引数を指定してください。
- **No File:, No Font:**
そのファイルを収めてあるディスクを正しいドライブにセットして、Y キーを押してください。
- **File Fault:, Font Mismatch:, Command Error:**
dvi ファイル関係のエラーです。ほとんど出ることはないと思われます。dvi ファイルが壊れている可能性があります。もう一度 \TeX にかけて dvi ファイルをつくりなおしてください。
- **Unpack Error:**
フォントファイルが pk フォーマットでない場合、このエラーが出ます。そうでない場合には、pk ファイルが壊れている可能性があります。METAFONT あるいは makefont.x を使って pk ファイルをつくりなおしてください。
- **Program Stop:**
重大なエラーです。使用を即座に中止し、著者まで症状を報告してください。ただし、もう一度インストールしなおすと直ることがあります。

3.5 ファックスファイル変換

最近モデムがとても安くなり、9600bps や 14400bps のモデムが個人でも買える値段になってきました。そして、9600bps 以上のモデムを買うと、たいてい、ファックス機能がついています。これを利用して T_EX ファイルをファックス経由で送ろうというのが本節の目的です。

さすがにユーザの絶対数の違いというわけではないでしょうが、MS-DOS マシンを見てみると、T_EX ファイルをファックスで送信できる形式に変換するプログラムとして、anti-top 氏および SHIMA 氏が作成した PBM2FAX.EXE がありますし、逆にファックスで受信したファイルを T_EX ファイル形式に変換するプログラムには、SHIMA 氏が作成した FAX2PBM.EXE があります。

これらを SSIZZ¹⁾氏が X680x0 用に移植したのが tt12fax と fax2ttl です。これらは、アーカイブファイル tt12fax.lzh および fax2ttl.lzh という名前で本書添付ディスクの 8 に収録されています。詳しい説明は、そのアーカイブに含まれたドキュメントに譲るとして、本節では、その使い方を中心に説明していきたいと思います。

なお、当たり前といえば当たり前ですが、T_EX ファイルをファックスで送受信するには、本書添付ディスクに収録されている既述のプログラムのほかに、ファックスモデム、そして、ファックス送受信プログラムを用意しなければなりません。注意してください。

また、ファックスモデムとその送受信のしかたについては、詳説する紙面の余裕がありませんので省略し、ここでは T_EX ファイルと Starfax 形式のファイルとの間の変換だけに話を限定して説明することにします。

同様の理由から、ファックス送受信プログラムについても深く触れることはしません。ただ、ファックス送受信プログラムでは、canalian²⁾氏の“FAXION 大魔王”や、SSIZZ 氏の“LESQUA” (レスカ … 原作は MS-DOS 版で、原作者は護法童子³⁾氏) などが有名です。これらはいずれも NIFTY-Serve で手に入れることができます。

ただ、ファックスモデムには、ファックス ↔ モデム間、およびモデム ↔ コンピュータ間の相性の問題がかなりあります。しかも、ファックス送受信のしかたは必ずしも簡単ではありません。これについては、手持ちのファックスモデムのマニュアルなどを参照してください。

1)NIFTY-Serveでの ID
は PFA00204 です。

2)NIFTY-Serveでの ID
は PED00445 です。

3)NIFTY-Serveでの ID
は PFA01121 です。

① ファックスモデムと \TeX

普通、家庭用ファックスといえば、スキャナとプリンタとモデムが一体になったものを指しますが、ここでいうファックス機能とは、家庭用ファックスからスキャナとプリンタ部を取り除いたものを指します。したがって、ファックス機能を持ったモデムを利用するのであれば、スキャナとプリンタが別途必要になります。そして、これら 3 つを相互接続するためには、もちろん、コンピュータも必要です。

しかし、本書の読者は、 \TeX を使おうとしているのですから、コンピュータはすでに持っているはずです。

また、 \TeX の文章をファックスで送れる

環境にあるならば、スキャナは必須ではなくなります。

さらに、 \TeX システムを持っていれば、プリンタすら不要になります。 \TeX ユーザならたいていプリンタを持っているでしょうが、たとえ何かの都合でプリンタが使えないとしても、プレビューアがあれば受信したファックスデータを見ることができるからです。

スキャナとプリンタがないために、単にモデムとして使用していたファックスモデムが、 \TeX システムと結び付くことで、ファックスとしての用途が開けてくるのです。

3.5.1 ファックス送受信の手順

\TeX ファイルをファックス送信する場合は次のような手順で行います。

- (1) dvi ファイルをファックスフォーマットのファイルに変換する。
- (2) 変換したファイルをファックス送信プログラムで送信する。

ファックス受信する場合は次のような手順で行います。

- (1) ファックス受信プログラムで受信する。
- (2) ファックスフォーマットのファイルをタイトルファイル (title.sys) 形式に変換する。
- (3) プレビューアやプリンタドライバのタイトルファイル読み込み機能を使って画面表示したり、プリンタ出力したりする。

ここで、一口に「ファックスファイルのフォーマット」といっても、これには各種ソフトによってさまざまな形式が採用されています。ここで、そのすべてを説明することはできませんので、本書では、ファックスファイルのフォーマットとして代表的な「Starfax 形式」を取り上げることにします。

3.5.2 \TeX のファイルを Starfax 形式へ変換する

まず、 \TeX の dvi ファイルを Starfax 形式のファイルに変換してみましょう。

そのために必要なツール類は、添付ディスク 8 に収録されている 2 つのアーカイブファイル tex2ttl.lzh と ttl2fax.lzh のなかに入っていますので、最初にこれらのアーカイブファイルを展開してください。アーカイブファイルの展開に LHA.x を使った場合は、以下のように行います。

```

A> lha x tex2ttl.lzh

Extracting from Archive : tex2ttl.lzh

TeX2TTL.DOC          : 解凍終了    [    2210 ] (    2210 )
TeX2TTL.X            : 解凍終了    [   17438 ] (   17438 )
TeX2TTL.CFG          : 解凍終了    [     432 ] (     432 )
mf208.tar            : 解凍終了    [   30720 ] (   30720 )

A> lha x ttl2fax.lzh

Extracting from Archive : ttl2fax.lzh

PBM2FAX.DOC          : 解凍終了    [    5521 ] (    5521 )
README.ADD           : 解凍終了    [    2023 ] (    2023 )
TTL2FAX.X            : 解凍終了    [   22934 ] (   22934 )
README.68            : 解凍終了    [     805 ] (     805 )

A>

```

ここで、展開されたファイルについて簡単に説明しておきます。

tex2ttl.lzh には、TeX2TTL.DOC、TeX2TTL.CFG、TeX2TTL.X、mf208.tar という 4 つのファイルがアーカイブされています。

○ TeX2TTL.DOC

このファイルは、tex2ttl のマニュアルです。本節のかわりに、こちらを読んでもらってもかまいませんが、若干注意点があります。tex2ttl は特殊なフォントを使用する都合上、このドキュメントファイルの冒頭で「フォント作成」についての説明があります。しかし、本書の添付ディスクによって構築した \TeX システムと、このドキュメントが想定する \TeX のシステム環境とでは、ディレクトリやファイル類の構成が違うので、「フォント作成」の説明に書かれているとおりに実行しても正しく動きません。「フォント作成」方法については本項で述べますので、こちらを参照してください。

○ TeX2TTL.CFG

このファイルは、プリンタドライバ用のコンフィギュレーションファイルです。あとで説明しますが、このコンフィギュレーションファイルを指定してプリンタドライバを起動すると、プリンタドライバはビットマップイメージをプリンタに出力するかわりにファイルに出力するようになります。

○ TeX2TTL.X

ダンプしたファイルをタイトルファイル形式に変換するプログラムです。

○ mf208.tar

208dpi の METAFONT を作成するための METAFONT ソース (の一部) です。makefont.x を使用すれば任意サイズのフォントを作成することができますから、これを使用する必要はないでしょう⁴⁾。

4) しかし、METAFONT のユーザインターフェースにバッチファイルが用いられていた頃には、きわめて有用なものでした。

次に `ttl2fax.lzh` には、`PBM2FAX.DOC`、`README.ADD`、`README.68`、`TTL2FAX.X` の 4 つのファイルがアーカイブされています。

○ `PBM2FAX.DOC`

このファイルは `ttl2fax` のオリジナル版である `PBM2FAX` のマニュアルです。
X680x0 版とは内容的に大きく異なっていますので、参考程度に目を通して
おいてください。しかし、一度はしっかり読んでおいたほうがいいでしょう。

○ `README.ADD`

このファイルも `PBM2FAX` 関係です。参考程度に目を通せば十分ですが、一度
はしっかり読んでおくべきでしょう。

○ `README.68`

`ttl2fax` のマニュアルです。

○ `TTL2FAX.X`

`PBM2FAX.EXE` の X680x0 移植版の実行プログラムです。

ファイルの説明は以上で終わりです。`TeX2TTL.X` と `TTL2FAX.X` をパスの通った
ディレクトリのなかに収めておいてください。

◆ フォントの作成

さて、先のアーカイブ内のファイルについての説明で、`tex2ttl` は特殊なフォ
ントを使用するのだと述べましたが、実際のところは特殊といってもそれほど大
層なものではありません。ファックスの基本解像度が、プレビューアで使用する
ディスプレイ用のフォント、あるいはプリンタドライバで使用するプリンタ用の
フォントとは、若干異なる値だということだけのことです。

すでにご存じの読者もいるかと思いますが、ファックスは横 208 dpi × 縦 196
dpi という解像度になっています。`TeX` は縦横同じ dpi しか扱えませんが、多
少縦方向に伸びるのは我慢して、208 dpi のフォントを作成することにします。

そこでまず、このフォントの作成を行わなければなりません。

後出の第 5.4.2 項「`makefont` によるフォント作成」(p.220)を参考にして 208
dpi のフォントを作成するのですが、プリンタドライバ実行時点で出力される足
りないフォントのレポートを見ながら、そのつどフォント作成を行うほうが効率
的でしょう。

◆ ファイル形式の変換

いよいよ Starfax 形式ファイルへの変換作業に入ります。この作業は、いった
ん `dump` ファイルを作成し、その後、タイトルファイルに変換するという、2 つ
のステップを踏みますので、注意しながら作業をしてください。

まず、プリンタドライバ起動用にコンフィギュレーションファイルの設定を行います。tex2ttl.lzh を展開して得られたファイル TeX2TTL.CFG を \TeX の HOME ディレクトリにコピーします。

```
A> copy TeX2TTL.CFG %TEXHOME%
```

そして、環境変数 PRINT.CFG を設定します。

```
A> set PRINT.CFG=%TEXHOME%/TeX2TTL.CFG
```

以上で準備は完了しました。

いよいよ \TeX の dvi ファイル形式から Starfax 形式への変換作業に入ります。ここでは、変換したい dvi ファイル名が foo.dvi であるとしましょう。

プリンタドライバを、通常使用するのと同様に、foo.dvi を引数にして起動します。

```
A> print foo
```

すると、カレントディレクトリに TeX2ttl.dmp というファイルができます。今度は、このファイルを引数にして、TeX2ttl.x を起動してください。

```
A> TeX2TTL TeX2ttl.dmp
```

すると、カレントディレクトリに dump01.ttl, dump02.ttl, ... というファイルが、ページ数分だけ作成されます。dump に続く 01, 02, ... がページ数を示しています。

たとえば、4 ページ分のファイルが作成されたとしましょう。これらのファイルを Starfax 形式のファイルに変換します。

```
A> TTL2FAX foo.fax dump01.ttl dump02.ttl dump03.ttl dump04.ttl
```

これで、Starfax 形式のファイル foo.fax が作成されます。

本節の冒頭でも触れましたが、このファイルは “FAXION 大魔王” や “LESQUA” を使って、ファックス送信することができます。たとえば、LESQUA を使って電話番号 “012-345-6789” にファックス送信する場合、次のようにしてください。

```
A> le -sATD0123456789 foo
```

3.5.3 Starfax 形式ファイルを TeX へ変換する

今度は逆に、“LESQUA” などを使って受信したファックスデータをデバイスドライバで出力してみましょう。そのために必要なツール類は、本書の添付ディスク 8 中のアーカイブファイル `fax2ttl.lzh` に含まれています。

まず最初に、`fax2ttl.lzh` を展開してください。

```
A> lha x fax2ttl.lzh

Extracting from Archive : fax2ttl.lzh

FAX2PBM.DOC      : 解凍終了 [ 5521 ] ( 5521 )
README.68       : 解凍終了 [ 1246 ] ( 1246 )
FAX2TTL.X        : 解凍終了 [ 26466 ] ( 26466 )
FAX2TTL.TEX      : 解凍終了 [ 1869 ] ( 1869 )
FAX2TTL.DVI      : 解凍終了 [ 3208 ] ( 3208 )
FAX2TTLA4.DVI   : 解凍終了 [ 3568 ] ( 3568 )
FAX2TTLB4.DVI   : 解凍終了 [ 3360 ] ( 3360 )

A>
```

展開してできあがったファイルは、`FAX2PBM.DOC`、`README.68`、`FAX2TTL.X`、`FAX2TTL.DVI`、`FAX2TTLA4.DVI`、`FAX2TTLB4.DVI`、`FAX2TTL.TEX` の 7 つです。

- `FAX2PBM.DOC`
このファイルは、`fax2ttl` のオリジナルである `FAX2PBM` のマニュアルです。
`fax2ttl` を使う場合は、参考程度に目を通しておいってください。
- `README.68`
このファイルは `fax2ttl` のマニュアルです。
- `FAX2TTL.X`
Starfax 形式ファイルをタイトルファイルに変換するプログラムです。
- `FAX2TTL.DVI`、`FAX2TTLA4.DVI`、`FAX2TTLB4.DVI`
これらのファイルは、`FAX2TTL.X` で変換したタイトルファイルを、プリンタドライバで出力するための `dvi` ファイルです。各ファイルを使うと、どのように出力されるのかについては、のちほど触れます。
- `FAX2TTL.TEX`
直前に示した 4 つの `dvi` ファイルのもとになったソースです。

◆ プリンタドライバでの出力

さて、それでは Starfax 形式から、デバイスドライバで出力できるタイトル

ファイル形式への変換作業に入りましょう。データ形式を変換したい Starfax 形式のファイル名を、たとえば `foo.fax` とします。このファイルを `FAX2TTL.X` で処理します。

```
A> FAX2TTL foo.fax
```

すると、カレントディレクトリに `fax01.ttl`, `fax02.ttl`, ... というファイルがページ数分だけ作成されます。`fax` に続く `01`, `02`, ... がページ数を示します。

たとえば、この処理によって 4 ページ分のファイルが作成されたとしましょう。これをプリンタに出力するには、先ほど `fax2ttl.1zh` に含まれていた 4 つの `dvi` ファイルをプリンタドライバで処理します。

○ FAX2TTLA4.DVI

この `dvi` ファイルを 360 dpi プリンタで出力すると、A4 サイズのファックスが、そのままのサイズで出力されます。

```
A> print FAX2TTLA4 1-4
```

また、B4 サイズのファックスなら、そのまま B4 サイズで出力されます。

○ FAX2TTLB4.DVI

この `dvi` ファイルを 360 dpi プリンタで出力すると、B4 サイズのファックスが A4 サイズに縮小されて出力されます。

```
A> print FAX2TTLB4 1-4
```

他のサイズのファックスでも同じ縮小率 (81.7%) で出力されます。

○ FAX2TTL.DVI

この `dvi` ファイルで出力すると、ファックスの元の解像度のままで出力されます。ファックスの解像度は 208 dpi です。たとえば 360 dpi のプリンタなら 0.58 倍に縮小されたイメージで出力されます。紙の大きさは不自然ですが、ドットの拡大・縮小は行われていないので、きれいに出力することができます。

```
A> print FAX2TTL 1-4
```

なお、360 dpi でない他のプリンタに出力したり、画面に出力したりする場合は、`FAX2TTL.TEX` を修正して専用の `dvi` ファイルを作成すれば、正しいサイズで出力できるようになります。

たとえば、A4 サイズのファックスを 180 dpi のプリンタに A4 サイズで出力するためには、次のようにします。

まず、FAX2TTL.TEX を A4to180A4.tex というファイル名でコピーします。

```
A> copy FAX2TTL.TEX A4to180A4.tex
```

次に、A4to180A4.tex をエディタで修正します。

```
A> ed A4to180A4.tex
```

List 3-3 ● A4to180A4.tex の一部

```
29: % 1dot to 1dot
30: %\def\ttl#1{\special{!title #1 scaled 0.5 }\nextpage}
31: % a4 fax to a4 360dpi
32: %\def\ttl#1{\special{!title #1 scaled 0.8333333333333333 }\nextpage}
33: % b4 fax to a4 360dpi
34: \def\ttl#1{\special{!title #1 scaled 0.703125 }\nextpage}
```

この 31 行目から 34 行目を次のように修正してください。

List 3-4 ● A4to180A4.tex の修正結果

```
29: % 1dot to 1dot
30: %\def\ttl#1{\special{!title #1 scaled 0.5 }\nextpage}
31: % a4 fax to a4 180dpi
32: \def\ttl#1{\special{!title #1 scaled 1.6666666666666666 }\nextpage}
33: % b4 fax to a4 180dpi
34: %\def\ttl#1{\special{!title #1 scaled 1.40625 }\nextpage}
```

32 行目と 34 行目は、タイトルファイルの拡大率を指定するものです。scaled の直後の数値が拡大率で、180dpi プリンタの場合、これを 1.0 にすると、拡大・縮小は行われません。また、360dpi プリンタで 180dpi プリンタの出力結果と同じにするためには、ここを半分の数値にしなければなりません。

修正したものを \LaTeX にかけます。

```
A> latex A4to180A4
```

無事にコンパイルが終了すれば、A4 ファックスを 180 dpi プリンタに A4 サイズで出力するための専用 dvi ファイル A4to180A4.dvi が得られます。

次のようにすれば、プリンタに出力できます。

```
A> print A4to180A4 1-4
```

これでファックスで受信したデータを \TeX で表示できるようになりました。

最近、ファックスを使ったショッピングやイベント等の情報サービスが増えてきています。普通のファックスですと、実際に紙に印刷しないと見ることはできません。しかし、 \TeX システムでなら、プレビューアを使って画面上で閲覧するだけという活用方法も考えられます。このようにすれば、無駄に紙を消費することなく、不要な情報だと思えば、これを捨てることもできます。

また、自分のファックスにいたずらファックスを送り付けられて無駄にファックス用紙を使ってしまったという声を聞いたことがあります。けれども、ファックスモデムと \TeX システムの組み合わせなら、そのような心配はありません。いたずらファックスはファイル上で消してしまえばいいのです。

このようなことが、ファックスモデムと、“LESQUA”等のファックス送受信ソフト、および \TeX システムがあるだけで実現できます。あなたも、 \TeX とファックスの新しい可能性を体験してみてください。

Exercise

前章までに X680x0 版の $\text{T}_{\text{E}}\text{X}$ について解説し、そのシステムを構築してきました。すると、次は $\text{T}_{\text{E}}\text{X}$ を使用することで実際にどんな出力が得られるのかを試してみたくなったり、さらには自分で書いた文章を出力してみたくなったりするのではないのでしょうか。本章では、数多の $\text{T}_{\text{E}}\text{X}$ のなかでも最も一般的に使用されている $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ を使って、その処理およびソースの作成について解説してみることにします。

4.1 はじめに

第 1.2.3 項「 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ の種類と実行ファイル」(p.10) で述べたように、 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ には、基本となる“データベース”の違いによって、いくつかの種類があります。第 6 章「 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ family」(p.229) で紹介する、 $\mathcal{A}\mathcal{M}\mathcal{S}-\mathrm{T}_{\mathrm{E}}\mathrm{X}$ や $\mathrm{P}_{\mathrm{I}}\mathrm{C}_{\mathrm{T}}\mathrm{E}_{\mathrm{X}}$, $\mathrm{M}_{\mathrm{u}}\mathrm{s}\mathrm{i}\mathrm{c}_{\mathrm{T}}\mathrm{E}_{\mathrm{X}}$ などがその例です。本章では、このようにいくつかある $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ のなかで最もよく用いられている $\mathrm{L}_{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ を例にとって、その処理の実際について解説します。

その解説では、特に $\mathrm{L}_{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ のソース作成の段階で実際の $\mathrm{L}_{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ ソースの作成のノウハウについて触れている部分がたくさんあります。しかし、本書は $\mathrm{X}_{680}\mathrm{x}0$ 版 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ の実行環境を提供することに主眼を置いていますから、実際のソースの作成について、あまり詳しく解説しているわけではありません。必要最小限、つまり、簡単なレポートであれば作成できる程度にとどめています。

もし、本章で述べている以上のことを知りたい、あるいは行いたいということであれば¹⁾、第 6 章「 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ family」(p.229) で紹介している参考文献を参照してください。また、今回は特に鈴木裕信氏のご好意により、第 6 章「 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ family」の $\mathrm{L}_{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ の解説部分 (p.233) でも紹介している『ひろのぶの $\mathrm{L}_{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ 入門』²⁾ を添付ディスク 8 に収録することができましたので、こちらも参考にしてください。

なお、本章に記述されている内容は、 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ システムのなかでも、アスキーの日本語 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ および 縦組対応日本語 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ ($\mathrm{p}_{\mathrm{T}}\mathrm{E}_{\mathrm{X}}$ ³⁾) に特化したものです。したがって、D. E. Knuth 博士が開発したオリジナルの $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ や、NTT- $\mathrm{j}_{\mathrm{T}}\mathrm{E}_{\mathrm{X}}$ などに對して普遍的に適用できるものではありませんから、その点に関してはご注意ください⁴⁾。

1) $\mathrm{L}_{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ を本気で使う気になれば、当然そうなるはずですし、そうならなくても困ります。

2) 『やさしい $\mathrm{L}_{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ のはじめかた』(鈴木裕信著 オーム社刊 1991 年) のもとになったドキュメント。

3) $\mathrm{p}_{\mathrm{T}}\mathrm{E}_{\mathrm{X}}$ とは「publish (出版) $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ 」を意味していて、縦組と横組が混在する日本語特有の文書作成が可能ないように日本語 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ を拡張したものです。日本語 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ に対して上位互換の $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ といえます。

4) オリジナルの $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ および NTT- $\mathrm{j}_{\mathrm{T}}\mathrm{E}_{\mathrm{X}}$ 、アスキーの (縦組対応) 日本語 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ 間の相違点については、アスキーから刊行されている各種文献に特に詳しく書かれていますから、参照してください。

4.2L^AT_EX で遊ぶ

前章までの解説で、読者の皆さんは、すでに X680x0 版の T_EX システムをインストールし、インストールのチェックもかねて、いくつかのサンプルを出力していることと思います。そこで、本節ではすでに T_EX のインストールが終了しているものとして話を進めますから、まだインストールをしていない人は、先にインストールをすませておいてください。

4.2.1 L^AT_EX とは

まず、本章が対象として扱う L^AT_EX について簡単に説明しておきましょう。

L^AT_EX とは、Digital Equipment Corporation (DEC) 社の L. Lamport 氏が UNILOGIC 社の文書整形システム Scribe を参考に作成したもので、数多の T_EX ファミリー¹⁾ のなかでも最も代表的といえる T_EX です。

文章や数式の処理のみならず、表や簡単な図をかくためのマクロやフォントも用意されており、使い勝手がたいへんよくなっています。また、「スタイルファイル」と呼ばれる文書の体裁を定めた独特のファイル概念を導入したことで、異なる体裁の文書を簡単に作成できることも見逃せません。さらに、目次・相互参照・文献表の作成などの処理を、ほぼ自動的にしてくれることも特徴といえるでしょう。

したがって、L^AT_EX は、手紙やレポート、書籍といった、ある程度まとまった量でまとまった体裁の文章を書こうというときにこそ、力を発揮してくれます²⁾。

そのため、現在、T_EX のソースファイルといえば、ほとんどが L^AT_EX 用に作成されているといっても過言ではありません。実際、T_EX を出版に利用する場合も、ほとんどが L^AT_EX をもとに行われているようです³⁾。

さて、「L^AT_EX が非常に使いやすいものであるらしい」ことはわかっていただけたのではないかと思います。それでは、次項からは実際に L^AT_EX を使用してみることしましょう。

1) 第6章「T_EXfamily」
(p.229) で紹介します。

2) たとえばカセットレベルであるとか、ちょっとした文書程度であれば、むしろワープロなどを使用したほうが手早く作成することができるでしょう。

3) 本書もそのうちのひとつといえます。

4.2.2 L^AT_EX を使ってみる

L^AT_EX 自体は、第 2.2.5 項「インストールチェック」(p.39)のインストールチェックで問題が起きなかったならば、動作する環境が整っているはずです。失敗している場合には、残念ですが、もう一度第 2 章「Install」(p.21)から始めてください。

◆ L^AT_EX の処理

第 1.2.2 項「T_EX の処理の流れ」(p.6)で説明しましたが、L^AT_EX も基本的に T_EX の処理と同じ手順をふみます。

(1) 原稿となるファイルを作成する。

原稿となるファイルを、テキストエディタで作成します。

(2) L^AT_EX で dvi ファイルに変換する。

L^AT_EX で処理し、dvi ファイルに変換します。

(3) dvi ファイルをプレビューする。

指定どおり、あるいはイメージどおりに文書が作成されているかどうかをディスプレイで確認します。そうになっていなければ、修正箇所を見つけて、気に入った出力になるまでソースの編集、L^AT_EX での処理を繰り返します。

(4) プリンタで印刷する。

指定どおりの文書が作成できたら、実際にプリンタで印刷します。

それではこれから、L^AT_EX の文書を出力するところまで、ひととおり実際に過程を追ってみることにしましょう。

◆ L^AT_EX ソースの作成

4) ここでは「そのまま L^AT_EX の処理にかけることができる」という意味です。

まずは、「原稿となるファイルの作成」です。

実際にはこの段階で、あなた自身が「原稿となるファイル」を一からつくらなければなりません。しかし、残念ながら、今の段階で L^AT_EX のソースファイルを自力で作成してもらうのは無理でしょうから、「原稿となるファイル」を一からつくっていただく過程については後述することにします。

かわりにここでは、完結した⁴⁾標準的 L^AT_EX のソースファイルを以下に提示しますから、これを「原稿となるファイル」として利用していただくことにしましょう。

なお、先に述べたように L^AT_EX はある程度まとまった規模の文書を作成するときこそ真価を発揮してくれますから、ここでも少し長めの文書を例にします (List 4-1)。このソースを、テキストエディタを使って x68.tex のファイル名で作成してください。

テキストエディタの使用法については、あなたが使用するテキストエディタのマニュアルを参照してください。

List 4-1 ● サンプルソース

```

1: %_文書のスタイルを指定します。
2: \documentstyle[b5j]{jarticle}
3:
4: %_タイトルを定義します
5: \title{X680x0_について}
6: \author{Chikumi_Yoshino}
7: \date{1993/12/27}
8:
9: %_ここから実際に文書が始まります
10: \begin{document}
11: %_タイトルを出力します。
12: \maketitle
13: %%%_↓_jreport_,_jbook_のとき、コメントアウトしてください。
14: %\chapter{X680x0_の今後}
15: \section{X680x0_の現状}
16: %_参照用のラベル
17: \label{1st_section}
18: 1993_年には、待望久しかった_X68030_が発売されました。X68030_は
19: X68000_シリーズの最高峰として存在するマシンですが、基本的な設計においては
20: 従来機種との違いはなく、単に_CPU_をより高速な_MC680EC30_に置き換えて
21: バス幅を_32bit_に拡張しただけのものとして考えることもできます。
22: そういった観点からは、この_X68030_はなんら新しい部分がない魅力のない
23: 機械であるといった批判を免れないのは事実です。
24:
25: %_改行が段落になります。
26: しかしながら、ベースマシンである_X68000_のグラフィックやサウンド機能は
27: 現在の他の機械に比べて著しく劣ると思われる部分はなく、若干の互換性の
28: 問題はあるものの、従来機種で動いていたソフトウェアをほぼそのまま高速に
29: 実行できる点だけでも評価できると思われます。CPU_がソケットに実装されて
30: いる事実や、新しく書き換えられた_IOPS_ROM_が_CPU_の種別を判別している
31: 点など、相変わらずマニアックな設計思想はそのまま受け継がれており、インテ
32: ル一色のパソコン世界において_Mac_とともに\footnote{市場規模では_Mac_の
33: ほうが大きいといえますが。}異色の存在であるのは事実でしょう。
34:
35: 今後の展開がいまひとつ不明なので将来性については若干の不安があるものの、
36: 熱心なマニアの手によってさまざまなソフトウェアが開発されていますので、
37: それらを入手できる環境であれば、なかなかおもしろい機械であると思われます。
38:
39: \section{ソフトウェア (System)}
40:
41: %_参照をします
42: \ref{1st_section}節で述べたように、新しい_X68030_発売にともなって
43: 従来機種で使われていた_Human68k_も_Ver_3.0_になりました。Human68k_は
44: %_‘_’ は並べて書くことで長さが変化する
45: _MS--DOS_に類似したコマンド形態をもつ独自の_DOS_です。
46: PC--9800_の_MS--DOS
47: フロッピーディスクをそのまま読み取り、書き込み可能な点等からも
48: MS--DOS_との類似性は非常に高いのですが、X680x0_ではメモリ空間が最初から
49: 16M バイトフラットアドレッシングであるために、メモリの扱い
50: が_MS--DOS_に比べて圧倒的に簡便になっています。
51: このために_MS--DOS_では直接動かせない
52: 大規模なソフトウェアも動かすことができ、UNIX_上の
53: 大規模ソフトウェアが比較的前から移植され、動作していたのが
54: 特徴でもありました。
55:
56: 残念ながら、X68030_に標準搭載されている_CPU_が_MMU_\footnote{Memory_Manag
57: erement_Unit_の略。}
58: を持たない_MC680EC30_であるためにいまだに

```

```

58:  仮想記憶といったものは使えません。
59:
60:  \section{アプリケーションソフトウェア}
61:
62:  X680x0シリーズには、昔から「ビジネスソフトウェア」はあまり発売されて
63:  いませんでした。1993年ににおいても目立った実用的なソフトウェアは
64:  発売されていません。
65:  半面、「開発用ツール」や「グラフィックエディタ」、「ゲーム」といった
66:  類の趣味的な要素の強いソフトウェアは、
67:  市場規模が比較的小さいにもかかわらず、
68:  それなりの「売り上げ」をあげるといった特徴もあります。
69:  しかし、最近は、ゲームソフトウェアを開発していたかなりの数の
70:  ソフトウェアハウスが撤退をしたのは事実といえます。
71:  雑誌等で「X68000衰退論」が論じられたのはご承知の
72:  とおりです。
73:
74:  ただ、実際に1993年に発売されたゲームソフトウェアのレベルは非常に
75:  高く、「少数精鋭、買って損なし」といった状態といえます。
76:  実購入数の低下さえクリアできれば、1994年にはある程度の数が期待できる
77:  かもしれません。SX-WINDOWについては、純正の開発環境の提供が著しく
78:  遅れているので、若干不安要素が残っている点を指摘するのみにと
79:  どめておきます。
80:  %本文の終わり
81:  \end{document}

```

4.2.3 L^AT_EX による処理

ソースができたなら、次は「L^AT_EX で dvi ファイルに変換する」、つまり L^AT_EX による処理を行います。

◆ サンプルソースを処理する

第 2.2.5 項「インストールチェック」(p.39)のインストールチェックで L^AT_EX が動作することは確認していますから、先に示したサンプルソース List 4-1 を、L^AT_EX で処理してみましょう。

```

A> virtex "&jlplain \input x68.tex"
This is pTeX, C Version 2.99 j1.7 p1.0.9F EW (no format preloaded)
LaTeX Version 2.09 <24 May 1989>
(x68.tex (c:/tex/jmacros/jarticle.sty
Document Style 'jarticle' <18 Dec 88>.
(c:/tex/jmacros/jart10.sty)) (c:/tex/jmacros/b5j.sty)
No file x68.aux.
LaTeX Warning: Reference '1st_section' on page 1 undefined.
[1]
Overfull \hbox (7.09404pt too wide) in paragraph at lines 56--59
[]\tenmin 残念ながら、\tenrm X68030 \tenmin に標準搭載されている
\tenrm CPU \tenmin が \tenrm MMU []\tenmin を持たない \tenrm MC680
EC30
[2] (x68.aux)
LaTeX Warning: Label(s) may have changed.
Rerun to get cross-references right.
(see the transcript file for additional information)
Output written on x68.dvi (2 pages, 6560 bytes).
Transcript written on x68.log.
A>

```

L^AT_EX で文書进行处理する場合には、この例のようにコマンドラインから、

```
virtex &jlplain inputfile
```

とタイプします。しかし、毎回このようにタイプするのは面倒です。そこで、あなたが本書の添付ディスクによってインストールした T_EX システムにはこの設定を記述したバッチファイルが用意されています。コマンドラインから、

```
LaTeX inputfile
```

と入力してください。

ところで、L^AT_EX を起動させると、画面上に c:/TeX/jmacros/jarticle.sty などという表示⁶⁾ が出力されますが、この表示は各自の構築した L^AT_EX のシステム構成や処理を行う L^AT_EX ソースの違いによって変化しますので、つねに上記の実行例のような表示になるというわけではありませんから、注意してください。

さて、ここでコマンドラインから `dir x68.*` と入力して、ディレクトリ情報を見てみましょう。

```

A> dir x68.*

```

TeX	A:\TeX			
4 ファイル	22921K Byte 使用中		17862K Byte 使用可能	
ファイル使用量	14K Byte 使用			
x68	tex	3460	94-05-11	17:20:38
x68	log	1384	94-05-11	17:21:20
x68	aux	369	94-05-11	17:21:18
x68	dvi	6572	94-05-11	17:21:20

```

A>

```

6) この場合は、jarticle.sty がディレクトリ c:/TeX/jmacros から読み込まれたことを示します。

LaTeX は、処理を行う過程でいくつかのファイルを生成します。これらのファイルは、LaTeX で相互参照を行う場合や、目次・表目次などの作成を行う場合に作成され、拡張子ごとにその役割が決まっています。そのうち代表的なものを、簡単にここでまとめておきましょう。

○ aux ファイル

相互参照を行う場合、および目次・表目次などの作成を行う場合に作成されます。目次と相互参照に利用する情報は、LaTeX で `%TEXHOME%\macros` 配下の `lablst.tex` を処理すれば出力できます。

具体的には、まず `lablst.tex` を LaTeX で処理します。続いて、メッセージに応じて、出力したい aux ファイルの拡張子を除いた部分を、続いて出力の形式 (文書スタイル) を入力します。すると、`lablst.dvi` が出力されますから、これをプリンタドライバにかけてください。参照ラベルが多い場合などは利用しているラベルの一覧を出力しておくとても非常に重宝しますから、試してみるとよいでしょう。

○ dvi ファイル

LaTeX の出力が書き込まれています。LaTeX の画面出力や印字出力を得る場合には、このファイルをプレビューアやプリンタドライバで処理することになります。このファイルが出力されない場合には、強制終了した場合のほか、なんらかの致命的な欠陥 (ソースレベルあるいは環境レベルで) がある場合もあります。

○ idx ファイル

索引の作成をする場合 (コントロール・シーケンス `\makeindex` 指定時) に出力されるファイルです。

○ log ファイル

LaTeX の処理過程を記録したファイルです。画面上に出力される情報のほか、追加情報が書き込まれている場合もありますから、エラーが起こった場合に参照するとよいでしょう。

○ toc ファイル

目次を作成する場合 (コントロール・シーケンス `\tableofcontents` 指定時) に出力されるファイルです。

これらのファイルはそれぞれ役割を持って作成されるものですから、ここでは消去したりせずに残しておいてください⁷⁾。

7) 消去しても再度 LaTeX で処理すれば、また生成してくれます。

◆ 警告 (Warning)

さて、この実行結果をよく見ますと、LaTeX が途中で警告 (Warning) を発していることに気づきます。

LaTeX Warning: Label(s) may have changed.

このメッセージが警告です。

L^AT_EX は、原稿を処理していく過程で、このようにさまざまな警告を発します。

この例で出力された警告は、L^AT_EX で当該ソースをはじめて処理する場合に、よく発生するものです。この警告は、ページ等の相互参照によって出力される情報に誤りがある可能性があることを意味しています。

L^AT_EX は、処理を行う前に原稿が全部で何ページになるのかを計算しないで、1 ページごとに文字をきれいに並べて順次出力していきます。このため、ソースの途中にページ番号の参照があった場合には、すべてのページ番号が決定されるまで正しいページ参照の処理結果は得られません。L^AT_EX は、1 回目の処理の際に参照があるたびにその情報を別のファイル (aux ファイル) に出力しておき、2 回目の処理をするとき⁸⁾ に、1 回目の処理で作成された aux ファイルを参照して、ページ番号等を参照部分に挿入するのです。

ところが、このときに挿入される文字数が多い場合、あるいはこのサンプルでは作成しませんでした。目次を作成する場合には、その分のページ増加にともなって、再びページの移動が起こることがあります。この場合には移動が起こった分の処理も必要ですから、結果的に参照を行う場合および目次を作成する場合、3 度 L^AT_EX の処理を行わなければならない⁹⁾ と思ってください。

このほかにも、L^AT_EX は次のような警告をしばしば発します。これらの警告に対しては、プレビューの結果に応じて無視するか、対処するかを判断します。

○ LaTeX Warning: Marginpar on page 12 moved.

本文の脚注がつけられた位置に対応する場所に脚注文章を出せなかった場合に発します。具体的には、脚注を指定した位置のすぐ前にも脚注があるために、あるいは非常に長い脚注があるために、脚注同士の出力が重なることのないよう、後出の脚注文を次ページなどに追い出した場合に発します。

したがって、この警告を発したときには、本文での脚注マークの位置と脚注が多少ずれて印刷されます。

○ LaTeX Warning: Reference 'XXX' on page 28 undefined.

参照されているラベル 'XXX' が見つからない場合に発する警告です。ラベル自体が間違っている場合と、原稿が未完成である場合とがあります。

○ LaTeX Warning: No typeface in this size, using \rm.

指定されたサイズのフォントがないために、代替のフォントが使われた場合に発します。

L^AT_EX は、警告のほかに以下のようなメッセージを出力することがあります。

○ Overfull \hbox (1.96pt too wide)...

原稿を規定の幅に収めることができず、横方向にはみ出した場合に発生します。L^AT_EX は、原稿を規定の幅に入れるように文字の間隔を縮めたり広げたりすることで調整を試みますが、文章の構成上、どうしても調整できなかった場合には、このメッセージを出力します。

○ Underfull \hbox (badness 1019)...

横方向の幅が広すぎて文字間隔が必要以上に開いてしまった場合に発生します。

8) L^AT_EX で同じファイルを処理します。特に最初の処理と違ったことをする必要はありません。

9) 論理的には何度処理をしても解決しない参照が起こり得ます。このような場合は手作業で修正を行います。

この場合も L^AT_EX はきちんと規定の文字間隔になるように調整を試みますが、どうしても調整できなかった場合には、このメッセージを出力します。

この 2 つのメッセージは、文章の横方向 (hbox) に関する不都合を報告するメッセージですが、まったく同じように縦方向 (vbox) にも同じような不具合を報告するメッセージが存在します。横方向のメッセージについては、よほどひどい Overfull や Underfull でない場合は我慢することもできますが、縦方向の場合はページを超えて出力がはみだし、出力が欠けてしまう場合がほとんどです。いずれにしても、これらのメッセージについて対処するかしないかは、仕上がり (プレビュー結果) を見てから決めます。

◆ エラー (Error)

今回のサンプルでは発生しませんが、場合によっては L^AT_EX が致命的な不具合を感知して、エラーメッセージ (Error) を出力したうえ “?” を表示したまま止まってしまうこともあります。このようなエラーメッセージが出力されるのは、特に初級者の場合、ほとんどが L^AT_EX のソースファイルにおける記述上の不具合を原因としていますから、適宜修正してやらなければなりません。

ここで少し寄り道をして、エラー発生時に L^AT_EX がどのような表示をするのかを見てみましょう。先のサンプルソース List 4-1 (p.81) の 60 行目、

```
\section{アプリケーションソフトウェア}
```

という記述を、

```
\sectoen{アプリケーションソフトウェア}
```

上のように変更 (\section を \sectoen とタイプミスしたという想定) したファイルを、x68_err.tex というファイル名で作成してみてください。これを L^AT_EX で処理してみます。

```
A> virtex &jlplain x68_err.tex
This is pTeX, C Version 2.99 j1.7 p1.0.9F EW (no format preloaded)
LaTeX Version 2.09 <24 May 1989>
(x68_err.tex (o:/tex/jmacros/jarticle.sty
Document Style 'jarticle' <18 Dec 88>.
(o:/tex/jmacros/jart10.sty)) (o:/tex/jmacros/b5j.sty)
No file x68_err.aux.
LaTeX Warning: Reference '1st_section' on page 1 undefined.
[1]
Overfull \hbox (7.09404pt too wide) in paragraph at lines 56--59
[]\tenmin 残念ながら、\tenrm X68030 \tenmin に標準搭載されている
\tenrm CPU \tenmin が \tenrm MMU []\tenmin を持たない \tenrm MC680EC30
! Undefined control sequence.
1.60 \sectoen
{アプリケーションソフトウェア}
?
```


！で始まる 1 行が「エラー・インジゲータ」と呼ばれるもので、発生したエラーの原因を究明する手がかりとなります。この場合、

！ Undefined control sequence.

というのがエラー・インジゲータにあたり、出力されているメッセージのとおり、未知のコントロール・シーケンスを使用した場合に発生します。

もし、ここでもう少し詳しいエラー情報を知りたい場合には、？が表示されて入力待ちになっている状態から、h を入力してみましょう。

```
A> virtex &jlplain x68_err.tex
This is pTeX, C Version 2.99 j1.7 p1.0.9F EW (no format preloaded)
LaTeX Version 2.09 <24 May 1989>

:
:
! Undefined control sequence.
1.60 \sectoen
      {アプリケーションソフトウェア}

? h
The control sequence at the end of the top line
of your error message was never \def'ed. If you have
misspelled it (e.g., '\hobx'), type 'I' and the correct
spelling (e.g., 'I\hbox'). Otherwise just continue,
and I'll forget about whatever was undefined.

?
```

ところで、エラーが発生した箇所の特定は、先に説明したエラー・インジゲータの次の行に表示されている「エラー・ロケータ」と呼ばれる数字を参照することによって可能です。この場合には、「1.60」という表示から、Line No.60、すなわちソース x68_err.tex の 60 行目に問題があることがわかります¹⁰⁾。

さて、エラーの原因は、定義されていないコントロール・シーケンス \sectoen を使用してしまったということに問題があります。具体的には、本来 \section と記述すべきところを \sectoen と記述したことに原因があるのですから、これを本来記述するはずだったコントロール・シーケンス \section に書き換えてみることにしましょう。

先ほど L^AT_EX で処理をした過程、つまり、エラーが出力されて？が出力されている状態から、I\section と入力してみます。

10) エラーによっては、エラー・ロケータで表示された行ではなく、もっと以前の行にその根本原因があることもありますから、注意しなければなりません。

```

A> virtex &jlplain x68_err.tex
This is pTeX, C Version 2.99 j1.7 p1.0.9F EW (no format preloaded)
LaTeX Version 2.09 <24 May 1989>

:

:

! Undefined control sequence.
1.60 \sectoen
      {アプリケーションソフトウェア}

? h
The control sequence at the end of the top line
of your error message was never \def'ed. If you have
misspelled it (e.g., '\hobx'), type 'I' and the correct
spelling (e.g., 'I\hbox'). Otherwise just continue,
and I'll forget about whatever was undefined.

? I\section
[2] (x68_err.aux)
LaTeX Warning: Label(s) may have changed.
      Rerun to get cross-references right.
Output written on x68_err.dvi (2 pages, 6560 bytes).
Transcript written on x68_err.log.

```

今度は処理ができました。? の状態へ入力した文字列の最初の 1 文字 I (あるいは i) は、続いて入力された文字列とエラー・ロケータの隣に表示されている文字列とを置換するための指示です。したがって、この場合、\sectoen と \section が置換されたことになります。

ここで、もう一度 L^AT_EX で処理してみてください。

```

A> virtex &jlplain x68_err.tex
This is pTeX, C Version 2.99 j1.7 p1.0.9F EW (no format preloaded)
LaTeX Version 2.09 <24 May 1989>
(x68_err.tex (c:/tex/jmacros/jarticle.sty
Document Style 'jarticle' <18 Dec 88>.
(c:/tex/jmacros/jart10.sty)) (c:/tex/jmacros/b5j.sty)
(x68_err.aux)
[1]
Overfull \hbox (7.09404pt too wide) in paragraph at lines 51--54
[]\tenmin 残念ながら、\tenrm X68030 \tenmin に標準搭載されている
\tenrm CPU \tenmin が \tenrm MMU []\tenmin を持たない \tenrm MC680EC30
! Undefined control sequence.
1.60 \sectoen
      {アプリケーションソフトウェア}

?

```

また同じところで止まってしまいました。

このように L^AT_EX 上で対話的に行う修正の場合には、ソースファイル自体を書き換えているわけではありません。したがって、対話的に修正を行ったあとでは、ソースに対しても同様の修正を行わなければなりません。

そこで、修正を行うため L^AT_EX の処理を抜け出したいときには、入力待ちの ? が表示されている状態から “x” を入力してください。あるいは、? の状態から “e” を入力することで L^AT_EX 上から直接エディタを起動して、当該エラー発生行を呼び出すことも可能です。

ただし、エディタを起動させるためには、あらかじめ環境変数 TEXEDIT に起動したいエディタのファイル名、および当該エディタで指定行にジャンプするためのオプションなどを登録しておかなければなりません。なお、Human68k 標準のエディタ ED.X の場合、起動時に指定行へのジャンプを行う機能がありませんから、残念ながら、この機能は使用できません (ソースの呼び出しだけなら可能です)。

```
A> virtex &jlplain x68_err.tex
This is pTeX, C Version 2.99 j1.7 p1.0.9F EW (no format preloaded)
LaTeX Version 2.09 <24 May 1989>
:
:
! Undefined control sequence.
1.60 \sectoen
      {アプリケーションソフトウェア}
? x
Output written on x68_err.dvi (1 page, 3644 bytes).
Transcript written on x68_err.log.
A> set TEXEDIT=emx.x %s -g%d
A> virtex &jlplain x68_err.tex
This is pTeX, C Version 2.99 j1.7 p1.0.9F EW (no format preloaded)
LaTeX Version 2.09 <24 May 1989>
:
:
! Undefined control sequence.
1.60 \sectoen
      {アプリケーションソフトウェア}
? e
Output written on x68_err.dvi (1 page, 3644 bytes).
Transcript written on x68_err.log.
      <エディタ emx.x を自動起動>
```

この設定例では、エディタ emx.x (μ Emacs¹¹⁾) が自動起動されます。環境変数 TEXEDIT を設定する場合には、%TEXHOME%\TeXenv.bat に追加記入しておくといでしょう。

以上で述べた以外にも、L^AT_EX での処理中、入力待ち状態 (? が表示されている状態) に陥った場合に使用することができる指示はいくつかありますから、こ

11) 第6章「T_EXfamily」
(p.269) で紹介します。

こでまとめておくことにしましょう。具体的にどうなるかについては、それぞれ、先ほどの `x68_err.tex` を処理することで試してみてください。

○ **〈改行〉**

何も入力しないでリターンキーを押した場合です。この場合、 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ の判断でエラーを修正して、処理を続けます。必ずしもうまくいく場合ばかりではありませんが、ほかにエラーが存在するかどうかを見きわめたい場合や応急処置として利用する場合には、十分に使用できるでしょう。

車の運転にたとえるなら、「教官が同乗して口うるさくチェックを入れている状況」といえます。

○ **S あるいは s**

「エラーが発生するたびに〈改行〉を入力している」という状態になります。画面上に出力されるエラーメッセージを追いきれない場合もあるかもしれませんが、画面出力は拡張子 `.log` のファイルに書き出されていますから、ソースの修正をする際にはこのファイルを参考にすれば問題ありません。

車の運転にたとえるなら、「1人で運転している状況」といえます。

○ **R あるいは r**

エラーによって停止する状態を S の場合以上に抑制したもので、指定したファイルが見つからないというような致命的なエラーでも停止しません。

車の運転にたとえれば、「幼児が車を運転している状況」といえます¹²⁾。

○ **Q あるいは q**

エラーによって停止する状態を S はおろか R の場合以上に抑制したもので、指定したファイルが見つからないというような致命的なエラーでも停止しないのに、エラーメッセージの画面出力すら抑制します。とにかく早く処理結果を得たい場合には利用価値があるかもしれません。

車の運転にたとえれば、「無人の車が暴走する状況」といえます。

○ **Istr あるいは istr**

$\mathrm{T}_{\mathrm{E}}\mathrm{X}$ は次に読み込む 1 行の前に、I に続いて入力された文字列 *str* を挿入して処理をします。具体的には、画面上、エラー・ロケータの隣に出力された文字列を訂正したい場合に使います (先の実行例を参照してください)。

○ **n**

n は 100 以下の値で、 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ は次に読み込むソースから文字やコントロール・シーケンスを指定された値だけ読み飛ばし、確認のために再度入力待ちの状態になります。

○ **H あるいは h**

詳しいエラー情報を出力します。ただし、英語です。

○ **X あるいは x**

強制終了 (`eXit`) します。このとき、処理が終了しているページまでを `dvi` ファイルに書き出します。

○ **E あるいは e**

X と同様に強制終了したあとで、環境変数 `TEXEDIT` で指定されたエディタを起動、カーソルを処理対象ファイルの適当な箇所に移動してくれます。

12) もっとも、小学生の少女が 1 人で、信号や車といった「障害物」が少ない国道を数十キロもドライブしたという事例を見れば、酔っぱらい運転にたとえるほうがふさわしいかもしれません。

なお、特に L^AT_EX の初級者が犯しやすい間違い、およびその際に発生するエラーについては、本章の最後、第 4.6.1 項「日常あるいは平穏な日々」(p.157)で述べることにして、ここでの説明は省略します。

◆ 2 回目の処理

さて、話を List 4-1、x68.tex に戻しましょう。

List 4-1 の場合、2 度目の処理以降は、これまでに挙げた、どの警告も発しないようになります。

```
A> virtex &jlplain x68.tex
This is pTeX, C Version 2.99 j1.7 p1.0.9F EW (no format preloaded)
LaTeX Version 2.09 <24 May 1989>
(x68.tex (c:/tex/jmacros/jarticle.sty
Document Style 'jarticle' <18 Dec 88>.
(c:/tex/jmacros/jarti0.sty)) (c:/tex/jmacros/b5j.sty) (x68.aux) [1]
Overfull \hbox (7.09404pt too wide) in paragraph at lines 56--59
[]\tenmin 残念ながら、\tenrm X68030 \tenmin に標準搭載されている
\tenrm CPU \tenmin が \tenrm MMU []\tenmin を持たない \tenrm MC680EC30
[2] (x68.aux)
(see the transcript file for additional information)
Output written on x68.dvi (2 pages, 6512 bytes).
Transcript written on x68.log.
A>
```

これで、サンプルソース List 4-1 が「dvi ファイル」と呼ばれるファイル形式に変換されました。この例の場合は、元のソースファイル名が x68.tex でしたから、x68.dvi というファイルに結果が出力されます¹³⁾。dvi ファイルは、これまで何度か説明してきたようにプレビューで閲覧したり、プリンタドライバで印刷したりすることができる形式のファイルです。

13) T_EX の出力結果は、ソースファイルの拡張子を dvi に置き換えたファイル名で書き出されます。

4.2.4 dvi ファイルをプレビューする

サンプルソース List 4-1 の L^AT_EX による処理は完了しました。今度は指示したイメージのとおり文書が作成されているか、処理結果をディスプレイ画面に出力して確認してみます。

◆ フォントマネージャの常駐

ディスプレイ画面で閲覧を行うためには、プレビューア (preview.x) を使用しますが、プレビューアを使用するためにはフォントマネージャ (fontman.x) を常駐させておかなければなりませんから、気をつけてください。

まず、フォントマネージャを常駐させます。

```
A> fontman c:/tex/myfonts.fm
X68k Font Manager Ver 3.00c Copyright (C) 1989,...,92 by E x t
  フォントマネージャが常駐しました。
  コンフィギュレーションを読み込みます。
  ドライバを登録します。
      :
      :
登録は正常終了しました。
A>
```

「登録は正常終了しました。」というメッセージが出力されていることを確認してください。このメッセージが出力されていないようであれば、フォントマネージャのインストールに失敗している可能性もあります。

ここで示した例のように、フォントマネージャを常駐させるにはコマンドラインから以下のように入力します。

```
fontman configfile
```

本書添付のインストーラでインストールしていれば、%TEXHOME%\myfonts.fm が configfile にあたります。もし、長いファイル名を入力するのが面倒であれば、例によって本書の添付ディスクからインストールした T_EX のシステムにはバッチファイルが用意してありますから、

```
FMset
```

としても同じ結果が得られます。

◆ プレビューアの実行

それでは、List 4-1 の処理結果をディスプレイ画面で閲覧してみましょう。

```
A> preview x68
X680x0 TeX Previewer Ver 2p09b Copyright 1989,90, ... ,93 by E x t
  Tpic Specials support Ver 1.00a Copyright 1993 by TSG+SHIMA
```


上記の例のように、プレビューはコマンドラインから以下のように入力することで起動します。

```
preview file(.dvi)
```

インストーラによって構築された標準の環境であれば、まず画面が白地に変化して、その後しばらく待つと、先ほど作成した文書の 1 ページめが画面に現れるはずです。

画面に出力が現れるまでの時間や、日本語フォントの美しさなどは、各人のマシンや周辺ハードウェアの違い、日本語フォントの設定状況によって相当な差があります。特にアウトラインフォントを使い、これをクロック 10MHz の X68000 で表示する場合には相当待たされます。

一般に画面で L^AT_EX の出力を閲覧する場合には、イメージしていたレイアウトどおりになっているかどうかをチェックしたり、誤字脱字がないかを調べたりするのが目的でしょうから、この場合は高速表示できる X680x0 本体の ROM フォントを使うのがよいかもしれません。

ROLL UP キーと ROLL DOWN キーでページめくりができますが、クロック 10MHz の X68000 では `preview.x` の `-buf` オプションでバッファを多めにとっておかないと待たされることが多いでしょう。

さて、プレビューで閲覧した結果はどうでしょうか。本来であれば、ここで指示したイメージどおりに出力されるかどうか、誤字脱字がないかどうかなどを調べておかねばなりません。そして、問題があったら、「L^AT_EX ソースの作成」(p.80)まで戻って、ソースを修正し、L^AT_EX での処理を行い、またプレビューする、という過程を繰り返します。

ここで注意しておきますが、もしソースに大幅な手直しがあった場合には、たとえばページ番号なども変わってくるでしょうから、変更前のソースを処理した際に生成されたファイル¹⁴⁾は消しておいたほうがよいでしょう。

14)aux ファイルや toc ファイルなどのことです。

4.2.5 プリンタで出力する

満足のいくプレビュー画面が得られたら、プリンタで印刷を行います。

通常、印刷をするのは「完全に仕上がった原稿を紙に印刷する」場合がほとんどですから、このときには、多少の時間がかかっても各人の環境で最も美しいフォントを使うように設定しておきましょう。フォントの品質とプリンタの性能の差がそのまま印刷結果に反映されます。いうまでもありませんが、一般的に ROM フォントよりはアウトラインフォントのほうがきれいですし、180dpi のプリンタよりは 360dpi のプリンタのほうがきれいです。

さて、紙にプリンタで印字するためには、プリンタドライバ (`print.x`) を用います。プリンタドライバを起動する場合には、コマンドラインから以下のように入力してください¹⁵⁾。

```
print file(.dvi)
```

具体的には、次のように実行すればプリントアウトすることができます。

15) よくある失敗が、シャープの XC に添付されていた同名ファイルを誤って実行してしまうというものです。恥ずかしい話ですが、本章の筆者にも経験があります。引数を指定しないで `print.x` を実行させたときに、Copyright などの情報が表示されるようであれば、プリンタドライバを起動させたことを示しています。

◎ L^AT_EX の出力性能

360 dpi のプリンタでは 180 dpi のプリンタに比べて同じ面積に対して 4 倍の点を印刷できます。なぜなら、同じ長さを表現するために、360dpi は 180dpi の 2 倍の点を使用する（縦横それぞれ 2 倍の点を使用すれば、当然 4 倍になる）からです。

さて、通常、漢字のフォントの大きさは“16 ドット×16 ドット”といった点の数で表すことが多いのですが、普通、点を使って漢字を表現する場合（たとえば 24 ドット×24 ドットで構成された漢字を出力した場合）、180dpi の漢字 1 文字は 360dpi の漢字 1 文字の 4 倍の大きさになります。ところが、L^AT_EX の印刷は 180dpi でも 360dpi でも、文字の大きさはまったく同じに印刷されます。冒頭で述べたように、360dpi は 180dpi に対して同じ面積を 4 倍の点で印字することができますから、L^AT_EX で同じ大きさの印刷物を印刷する場合、4 倍分だけ、文字の縁のギザギザが目立たないきれいな印字結果を得ることができるのです。

ところで、L^AT_EX の出力する dvi ファイルには、通常のプリンタで出力できる 360dpi よりのはるかに高い精度、具体的には、1pt(ポイント) の 65536 分の 1 にあたる、1 sp(スケールドポイント) = 0.5362×10^{-5} mm の精度で文字の位置情報が書き込まれています（これが L^AT_EX の出力結果をそのまま出版物に使える秘密でもあるのです）。

ところが、先に述べたように、市販されているプリンタの精度はせいぜい数百 dpi でしかありません。この程度の性能では、1 sp、dpi に換算すれば 470 万 dpi を超えるという、きわめて高い L^AT_EX の精度を表現しつくすことはできないのです。ですから、あなたのプリンタでの印刷結果を見て、それを L^AT_EX の出力品位だとは思わないでください。一般に市販されているプリンタでは、L^AT_EX の美しい出力結果を得るには性能が低すぎるのです。

```
A> print x68
```

```
X680x0 TeX Printer Driver Ver 2p09b Copyright 1989, ... ,93 by E x t
      Tpic Specials support Ver 1.00a Copyright 1993 by TSG+SHIMA
      RasterOutput Devices support Ver 1.00 Copyright ... by T. Hilano
[1] [2] [3]
```

```
A>
```

4.2.6 L^AT_EX へようこそ

前節までに、L^AT_EX の処理の流れを確認してきましたが、同時に L^AT_EX が具体的にどのような出力を行うことができるものなのか、おわかりいただけたのではないかと思います。

まだ、一から L^AT_EX のソースを作成していくことはできないでしょうが、今回示したサンプルソースを書き換えることで、自分の書いた文章を L^AT_EX で出力することはできるかもしれません。

はじめのうちは、とにかくそのような試みを繰り返して L^AT_EX での処理方法に慣れてください。

もちろん、いくつかの理由によって、あなたが作成した新たなソースが L^AT_EX の処理を通らないこともあるでしょう。場合によっては、「ワープロを使ったほうがはるかに早く文書になっていた」と思うときもあるかもしれません。

その意味では、「今、あなたは L^AT_EX の世界の入口にいますが、“世界の中”には入っていません」といったほうが、表現としてより正確であるといえるでしょう。おそらく、ワープロユーザが “T_EX を自然に使いこなす” ようになるためには、BASIC オンリーのプログラマが C 言語に移るとき以上の抵抗を感じるに違いないのですから。

しかし、そこであきらめないでください。「苛立つほど退屈な反復のなかにある基本」を身につけなければ使い物にならないということは、T_EX の世界にかぎったことではないはずです。

そこで次節からは、もう少し根本的な「退屈な基本」の領域、あなた自身が L^AT_EX のソースを作成するためのノウハウについて踏み込んでみることにしましょう。

4.3 最小の L^AT_EX ガイド

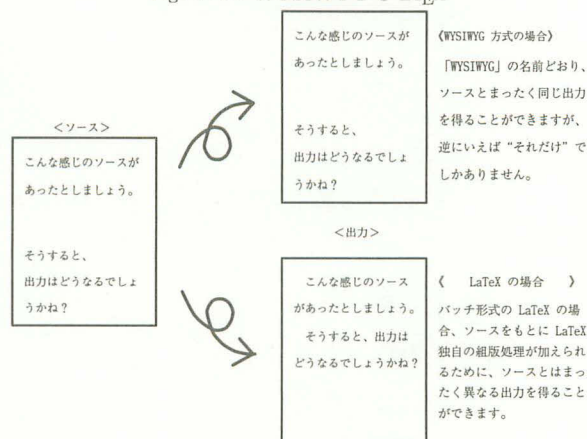
本節に至るまでの間に、あなたは何度か L^AT_EX のソースファイルをご覧になったことと思います。たとえば、第 1.2.2 項「T_EX の処理の流れ」の List 1-2 (p.7) や、「L^AT_EX ソースの作成」の List 4-1 (p.81) などはその一例です。それらのソースを見ればわかるように、執筆者はテキストエディタを使用して、L^AT_EX で処理したい原稿に L^AT_EX への指示命令であるコントロール・シーケンス (control sequence)¹⁾ を埋め込んでおかなければなりません。このことが、WYSIWYG 方式に慣れているユーザが T_EX に対し最も抵抗を感じ、最も面倒に思うところであるといえます。

たとえば WYSIWYG 方式の場合、空行 (改行だけの行) を 3 行入力すれば、この空行は「3 行分の縦方向の空白」として反映されます。これに対して L^AT_EX の場合には、後述するように、空行は段落の区切りを示し、さらに複数の連続する空行はひとつの空行として扱われますから、たとえばソース中に空行が 3 個連続している場合でも、実際に出力される「縦方向の空白」は、ソース中に空行が 1 個しかない場合と変わりません。つまり、もし L^AT_EX において「縦方向の空白」をつくりたいのであれば、「縦方向の空白」を作成するコントロール・シーケンス²⁾ を使って明示的に空白の高さを指定しなければならないことになります。

1) 「コマンド」という表現をしている場合もありますが、意味内容に違いはありません。本節では Knuth 博士の「T_EX book」にならって、「コントロール・シーケンス」と表記します。

2) 縦方向の空白を作成するためのコントロール・シーケンスには、「縦方向の空白を制御する」(p.155) で詳説する `\vspace` があります。

Fig. 4-1 ● WYSIWYG と L^AT_EX



この例に代表されるように、WYSIWYG 方式に慣れていればいるほど、L^AT_EX を使用する場合に、そのギャップに悩むこともあるかと思います。しかし実際の

ところ、L^AT_EX のソース作成において注意しなければならないことは、さほど多くありません。

本節では、WYSIWYG 方式と比べて特に注意しなければならないことも含めて、L^AT_EX のソース作成に際して最も基本的なこと、いいかえれば L^AT_EX のソースを作成するための最小限の約束について概説していくことにします。説明だけが続いて退屈してしまうかもしれませんが、本節で解説することは L^AT_EX を利用するための最も基本的なことですから、我慢しておつきあいください。もし本節を十分に理解することができれば、少なくとも既存のプレーンテキストを L^AT_EX でベタ出力³⁾することは可能になるはずですから。

4.3.1 コントロール・シーケンス

L^AT_EX への指示命令は、エスケープ文字 (escape character : “\”)⁴⁾ から始まるコントロール・シーケンスによって行います。

ここでいう「エスケープ文字」とは、キーボードのキー配置を変更するプログラムを使用していないかぎり、X680x0 の場合にはキーボード上の ¥ キーを押したときに出力される文字のことを指します。このとき X680x0 では、`switch.x` コマンドによる設定に応じて “¥” か “\” のどちらかがディスプレイ上に出力されるはずです。“¥” と “\” は視覚的に異なる文字ですが、コンピュータの内部では同じコードで処理されていますから、画面上にどちらの文字が表示されても問題はなく、その文字がエスケープ文字にあたります。なお、本項ではオリジナルの T_EX が稼働する UNIX の環境にならい、“\” を使用しています。

さて、コントロール・シーケンスはエスケープ文字のあとに続く文字によって以下の 2 種類に分類することができます。

- 文字列が続くコントロール・ワード。
- 全角および半角の記号がひとつだけ続くコントロール・シンボル。

ここでいうコントロール・ワードの場合には、コントロール・ワードと、直後に続く本文との境界を明らかにしなければならないため、コントロール・ワードの直後に〈空白〉(いわゆる「半角スペース」のこと)を付加する必要がある場合がほとんどです。このことも含め、コントロール・ワードとコントロール・シンボルの詳細は、本節のコラム (p.98, 99) および「独特な働きをするコード」(p.99)の〈空白〉の箇所でも説明していますから、もう少し詳しく知りたい人はそちらを参照してください。

4.3.2 特殊文字

前項で、エスケープ文字 (“\”) について説明をしましたが、L^AT_EX にはエスケープ文字のように特別な機能を割り当てられていたり、独特な働きをしたりす

3) 論理構造や視覚構造を制御するコントロール・シーケンスを使用しないで得た出力のことです。このような L^AT_EX の使い方は、本来の「L^AT_EX を使う」という観点には反しているのですが、ヘタなワープロを使用するより美しい出力が得られるかもしれません。

4) ESC キーのことではありませんから注意してください。

◎ コントロール・シーケンス その 1 ～コントロール・ワード～

エスケープ文字の直後に 1 文字以上のアルファベットおよび全角文字が続いているものをコントロール・ワード (control word) といいます。

具体的には、エスケープ文字に続いて A, B, ..., Z および a, b, ..., z までの半角英文字、ならびに全角の英数字・かな・カナ・漢字を 1 文字以上組み合わせたもので、0, 1, ..., 9 のような半角数字、. (ピリオド) や : (コロン) などの全角および半角の記号を含みません。

ただし、標準の \LaTeX の環境には全角文字を使用したコントロール・ワードは定義されていません。

なお、 \TeX は大文字と小文字を認識するので、たとえば `\item`, `\Item`, `\ITEM` は、すべて異なるコントロール・ワードとして扱われます。

また、コントロール・ワードを構成する文字列の直後には、アルファベットおよび全角文字以外の文字 (たとえば半角の数字や全角および半角の記号) か、`<空白>` あるいは `<改行>` を続けなければなりません。これは、コントロール・ワードの終端を明らかにするためです。

たとえばコントロール・ワード `"\item"` に続けて `"X680x0"` と入力したい場合、`"\item_X680x0"` と記述します。もしこのとき、`"\itemX680x0"` と記述してしまった場合には、`\itemX` というコントロール・ワードの後に、`"680x0"` という文字列が続いているものとみなされてしまいます。なお、`\item_X680x0` と記述した場合、`\item` 直後の `<空白>` は出力されません。

(p.99 へ続く)

る記号や文字コードがあります。本項ではそのような記号および文字コードについて少し詳しく見てみることにします。

ただし、はじめて読む段階で、以下に説明する内容をすべて理解する必要はありません。「ここで示す文字をソース中に記述する場合には注意を要するのだ」という程度の認識があれば十分です。

◆ 特殊な機能を持つ文字

\LaTeX において特殊な機能を定義されている文字は、エスケープ文字を含めて Table 4-1 に示す 10 個です。これらの特殊文字を出力したい場合には、Table 4-1 の「文字の出力」の欄で示したコントロール・シーケンス⁵⁾を使用するか、ソース中に記述されているとおりに出力することができるコントロール・シーケンス⁶⁾を使用しなければなりません。

5) ここで、`\? (? は 1 文字の半角記号)` で表すことができるコントロール・シーケンスを、コントロール・シンボルといいます。

6) 第 4.4.5 項「ソースに忠実な出力をする構造」(p.126) で詳説します。

○ コントロール・シーケンス その 2 ～コントロール・シンボル～

エスケープ文字の直後にアルファベットではない記号、たとえば `,` `$` `%` などがひとつだけ続くものをコントロール・シンボル (control symbol) といいます。

コントロール・シンボルの場合にはエスケープ文字に続く記号がひとつのみであると決まっているために、コントロール・ワードのように終端を考慮する必要はありません。

コントロール・シーケンスのデリミタ (区切り) については、これを理解していない、と思わぬところに空白をつくってしまったり、逆に空白ができなかったり、コントロール・シーケンスが機能しなかったり、自作のマクロが期待どおりに動作しなかったり、といった事態が考えられますから注意しなければなりません。

(p.104 へ続く)

Table 4-1 ● L^AT_EX における特殊文字

文字	機能	文字の出力
<code>\</code>	エスケープ文字	<code>{\tt\symbol{'134}}</code>
<code>{</code>	グループ化を開始する	<code>\{</code>
<code>}</code>	グループ化を終了する	<code>\}</code>
<code>\$</code>	対応する <code>\$</code> にはさまれた領域を数式モードにする	<code>\\$</code>
<code>&</code>	表を構成する要素の区切りを表す	<code>\&</code>
<code>#</code>	コントロール・シーケンスの引数を示す	<code>\#</code>
<code>^</code>	数式モードで後続の 1 文字 (1 グループ) を上付きの添字にする	<code>{\tt\symbol{'136}}</code>
<code>_</code>	数式モードで後続の 1 文字 (1 グループ) を下付きの添字にする	<code>_</code>
<code>%</code>	<code>%</code> 以後の文字列を〈改行〉も含めてコメントにする	<code>\%</code>
<code>~</code>	改行を抑制する単語間スペース	<code>{\tt\symbol{'176}}</code>

◆ 独特な働きをするコード

Table 4-1 に示した 10 個の特殊文字のほかにも、〈空白〉や〈改行〉のように L^AT_EX で独特な働きをする文字 (コード) が存在します。

○ 〈空白〉 (`\space`)

スペースキーを押したときに出力される、ディスプレイ上でいう「半角スペース」コードのことです。

連続した複数の〈空白〉は 1 個とみなされます (Table 4-2 No.3)。通常は `\space` で「単語間空白」と呼ばれるスペースが得られます (Table 4-2 No.2, 3) が、コントロール・ワードの直後にある〈空白〉は単語間空白をつくりません (Table 4-2 No.4, 5)。

このため、もしコントロール・ワードと後続するアルファベットの間に単語間空白をつくりたい場合 (Table 4-2 No.6~8) や単語間空白を連続させたい場合 (Table 4-2 No.9) には `_` として、単語間空白の作成を明示的に指示しなければならないのです。

そこで、たとえばコントロール・シーケンスを Table 4-2 の No.10~13 のように記述すれば、つねにその直後の<空白>の有無を純粹に出力に反映することができるようになります。

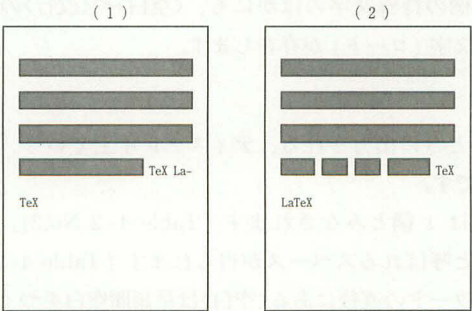
Table 4-2 ● <空白> に関するサンプル

No.	ソース	出力
1	TeXnician	TeXnician
2	TeX_nician	TeX nician
3	TeX_ _nician	TeX nician
4	\TeX_nician	T _E Xnician
5	\TeX_ _nician	T _E Xnician
6	\TeX\ _nician	T _E X nician
7	\TeX\ _ _nician	T _E X nician
8	\LaTeX\ _nician	L ^A T _E X nician
9	\LaTeX\ _ _nician	L ^A T _E X nician
10	{\LaTeX}nician	L ^A T _E Xnician
11	{\LaTeX}_nician	L ^A T _E X nician
12	\LaTeX{}nician	L ^A T _E Xnician
13	\LaTeX{ }nician	L ^A T _E X nician

また、ソース上、<改行>の直前および直後に記述された<空白>はすべて無視されますから、注意してください（詳細はあとで述べる<改行>の解説を参照）。なお、L^AT_EX は、処理を行う際に出力時の改行位置が単語の区切りになるように各単語間の空白を調節します。したがって、単語間空白にあたる<空白>部分は、半角英数字や記号が連続する途中に比べて特に出力時点の改行位置になりやすいのです。たとえば、ソース上の“TeXLaTeX”という 2 単語からなる文字列が出力時の行末にきて、なおかつ行末の空白が「TeX La」分しかなく、文字列のすべてが同じ行に入りきれないとしましょう。このとき、L^AT_EX では「TeX La- (改行) TeX⁷⁾」より「TeX (改行) LaTeX」として出力されやすいといえます (Fig. 4-2)。

7) 英単語のハイフネーションのしかたには規則がありますが、この例では考慮していません。ただし L^AT_EX は、誤ったハイフネーションのしかたにならないようにできていますから、ハイフネーション処理を行う場合にもさほどユーザが気を遣う必要はありません。

Fig. 4-2 ● L^AT_EX の改行制御 ～その 1

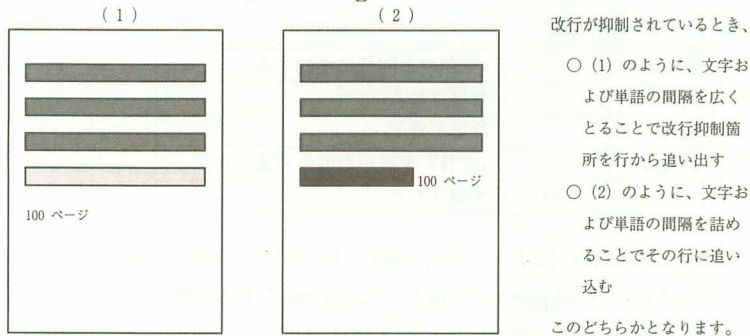


LaTeX では、(1) のような出力にはなりにくくなっています。

つまり、(2) のように、単語間あるいは文字間の空間を調整することで、単語の途中で改行が行われないように、可能なかぎり制御されます。

しかし、ソース上での「100_ページ」が出力時の行末にきた場合など、「100」と「ページ」が分かれて「100 (改行) ページ」と出力されてしまうと可読性に欠けます。この場合、若干無理をしても同一行に収めるべきです (Fig. 4-3)。

Fig. 4-3 ● L^AT_EX の改行制御 〜その2



このように、単語間空白に相当する位置での改行を抑制したい場合には、`<空白>`のかわりに改行を抑制して⁸⁾ 単語間空白をつくる“`~`”を使用します (Table 4-3 No.2)⁹⁾。

ただし、`~` と `<空白>` には以下の点で違いがあるので注意しなければなりません。

- `<空白>` と `~` とを混用した場合には、どちらもが単語間空白を構成する (Table 4-3 No.3~6)。
- 連続する複数の `<空白>` はひとつとみなされるが、“`~`” はひとつとはみなされない (Table 4-3 No.7, 8)。

Table 4-3 ● `~` に関するサンプル

No.	ソース	出力
1	<code>LaTeX_nician</code>	LaTeX nician
2	<code>LaTeX~nician</code>	LaTeX nician
3	<code>LaTeX~_nician</code>	LaTeX nician
4	<code>LaTeX_nician</code>	LaTeX nician
5	<code>LaTeX~_nician</code>	LaTeX nician
6	<code>LaTeX~_~nician</code>	LaTeX nician
7	<code>LaTeX~~~nician</code>	LaTeX nician
8	<code>LaTeX~~~~nician</code>	LaTeX nician

ところで、「全角スペース」は全角および半角の記号と同じ扱いを受けるので、入力されている分だけ出力されることになります。

○ `<改行>`

リターンキーを押したときに出力されるコードのことを示します。

ソースにおける 1 行の終了を意味するのですが、出力時の動作は `<改行>` の直前にくる文字によって変化します (Table 4-4)。

8) `\mbox{word}` とすることで、`word` での改行を抑制させることもできます。Fig. 4-3 のように「100 ページ」をつねにひとまとまりとして扱いたいときは、このコントロール・シーケンスを使用します。

9) 先に挙げたソース上で「100 ページ」という例の場合、「100」と「ページ」の間で改行抑制すると、「ページ」が前の行に追い込まれるか、「100」が次の行に追い出されます。追い込みの対象となる文字が「ページ」である理由は、「一」が禁則処理の対象文字だからです。

Table 4-4 ● <改行> の動作

直前	直後	動作
半角文字	—	<空白> と同様に機能する
全角文字	—	無視される
<改行>	—	段落の終了
<空白>	—	<空白> を除いた直前の文字に従う
—	<空白>	無視される

10) インデントの抑制を行うことは可能で、その場合には抑制したい部分で“noindent”を指定します。

11) Warning となる場合もありますが、無視してリターンキーを押し、処理を続行した場合には本文のように動作します。

連続した<改行>によって空行（改行コードだけの行）となっている場合には段落の終わり（\par）を意味し、次の行頭では自動的にインデント（字下げ）が行われます¹⁰⁾。複数の空行が連続する場合、1 個の空行として扱われることは第 4.3 節「最小の L^AT_EX ガイド」の Fig. 4-1（p.96）で示したとおりです¹¹⁾。

<改行>の直前および直後の<空白>はすべて無視されます。したがって、<改行>の直前に<空白>があった場合、Table 4-4 でいう「直前の文字」とは、<改行>以前にあって<改行>に最も近い<空白>以外の文字を意味します。

たとえば、次のようなソースがあったとしましょう。

List 4-2 ● <改行> と <空白>

- 1: \mbox{ <改行> }の手前にあるのが全角文字の
- 2: 場合、改行は無視されますが、Alphabet
- 3: などの半角文字の場合、\mbox{ <空白> }として処理されます。
- 4: 「\,Alphabet\,」の直後に、単語間空白に相当する
- 5: スペースが空いていることに注目してください
- 6: い。ちなみに、単語間空白がない場合の出力は Alphabet となります。
- 7:
- 8: 行末の\mbox{ <空白> }があり、その\mbox{ <空白> }の
- 9: 前に全角文字があるとき\mbox{ <空白> }
- 10: \mbox{ <空白> }と\mbox{ <改行> }は無視されます。
- 11: その\mbox{ <空白> }の
- 12: 前にあるのが半角文字の場合、たとえば、X680x0\mbox{ <空白> }
- 13: というとき、行末の\mbox{ <空白> }は無視され、
- 14: \mbox{ <改行> }は\mbox{ <空白> }として
- 15: 処理されます。「\,X680x0\,」の直後につくられる
- 16: 単語間空白に注目してください。
- 17:
- 18: 行頭の\mbox{ <空白> }は、単純に無視されます。
- 19: \mbox{ <空白> }のような全角文字でも
- 20: \mbox{ <空白> }Alphabetなどの半角文字でも変わりません。

このソースから、次に示す出力が得られます。

〈改行〉の手前にあるのが全角文字の場合、改行は無視されますが、Alphabet などの半角文字の場合、〈空白〉として処理されます。「Alphabet」の直後に、単語間空白に相当するスペースが空いていることに注目してください。ちなみに、単語間空白がない場合の出力は Alphabet となります。

行末の〈空白〉があり、その〈空白〉の前に全角文字があるとき〈空白〉と〈改行〉は無視されます。その〈空白〉の前にあるのが半角文字の場合、たとえば X680x0 というとき、行末の〈空白〉は無視され、〈改行〉は〈空白〉として処理されます。「X680x0」の直後につくられる単語間空白に注目してください。

行頭の〈空白〉は、単純に無視されます。このような全角文字でも Alphabet などの半角文字でも変わりません。

List 4-2 とその出力を比較しながら、行末に〈空白〉がある場合にどのように処理されるかを見てみましょう。

- (1) 〈改行〉の直前にある〈空白〉はすべて無視される (ソース 9, 12 行目)。
- (2) ソース 9 行目の場合、〈改行〉以前にある〈空白〉以外の文字、「き」が全角文字だから、〈改行〉は無視される。
- (3) ソース 12 行目の場合、〈改行〉以前にある〈空白〉以外の文字、「0」が半角文字だから、〈改行〉は〈空白〉として機能する。

これに対して、行頭に〈空白〉がある場合には、〈空白〉が無視されるだけで、特別な処理は行われません。

なお、「\ 〈改行〉」は、「_」と同様の扱いを受けます。

○ 〈タブ〉

タブキーを押したときに出力されるコードのことです。

〈空白〉と同様に機能します。

なお、「\ 〈タブ〉」は、「_」と同様の働きをします。

○ 半角カナ

半角のカタカナを示します。

半角カナは使用できません。ソースに含まれていた場合、L^AT_EX で処理をしたときに「無効な文字がある」旨を通告してきます。これを見無視してリターンキーを押した場合、その文字は無視されます。

4.3.3 L^AT_EX ソースの構造

本項では前項までに解説してきたことをふまえて、L^AT_EX のソースの基本構造について見ていくことにします。ほとんどすべての L^AT_EX ソースは、本項で示すのと同じような形式にのっとって書かれていますから、はじめのうちは「約束ごと」として、その作法を真似してください。

④ コントロール・シーケンス その 3 ～プリミティブ～

コントロール・シーケンスには、 \LaTeX の最も基本となる実行ファイル `virtex` に登録されていて、最も単純でそれ以上分解できない低レベルのものがああります。これを、「プリミティブ」といいます。

プリミティブには、条件分岐、文字列の処理、ファイルの入力などの働きを持つものがあり、ある意味ではプログラム言語といってもよいでしょう。しかし、これらプリ

ミティブはあくまでも低レベルな処理を行うことしかできませんし、その数は 300 個程度でしかありません。

そこで `virtex` は、複数のプリミティブを組み合わせてつくり出された各種のマクロを `fmt` ファイルから読み込むことによって、高度な機能と使いやすさを考慮したコントロール・シーケンスを多数、使用することができるのです。

さて、「 \LaTeX ソースの作成」に示したサンプルソース List 4-1 (p.81) にも見られるように、 \LaTeX のソースファイルは一般的に以下に示すような構造を持っています。

\LaTeX ソースの構造

```
\documentstyle[style_option]{stylefile}
preamble
\begin{document}
text
\end{document}
```

◆ `\documentstyle`

`\documentstyle` は、文書の体裁を定めるコントロール・シーケンスです。

\LaTeX の作成者である L.Lamport 氏は、 \LaTeX で論文や書籍を作成する場合に、印刷の体裁が異なる場合があることを考慮していました。そのために、 \LaTeX の本体部分¹²⁾ では、マクロなどの包括的な部分だけを設定し、実際に作成する文書の体裁などの詳細な部分は「スタイルファイル」と呼ばれる、ソースとは独立したファイルで用意することにしたのです。つまり \LaTeX は、処理を行う段階でソース中の指定にもとづいてスタイルファイルを読み込み、はじめて文書の体裁を確定するようになっています。このために \LaTeX では、異なるスタイルファイルを読み込むことで、体裁の異なる文書を簡単に実現することができるのです。

`\documentstyle` は、当該 \LaTeX 文書において利用する文書スタイルを設定するためのコントロール・シーケンスであり、この宣言がない場合には実際の文書の体裁が定まらないために \LaTeX で処理してもエラーになってしまいます。ですから、 \LaTeX のソースファイルを作成する場合には、まず、このコントロール・シーケンスから書き始めるようにしましょう。

なお、`\documentstyle` の引数には、マクロの追加などを行う `[style_option]`、利用する文書スタイルを定める `{stylefile}` があります¹³⁾。

サンプルソース List 4-1 (p.81) の場合、2 行目に記述されていました。

12) `jlplain.tex` によって生成される `fmt` ファイルを読み込んだ `virtex` のこと。

13) ちなみに \LaTeX では、コントロール・シーケンスの引数で省略できないものは `{ }` で囲み、省略可能な引数を記述する場合には `[]` で囲んで記述します。

◆ *style_option*

「スタイルオプション」とか「スタイルファイルオプション」などと呼ばれ¹⁴⁾、スタイルファイルによって定められた文書スタイル (*stylefile*) の一部を変更したり、特別な論理構造を追加したりする場合などに記述しますから、必ず記述しなければならない部分ではありません。そのかわりというわけではありませんが、いざ記述するという場合には、おのおのの指定が干渉を起こさない範囲と順番を守れば (後述)、“,” で区切って列記することができます。

スタイルオプションには、一部のオプションを除き、当該オプション事項を定義しているスタイルオプションファイル (スタイルファイルと同様に拡張子が *.sty* であるファイル) から拡張子を除いた部分を設定してください。

本書が構築する L^AT_EX において利用できるスタイルオプションのうち主なものを、Table 4-5 に示しておきます。

Table 4-5 ● スタイルオプション

記述	ファイル名	機能
11pt	*11.sty	11pt を \normalsize にする
12pt	*12.sty	12pt を \normalsize にする
a4j	a4j.sty	A4 サイズにする
a5j	a5j.sty	A5 サイズにする
b4j	b4j.sty	B4 サイズにする
b5j	b5j.sty	B5 サイズにする
twoside	(jarticle.sty)	偶数ページと奇数ページのレイアウトを変える
titlepage	(jarticle.sty)	タイトルや abstract を別ページに出力する
ascmac	ascmac.sty	アスキーの拡張マクロを使用する
landscape	landscape.sty	出力する用紙を横置きにする
jtwocolumn	jtwocolumn.sty	2 段組にする
multicol	multicol.sty	多段組の拡張を行う
makeidx	makeidx.sty	インデックス作成用のファイルを作成する
epic	epic.sty	図形描写機能を拡張する
piclatex	piclatex.sty	図形描写機能を拡張する

ただし、Table 4-5 における *twoside* および *titlepage* は、文書スタイルが *jarticle* のときにのみ指定できるオプション指定です。*twoside* オプションを指定した場合、たとえば偶数ページと奇数ページで綴じ代の幅を変更することなどが可能となります。また、*titlepage* オプションを指定した場合、*jarticle* スタイルの標準タイトル出力 (`\maketitle`) および *abstract*¹⁵⁾ の出力が、それぞれ本文とは別ページになるなど、製本を考慮した出力になります。

また、最も基本となる文字の大きさ (`\normalsize`) を設定するためのオプションが、11pt ならびに 12pt です。これらのオプションを指定しない場合、10 ポイントが `\normalsize` になります¹⁶⁾。

サンプルソース List 4-1 (p.81) の場合、2 行目の「b5j」という記述がスタイルオプションにあたります。この場合、`\normalsize` のフォントの大きさを特に指定していないために 10 ポイントがベースに、文書体裁については「b5j」

14) 筆者のあくまで個人的なとらえ方としては、2 つは別物だと思っています。ということかという、スタイルファイルオプションという場合にはスタイルファイルに制約されて指定の可否が決まるオプション指定を、スタイルオプションという場合にはスタイルファイルに制約を受けずに新たな論理構造や視覚構造を追加するオプション指定を、意味するものとして、両者を区別しているのです。しかし、このとらえ方が一般的なものかどうか筆者は知りませんし、まぎらわしいので、本文中は一部を除いてスタイルオプションという記述で統一しています。

15) 序文あるいは要約のことです。*abstract* 環境によって出力されます。

16) スタイルオプション 11pt, 12pt については、「L^AT_EX における文字の大きさ」(p.138) で詳説しています。

○ 日本語 $\text{T}_{\text{E}}\text{X}$ における全角文字の扱い

コントロール・ワードについてのコラムで若干触れた、日本語 $\text{T}_{\text{E}}\text{X}$ における全角文字の扱いがよくわからなかった人もいるかと思います。

日本語 $\text{T}_{\text{E}}\text{X}$ では、全角の 1 区 (読点やかっこなどの特殊文字)、2 区 (利用頻度の少ない数学記号などの特殊文字)、7 区 (ロシア文字) に含まれる文字を、半角の . (ピリオド) や : (コロン) などの記号と同様に “その他の文字 (Other character)” (カテゴリー・コード 12) として扱います。したがって、全角の 1, 2, 7 区をコントロール・シーケンスにする場合、これらはコントロール・シンボルとしてのみ使用でき

ます。

その他の全角文字は、半角のアルファベット A, B, ..., Z および a, b, ..., z と同様に、“英文字 (letter)” (カテゴリー・コード 11) として扱われます。したがって、そのような全角文字は半角のアルファベットと混在させることができます。

逆にいえば、半角のアルファベットからなるコントロール・ワードに対して、直後に続くそのような全角文字は、デリミタ (区切り) としての機能を果たしませんから、その間には半角の〈空白〉に値する文字を入れてやらなければならないのです。

17) B5 判、A4 判については「-width, -height」(p.188)を参照してください。

18) List 4-1 (p.81) の 14 行目の行頭に “%” が付いていることを確認しておいてください。サンプルソースで得られる体裁を `jarticle` スタイルといいます。このスタイルでは章節を表す構造のうち、`\chapter` が使用できません。詳しくは第 4.4.1 項「章節にかかわる構造」(p.113)で述べます。

と記述することによって B5 判 (無指定の場合は A4 判¹⁷⁾) をベースにしているのです。

ここで、サンプルソース List 4-1 の 2 行目の `b5j` の記述を以下に示すように変更したうえで、それぞれ第 4.2.3 項「 $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ による処理」(p.82)の「 $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ による処理」以降の過程を実行し、プリンタ出力を得てみましょう¹⁸⁾。

```
\documentstyle{jarticle}
```

```
\documentstyle[a5j]{jarticle}
```

```
\documentstyle[a4j,12pt]{jarticle}
```

```
\documentstyle[a4j,titlepage]{jarticle}
```

```
\documentstyle[a4j,jtwocolumn]{jarticle}
```

```
\documentstyle[landscape,a4j,jtwocolumn]{jarticle}
```

それぞれ、若干異なる体裁で出力されるはずですが、簡単に説明しておくと、`a5j` および `a4j` は、出力する用紙のサイズを A5 および A4 に変更しますし、`12pt` は、最も基本となるフォントの大きさを 12 ポイントに変更する (デフォルトは A4 判で 10 ポイント文字) ものです。`titlepage` は、文書スタイル `jarticle` に対してのみ作用し、標準のタイトル出力および序文 (abstract) の出力を、本文とは別ページに出力します。`jtwocolumn` は 2 段組を実現し、`landscape` は出力を横長 (用紙を横置きにした状態) にします。

さて、先に少しだけ触れましたが、これらのスタイルオプションは複数記述することができるものの、その記述順序には注意が必要です。たとえば、先に変更したサンプルソース List 4-1 の 2 行目を、以下のように変更してみます。

```
\documentstyle[jtwocolumn,b5j]{jarticle}
```

```
\documentstyle[a4j,landscape,jtwocolumn]{jarticle}
```

この変更をしたあと、それぞれについてプリンタ出力を得てみます。

前者の場合、 $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ の処理の段階で Warning が発生して、結果的に「B5 用

紙に出力すべし」という指定 (b5j) が無効になってしまいます。

```
A> virtex "&plplain \input x68"
This is pTeX, C Version 2.99 j1.7 p1.0.9F EW (no format preloaded)
LaTeX Version 2.09 <4 Aug 1988>
(x68.tex (c:/TeX/jmacros/jarticle.sty
Document Style 'jarticle' <18 Dec 88>.
(c:/TeX/jmacros/jart10.sty)) (c:/TeX/jmacros/jtwocolumn.sty)
(c:/TeX/jmacros/b5j.sty
LaTeX Warning: Sorry, I ignored 'b5j' style.
)
No file x68.aux.
LaTeX Warning: Reference '1st_section' on page 1 undefined.
[1] (x68.aux)
LaTeX Warning: Label(s) may have changed.
Rerun to get cross-references right.
Output written on x68.dvi (1 page, 6892 bytes).
Transcript written on x68.log.
A >
```

後者の場合も、同様に Warning が発生して、用紙を横置きにした出力を指定する landscape が無効になってしまいます。

```
A> virtex "&plplain \input x68"
This is pTeX, C Version 2.99 j1.7 p1.0.9F EW (no format preloaded)
LaTeX Version 2.09 <4 Aug 1988>
(x68.tex (c:/TeX/jmacros/jarticle.sty
Document Style 'jarticle' <18 Dec 88>.
(c:/TeX/jmacros/jart10.sty)) (c:/TeX/jmacros/a4j.sty)
(c:/TeX/jmacros/landscape.sty
LaTeX Warning: Sorry, I ignored 'landscap'.
) (c:/TeX/jmacros/jtwocolumn.sty)
No file x68.aux.
LaTeX Warning: Reference '1st_section' on page 1 undefined.
[1] [2] (x68.aux)
LaTeX Warning: Label(s) may have changed.
Rerun to get cross-references right.
Output written on x68.dvi (2 pages, 6824 bytes).
Transcript written on x68.log.
A >
```

これらの例のように、スタイルオプションには記述順序に制約があるものはいくつかありますから、注意しなければなりません。ここで紹介したスタイルオプションの記述順序については、詳しく説明するだけの紙面がありませんので省略しますが、先ほど出力に成功した例にならうようにすれば確実です。そのほかの

19) 記述順序を気にせずに使ってみて、問題が起こったときに「試行錯誤」しながら解決するというのが、最も現実的な解決方法であるかもしれません。

20) スタイルファイルについて述べると本が 1 冊できあがってしまいますし、なによりそれを詳説できるほどの能力が本章の筆者にはありません。詳しく知りたい人は、アスキーの『UNIX MAGAZINE』や『Super ASCII』のバックナンバー、技術評論社の『 \LaTeX 美文書作成入門』（奥村晴彦著）などから関連記事を探してください。ただし、この領域に関心を持つことは理解できますが、それは「正しい \LaTeX の使い方」に反しますし、一步間違えば「底なしの泥沼」（けっして悪い意味ではありません）に入り込みますから、努力と根性と体力(?)と、後戻りしない決意に自信のある方にしかお勧めできません。

21) `jbook` では、章の見出しがつねに奇数ページにくるように空白ページをつくるなど、製本を考慮した出力となります。

22) 頭についている `j` は、日本語対応であることを示しています。

スタイルオプションについては、ファイル内に記述されたドキュメントやコメントを参考にしてください¹⁹⁾。

◆ stylefile

「文書スタイル (ドキュメントスタイル)」あるいは「スタイルファイル」と呼ばれ、最も基本となる文書の体裁を定義する部分です。本書では、文書の体裁についていうときは「文書スタイル」、文書スタイルを定めたファイルについていうときは「スタイルファイル」という呼び方をすることにします。

スタイルファイルでは、版面 (余白部分を除いた、文字が印字されるスペース) の大きさや各種マージンの大きさといった、文書そのものの外観に加え、章節の番号や箇条書きのラベルをどのような形式にするのかというような設定がなされています²⁰⁾。

スタイルファイル (拡張子が `.sty` であるファイル) の拡張子を除いた部分を設定してください。

本書が構築する \LaTeX において利用できる文書スタイルのうち、主なものは Table 4-6 のとおりです。

Table 4-6 ● 文書スタイル

記述	ファイル名	文書の体裁
<code>ascjletter</code>	<code>ascjletter.sty</code>	日本語用の手紙形式
<code>ascletter</code>	<code>ascletter.sty</code>	英語用の手紙形式
<code>jarticle</code>	<code>jarticle.sty</code>	日本語対応の短い学术论文の文書形式
<code>jreport</code>	<code>jreport.sty</code>	日本語対応の中規模レポートの文書形式
<code>jbook</code>	<code>jbook.sty</code>	日本語対応の書籍規模の長い文書の形式

文書スタイルが `jbook` のときは、つねにスタイルオプション `twoside` が指定されているかのように、ページの左右にあたる偶数ページと奇数ページとで体裁が変化します (`jreport` では変化しません²¹⁾)。

サンプルソース List 4-1 (p.81) の場合、2 行目 “`jarticle`” の記述が文書スタイルの設定でした。サンプルソース List 4-1 によって出力された文書の体裁を、「`jarticle` スタイル」といいます²²⁾。この文書体裁は、学术论文などの比較的短い文書用の形式として開発されました。もっとも、「短い文書用」とはいえ、スタイルオプションの説明で述べたとおり、左右 (偶数奇数) ページで綴じ代などの取り方を変更できたり、文書タイトルや序文 (abstract) を本文とは独立したページに出力できたりと、製本も十分に考慮されたつくりになっています。

ここでサンプルソース List 4-1 の 2 行目の、`jarticle` という記述を、以下に示すように `jreport` あるいは `jbook` に変更し、どちらの場合にも 14 行目の行頭にある “`%`” の文字を消去したものを \LaTeX の処理にかけ、プリンタで出力してみましょう。

```
\documentstyle[b5j]{jreport}
```

```
\documentstyle[b5j]{jbook}
```

jreport は日本語対応の中規模論文の形式、jbook は日本語対応の書籍形式の体裁を持っています。2 つの文書スタイルの違いは、出力された文書体裁を見ればわかるでしょう。

なお、jarticle, jreport, jbook の違いについては、後述の第 4.4.1 項「章節にかかわる構造」(p.113)でも触れることにします。

◆ *preamble*

「プリアンブル」と呼ばれ、自作のマクロの定義やレイアウトパラメータ²³⁾の変更、タイトル用のパラメータ設定を行う部分です。

たとえば、サンプルソース List 4-1 (p.81) のような、jarticle スタイル²⁴⁾の標準タイトル出力 (`\maketitle`) を行う場合に設定をしなければならない、以下のようなコントロール・シーケンスはここに記述するのが一般的です。

- タイトル (*title_name*) を設定する。

```
\title{title_name}
```

- 著者名 (*author_name*) を設定する。

```
\author{author_name}
```

- 日付 (*date*) を設定する。

```
\date{date}
```

日付は必ずしも設定する必要はありません。省略した場合、L^AT_EX で処理を行った日付 (`\today`) が設定されます。

サンプルソース List 4-1 では 3~9 行目がプリアンブルに該当します。

◆ `\begin{document}`

`\begin{document}` は、出力対象となる文章が始まることを示すコントロール・シーケンスです。必ず記述されていなければなりません。

サンプルソース List 4-1 (p.81) では、10 行目に記述されています。

◆ *text*

実際に出力する本文を書く部分です。

なお、*preamble* の部分で述べた、スタイルファイルの標準タイトルを出力するコントロール・シーケンス `\maketitle` は、この部分に記述します。

サンプルソース List 4-1 (p.81) では、11~80 行の記述が実際に出力されます。

23) 上下左右のマージンや改行幅などの、レイアウトにかかわる数値設定のこと。スタイルファイルに設定されている既定値を変更したい場合などに設定します。

24) jreport や jbook でも共通のコントロール・シーケンスが使用できます。

◆ `\end{document}`

`\end{document}` は、出力対象となる文章が終了したことを示すコントロール・シーケンスです。そのため、`\end{document}` のあとは何を書いても無視されることになります。

L^AT_EX のソースであるかぎり、`\begin{document}` とともに必ず記述されていなければなりませんから、`\documentstyle` とあわせて、ソース作成の最初の時点で記述するようにしましょう。

なお L^AT_EX には、`\begin{document}` ~ `\end{document}` のように「`\begin` ~ `\end`」というペアで記述すべきコントロール・シーケンスがいくつか定義されており、自分でも定義することができます。これらのコントロール・シーケンスには含まれた領域を「環境」といい、環境のなかでは、ある特定の規則によって文章や数式を記述することができます。たとえば `\begin{document}` ~ `\end{document}` は「document 環境」といい、この領域に記述された文章が出力対象となるのです。

サンプルソース List 4-1 (p.81) では、`\end{document}` は 81 行目に記述されています。

4.3.4 最小の「お約束」

解説ばかりが続きましたから、いささか退屈してしまったかもしれません。しかし、前項までに説明してきたことは L^AT_EX の最も基本的な部分ですから、少しずつでも理解しておいてもらいたいところでもあります。

本項では、ここまでに示した最小の「お約束」について、もう一度復習しておくことにします。

- L^AT_EX への指示は、エスケープ文字 “\” から始まるコントロール・シーケンスで行う。

ここでコントロール・シーケンスとは、エスケープ文字 (escape character : “\”)²⁵⁾ の直後に続く L^AT_EX への指示命令のことをいいます。

X680x0 の場合のエスケープ文字とは、キーボードのキー配置を変更するプログラムなどを使用していないかぎり、キーボード上の π キーを押したときに出力される文字のことです。

- {, }, \$, &, #, ^, _, %, ~ の各文字には特殊な機能があるため、当該文字を出力したい場合、そのままソースにこれらの文字を記述することはできない。
- コントロール・シーケンスの直後に空白があるとき、その空白は、ほとんどの場面で当該コントロール・シーケンスの終端を示す。

コントロール・シーケンスは、文字列が続くコントロール・ワードと、1 文字の半角記号が続くコントロール・シンボルとに分類できます。そして、ここでいうコントロール・ワードの場合には、コントロール・ワードと直後

25) ESC キーのことではないので注意してください。

に続く文字列との境界を明らかにしなければならないため、コントロール・ワードの直後に〈空白〉を付加するのです。

- コントロール・ワードの直後以外の場所で〈空白〉を使用した場合には、その〈空白〉はほとんどの場面で「単語間空白」と呼ばれるスペースをつくる。ただし、連続する複数の〈空白〉はひとつとみなされます。また、〈全角の空白〉は全角および半角の記号と同様の扱いを受けます。
- 全角文字の直後の改行は無視され、そうでない場合の改行は〈空白〉として扱われる。
- 空行 (〈改行〉だけの行) は段落の終了を意味し、直後の文頭ではインデント (字下げ) が行われる。
- タブは〈空白〉として扱われる。
- 半角カナは使用できない。
- ソースのはじめに、次のコントロール・シーケンスを必ず記述し、実際に出力する文章は `\begin{document}` 以下に続ける。

```
\documentstyle[a4j]{jarticle}
\begin{document}
```

ただし、`a4j` および `jarticle` の記述は、好みと場合に応じて変更できます。

- ソースの終わりに、次のコントロール・シーケンスを必ず記述する。

```
\end{document}
```

本節冒頭で述べたように、以上の事柄に注意してさえいれば、少なくとも L^AT_EX によってベタ出力を行うことは可能ですし、ベタ出力でもそれなりに整った出力を得ることはできます。しかし、実際のところ、ベタ出力では L^AT_EX の神髄²⁶⁾ の 1 割も発揮させられません。

そこで次節からは、ベタではない出力を得るために、いいかえれば L^AT_EX らしい使い方をするために、`document` 環境の記述のしかたについて触れていくことにしましょう。

26) 詳しくは後出のコラムなどで述べますが、L^AT_EX の神髄とは、文章の論理構造を反映した文書が手軽に得られるということです。

4.4 論理構造を表す環境

L^AT_EX のコントロール・シーケンスのほとんどは、文章の論理構造を表すものであって、視覚的な制御を行うためのものではありません。どういうことかという、たとえば「L^AT_EX ソースの作成」で示したサンプルソース List 4-1 (p.81) において、章節を設定するコントロール・シーケンスである `\section` に与えた章節題は、本文の文字よりも大きな文字で出力されたはずですが、これはユーザが意識的に文字を大きく出力しようとした結果ではありません。あくまでも、ユーザは「ここから新しい章節ですよ」という文章の論理的な構造を宣言したにすぎないのです。

もうひとつ例を挙げておきましょう。「*style_option*」(p.105) および「*stylefile*」(p.108) で試みた、文書スタイルの変更では、それにとまって文書の体裁が大きく変化したはずですが、WYSIWYG 方式であれば作り直しに近いこの変更が、L^AT_EX ではたった数語の修正で実現できてしまいました。これは、L^AT_EX が文章の論理構造に注目するようになっており、その構造を文書の体裁へと反映しているからにほかなりません。

このように、L^AT_EX のコントロール・シーケンスのほとんどは文章の論理構造を示すものであって、L^AT_EX はコントロール・シーケンスによって示された文章の論理構造を、読者がとらえやすいようにレイアウトします。したがって、「L^AT_EX を使う」ためには、論理構造を示すコントロール・シーケンスを的確に用いなければなりません。

そこで本節では、L^AT_EX の根幹ともいえるべき、文章の論理構造を示すためのコントロール・シーケンスのうちのいくつかについて概説したいと思います。

その前に、ここで L^AT_EX の動作モードについて簡単に説明しておきましょう。L^AT_EX には、ソースを処理するために、以下のような大きく 3 つの動作環境が用意されています。

○ パラグラフモード (段落モード)

この動作環境では、入力テキストは単語や文のひとまとまりとして扱われ、T_EX は、これを行や段落、ページに分割していきます。通常、テキストを記述して出力する際の動作環境がこれに該当します。

○ LR モード (左右モード)

この動作環境では、入力テキストは左から右へと並べられるだけで、改行処理はなされません。たとえば、表を作成する場合に使用される動作環境がこれに該当します。

● ピクチャーモード (描画モード)

LR モードの特殊な一形態で、図形を描くためのモードです。picture 環境のなかでのみ使用され、図形描画のためのかぎられたコントロール・シーケンスしか使用できません。

○ 数式モード

この動作環境では、入力テキストは数式として扱われます。たとえば、dir は「d」と「i」と「r」の積とみなされ、特別なフォントによって *dir* のように出力されるのです。また、ソース中でコントロール・ワードの終端を示す以外の〈空白〉はすべて無視されるので、単語間空白をつくりたい場合には“\ ”のように明示的に空白の制御を指定しなければなりません。文字どおり、数式の表現を行う場合に使用される動作環境がこれに該当します。

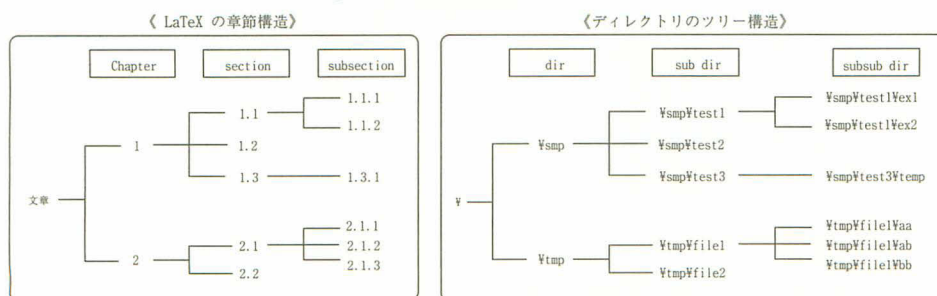
L^AT_EX は、ユーザがソース中に記述したコントロール・シーケンスに応じて、以上に示した大きく 3 つの動作環境のなかから、コントロール・シーケンスが意味する文章の論理構造を表現するために最適な動作環境を選択することで、文章の論理構造を反映した美しい出力を生み出しているのです。

4.4.1 章節にかかわる構造

最も代表的な文章の論理構造が「章節」です。L^AT_EX の場合、具体的にはディレクトリ・ツリーのイメージがそのまま、この構造のイメージに当てはまります (Fig. 4-4)。

どういうことかという、たとえば jbook スタイルの場合、ベースとなる文章があって、その下に章 (\chapter) が、さらに章の下に節 (\section) が、節の下に項 (\subsection) があるというように、階層構造になっています。これをディレクトリでいえば、まずルートがあって、その下に子ディレクトリ、さらにその子ディレクトリの下に孫ディレクトリがあって … というわけです。

Fig. 4-4 ● 章節の階層構造



1) これは、jreport スタイルや jbook スタイルの文書のひとつの章 (\section) として、jarticle スタイルの文書を簡単に取り込むことができるようにするためです。

しかも、ディレクトリの絶対パスの表記のように、各章節は基本的に独立していて、新たに章が始まれば節および項の番号はふたたび 1 から始まります。さて、その章節の構造と出力体裁ですが、これはスタイルファイルによって支配されます。たとえば、「 \LaTeX ソースの作成」(p.80)で紹介した jarticle スタイルは、比較的短い文章を書くためのものであるために、長文を対象としている jbook と比べて設定可能な章節が 1 レベル少なくなっています¹⁾(Table 4-7)。

ここで、Table 4-7 については以下の点に注意して見てください。

- 「出力形式」の欄では、実際に出力される文字の大きさや、空白の大きさについては考慮に入れていない。

あくまでも章節番号および章節題、本文が出力される位置関係の確認としてのみ利用してください。できるだけ、実際にサンプルソースをつくって出力を見てみるとよいでしょう。

- 「省略可能」なコントロール・シーケンスは、文字どおり省略しても問題ない。

「省略可能」なコントロール・シーケンスは、長い文書を大きく分割する際に使用するためのもので、その配下の章節の具体的な番号付けに対してなんら影響を与えない、きわめて例外的な章節構造になっています。たとえば Part I が 3 節で終了したとき、Part II は 4 節から始まります。

- 「使用可能」なコントロール・シーケンスは、表中に示した Level が低い順に使用していかなければ、章節のナンバリングが正しく行われぬ。

たとえば jarticle スタイルにおいて、冒頭で \section の前に \subsection を設置した場合、当該 \subsection は「第 0.1 節」の扱いとなり、架空の \section 第 0 節が存在したことになってしまいます。この意味がよくわからないときは、ディレクトリの場合、子ディレクトリをつくらなければ孫ディレクトリはつくれないことを考えていただければ、イメージがつかめるでしょう。

Table 4-7 ● 章節を表すコントロール・シーケンス

コントロール・ シーケンス	jarticle スタイル	jreport スタイル, jbook スタイル
	出力形式	出力形式
\part	省略可能 (Level 0)	省略可能 (Level -1)
	Part I (part タイトル) (本文)	Part I (part タイトル) <改ページ>
\chapter	使用不可	使用可能 (Level 0)
		Chapter 1 (chapter タイトル) (本文)
\section	使用可能 (Level 1)	使用可能 (Level 1)
	1 (section タイトル) (本文)	1.1 (section タイトル) (本文)
\subsection	使用可能 (Level 2)	使用可能 (Level 2)
	1.1 (subsection タイトル) (本文)	1.1.1 (subsection タイトル) (本文)
\subsubsection	使用可能 (Level 3)	使用可能 (Level 3)
	1.1.1 (subsubsection タイトル) (本文)	(subsubsection タイトル) (本文)
\paragraph	使用可能 (Level 4)	使用可能 (Level 4)
	(paragraph タイトル) (本文)	(paragraph タイトル) (本文)
\subparagraph	使用可能 (Level 5)	使用可能 (Level 5)
	(subparagraph タイトル) (本文)	(subparagraph タイトル) (本文)

ところで、Table 4-7 に示したように、標準の状態において jarticle スタイルで \paragraph 以下、jreport スタイルおよび jbook スタイルで \subsubsection 以下には、章節の番号が出力されません。これは、カウンタ secnumdepth によって章節番号の出力が抑制されているためです。

ここで「カウンタ」とは、 \LaTeX が生成するすべての番号の制御にかかわる変数の名前をいいます。基本的にカウンタの名前は、番号を出力するコントロール・シーケンスから「\」を除いたもの (番号を出力する環境の場合には環境名そのもの) です。たとえば、章節番号を出力するコントロール・シーケンス “\section” の場合、その番号付けの制御に使用されるカウンタの名前は “section” という具合になっています。そして、コントロール・シーケンス \section が使用されるたびに、カウンタ section に 1 が加えられ、章節番号を出力するために利用されているのです。しかし、なかには secnumdepth のように、単なる変数としてだけ存在し、番号の生成を行うコントロール・シーケンスを持たないカウンタも存在します。

話を戻しましょう。標準の状態では、カウンタ secnumdepth に 4 が登録されており、この値以上のレベル (Table 4-7 において Level ? で表示) では章節番号の出力が抑制されているのです。そこでもし、この抑制を行うレベルを変更したいのであれば、ソースに以下の記述を加えます。このとき、この記述以降に現れた level 以上のレベルを持つ章節では、番号の出力が抑制されます。

```
\setcounter{secnumdepth}{level}
```

◎ L^AT_EX について その 1 〜論理構造と視覚構造〜

Lamport 氏は、『文書処理システム L^AT_EX』(Edger Cooke・倉沢良一 監訳 / 大野俊治・小暮博道・藤浦はる美 訳、アスキー刊、1990 年)の第 1.4 節で、次のように述べています。

「デザイナーの役割は、読者がその考えを理解できるように手助けをすることである。文章を読みやすくするには、その体裁が論理的な構成を反映していなければならない。」

「L^AT_EX コマンドの基本的な役割は、文書の論理構成を示すためのものである。文書を書くときは、視覚的な体裁ではなく、その論理構成に集中すべきである。」

つまり、ここで Lamport 氏が述べているのは次のようなことです。以下、書面化された「文書」と文書の内容を意味する「文章」との違いに注意して読んでください。

○ 文章の執筆者は、文章が文書化された際の外観上の体裁を気にする必要はない。執筆者は、論理的な (= 論理的な構造を持った) 文章を書き上げることに集中すべきである。

○ 執筆者によって作成された文章を文書にするデザイナーは、文書の「きれい」な外観を重視してレイアウトするのではなく、文章の論理構造が読者にわかりやすくなる

ようなレイアウトをしなければならない。

このように、Lamport 氏は執筆者とデザイナーの役割をはっきりと区別しました。そして、よりたやすく、より論理的な文章を書くために、ここでいうデザイナーが行うべき役割を L^AT_EX に託したのです。

すなわち、L^AT_EX は文章の論理構造を反映してソース中に記述されたコントロール・シーケンスをもとにして、文章の論理構造が読者にわかりやすくなるようにレイアウトするツールとして作成されたのでした (Lamport 氏は、このような文書作成に対するアプローチを「論理デザイン (logic design)」と定義しています)。

これに対して、WYSIWYG 方式のツールの場合、ユーザが先に挙げた執筆者としての役割とデザイナーとしての役割を両立させなければなりません (Lamport 氏は、このような文書作成に対するアプローチを「視覚デザイン (visual design)」と定義しています)。そのために Lamport 氏は、WYSIWYG 方式のツールを使用した場合、往々にしてユーザが文書の外観上の「きれい」な体裁の完成にこだわってしまい、文章の内容についての論理性あるいは明確性については十分な考慮をしないですませてしまうことがあるとも指摘しています。

(p.128 へ続く)

4.4.2 引用にかかわる構造

章節を構成する文章のつらなりは、いくつかの文字が集まって単語になり、単語が集まって文となり、文が集まって段落となり、さらにその段落がいくつか集まることで形作られています。その章節を構成する「文」という単位に注目したとき、そこにいくつかの論理的構造を見出すことができます。

本項では、そのうちのひとつである「引用」という構造を紹介することにしましょう。

さて、引用という構造は、先に述べた「文」という構成単位に注目した場合、前後につらなる本文よりも論理的に下位の文章構成単位としての位置付けになります。これは、引用が前後にある本文の論旨の補助、補強のために用いられるという性格上、当然のことだといえるでしょう。

このような引用の性質を反映して、L^AT_EX が持つ引用の環境では、引用文を左

右適当な分だけ字下げして出力するようになっていました。そして、引用文の性質にあわせて 2 つの環境を選択することができるようになっているのです。

◆ quote 環境

引用の文章構造を実現するひとつめの環境が、`quote` 環境です。この環境は、短い文章を引用する場合や、短い文章を複数引用したい場合などに利用することを想定していて、引用文のなかに段落があったとしても、段落冒頭の字下げ（インデント）を行いません。

List 4-3 ● `quote` 環境のサンプルソース

```
1: \mbox{L.\,Lamport}氏は、\TeXと\LaTeXの違いを次のように位置付ける。
2:
3: \begin{quote}
4: \TeXは植字工
5:
6: \LaTeXは組版デザインの専門家
7: \end{quote}
8:
9: \noindent
10: すなわち、\LaTeXのコントロールシーケンスは\cdots
```

サンプルソース List 4-3 の出力サンプルを以下に示します。

L. Lamport 氏は、 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ と $\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ の違いを次のように位置付ける。

$\mathrm{T}_{\mathrm{E}}\mathrm{X}$ は植字工

$\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ は組版デザインの専門家

すなわち、 $\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ のコントロール・シーケンスは …

◆ quotation 環境

引用の文章構造を実現するもうひとつの環境が、`quotation` 環境です。

この環境は、引用する文章が長く、しかもいくつかの段落から構成されているような場合を想定しているため、`quote` 環境とは違って段落部分の字下げ（インデント）も行います。

List 4-4 ● `quotation` 環境のサンプルソース

```
1: 引用についての説明で、本文中、以下のように述べました。
2:
3: \begin{quotation}
4: さて、引用という構造は、先に述べた「文」という構成単位に注目した
5: 場合、前後につらなる本文よりも論理的に下位の文章構成単位としての位置付けに
```

なり

```

6:   ます。これは、引用が前後にある本文の論旨の補助、補強のために用いられる
7:   という性格上、当然のことだといえるでしょう。
8:
9:   このような引用の性質を反映して、\LaTeX が持つ引用の環境では、引用文
10:  を左右適当な分だけ字下げして出力するようになっています。そして、引用文の性
11:  質にあわせて 2 つの環境を選択することができるようになっているの
12:  です。
13:  \end{quotation}
14:
15:  \noindent
16:  そして、これがその実例なのです。

```

サンプルソース List 4-4 の出力サンプルを以下に示します。

引用についての説明で、本文中、以下のように述べました。

さて、引用という構造は、先に述べた「文」という構成単位に注目した場合、前後につらなる本文よりも論理的に下位の文章構成単位としての位置付けになります。これは、引用が前後にある本文の論旨の補助、補強のために用いられるという性格上、当然のことだといえるでしょう。

このような引用の性質を反映して、`LaTeX` が持つ引用の環境では、引用文を左右適当な分だけ字下げして出力するようになっています。そして、引用文の性質にあわせて 2 つの環境を選択することができるようになっているのです。

そして、これがその実例なのです。

4.4.3 箇条書きにかかわる構造

前項では、「文」という構成単位に着目して「引用」という構造に注目しましたが、それ以外にも「文」という構成単位に注目して考えられる論理構造があります。

それは、前後につらなる文章と論理的に下位あるいは同レベルに属する要素を列挙する場合、つまり「箇条書き」です。

文章の要点をわかりやすく示したいと思うとき、私たちは箇条書きを使用することがありますが、`LaTeX` にも要素を列挙するような箇条書きを行うための文章環境がいくつか用意されています。具体的には、`itemize` 環境、`enumerate` 環境、`description` 環境、`list` 環境などがこれにあたりますが、本項ではこれらのなかから 3 つ、`itemize` 環境、`enumerate` 環境、`description` 環境について概説します。

ここで `list` 環境の解説を省略したのは、`list` 環境は非常に奥が深いため、とうてい本項では紹介しきれないと思ったからです。いいかえれば、本項で解説するような「できあいの箇条書き環境」では満足できなくなった人にとって応用

範囲が広く、使い道も多いのが `list` 環境だといえます。意欲のある人は、ぜひ他の文献を参考に研究してみてください。

なお、「`list` 環境による出力例だけでも見てみたい」という人は、第6章「`TeXfamily`」(p.229)が `list` 環境によって実現されていますから、これを参考にするとよいでしょう (といっても、`list` 環境による出力例としてはさほど凝ったことはしていませんが)。

◆ `itemize` 環境

`itemize` 環境とは、項目を並べるだけという、最もオーソドックスな箇条書きの環境のことで、以下に示すようなものをいいます。

- これが `itemize` 環境である。
- 各項目の先頭にはラベルとして記号が付加される。
- 箇条書きのなかに箇条書きを行うという、入れ子 (ネスト) の構造を持つことも可能。その入れ子構造の深さを「ネストレベル」ということにする。
- これはネスト構造 (ネスティング) の第 1 レベルの要素である。
 - ネストを行うと、そのレベルに応じて記号も変化する。
 - これが第 3 ネストレベルである。
 - ネストは第 4 レベルまで行うことができる。
 - ここは第 2 レベルである。
 - ふたたびネストすることも可能になっている。
- ここが第 1 レベル。

この出力を得るためのソースを List 4-5 に示します。

List 4-5 ● `itemize` 環境のサンプルソース

```

1: %%%▼\itemize_環境をはじめるコントロール・シーケンスが
2: %%%\begin{itemize}です。
3: \begin{itemize}
4: %%%▼箇条書きの要素を示すコントロール・シーケンスが\item_です。
5: \item_これが\begin{itemize}環境である。
6: \item_各項目の先頭にはラベルとして記号が付加される。
7: \item_箇条書きのなかに箇条書きを行うという、入れ子(ネスト)の構造を持
8: つことも可能。その入れ子構造の深さを「ネストレベル」ということにする。
9: \item_これはネスト構造(ネスティング)の第1レベルの要素である。
10: %%%▼第2ネストレベルの\begin{itemize}です。
11: \begin{itemize}
12: %%%\item_ネストを行うと、そのレベルに応じて記号も変化する。
13: %%%▼第3ネストレベルの\begin{itemize}です。
14: \begin{itemize}
15: %%%\item_これが第3ネストレベルである。
16: %%%▼第4ネストレベルの\begin{itemize}です。
17: \begin{itemize}

```



```

18: \item_ネストは第4レベルまで行うことができる。
19: \end{itemize}
20: \item_第4レベルの\end{itemize}です。
21: \end{itemize}
22: \item_第3レベルの\end{itemize}です。
23: \item_ここは第2レベルである。
24: \item_第3レベルの\begin{itemize}です。
25: \begin{itemize}
26: \item_ふたたびネストすることも可能になっている。
27: \end{itemize}
28: \item_第3レベルの\end{itemize}です。
29: \end{itemize}
30: \item_第2レベルの\end{itemize}です。
31: \item_ここが第1レベル。
32: \end{itemize}
33: \itemize環境を終了するコントロール・シーケンスが
34: \end{itemize}です。

```

箇条書きの各要素となる `\item` 文の前に、〈空白〉が記述されていますが、これは `itemize` 環境におけるネストの関係 (`\begin{itemize}` と `\end{itemize}` の対応) をわかりやすくするために書かれているだけなので、記述しなくても出力には変わりありません。むしろ、 \LaTeX に慣れないうちは、〈空白〉や〈タブ〉を多用することでコントロール・シーケンス同士の対応関係を明らかにしたほうがよいでしょう。

ところで、このソースを標準の \LaTeX 環境で処理してみると、ラベルにあたる記号が、このサンプルの出力 (第1ネストレベルの場合、`○` のこと) とは異なっていることに気づくはずですが、どちらがよいかについてはたぶん好みの問題ですが、各ネストレベルのラベル記号を変更することができるようになっています。

第1ネストレベルのラベル記号をここでの出力サンプルのように変更する場合には、ラベル記号を変更したい箇条書き要素 `\item` より前に以下の記述を加えてください²⁾。記述以降に現れた第1ネストレベルのラベル記号が変更されます³⁾。

```
\renewcommand{\labelitemi}{\bigcirc}
```

同様に、変更したいのが第2ネストレベルのラベル記号であれば、上の記述において `\labelitemi` の部分を `\labelitemii` に、第3ネストレベルであれば `\labelitemiii` に、第4ネストレベルであれば `\labelitemiv` に変更した記述を、変更したい箇条書き要素の前に加えてください。ネストレベルが、`\labelitem` に続く小文字のローマ数字⁴⁾で表現されていることに注意してください。

◆ enumerate 環境

`itemize` 環境では、同一ネストレベルのラベルは、特に変更しないかぎり同じものでした。これに対して、`enumerate` 環境では、各要素が同一ネストレベルにおいて何番目の要素であるかを一見して判断できるように、通し番号をつける箇条書き環境です。

具体的には、以下に示すような出力をとります。

2) 同一のレベルで複数の記号を使用することもできます。ただし、単純な箇条書きで列挙される各要素は、ほとんどの場合、論理的に同値のはずですが、同一レベルでラベルが変化すると「各要素は同値」という論理性がとらえにくくなるので、必ずしもお勧めはできません。

3) この変更指定を `itemize` 環境に入ってから行った場合、変更はそのネストレベルに対応する `\end{itemize}` が最初に記述されているところまで有効です。詳しくは第4.4.7項「“グルーピング”にかかわる構造」(p.132)で述べます。

4) ここでアラビア数字が使用できない理由は、数字がコントロール・シーケンスのデリミタとして機能してしまうからにはなりません。

- (1) 第 1 ネストレベルの 1 番目の最初の要素。
- (2) 第 1 ネストレベルの 2 番目の要素。
 - (a) `itemize` 環境のように、ネスティングも可能。
 - `enumerate` 環境のなかに `itemize` 環境をつくることもできる。
 - i. もちろん、逆に `itemize` 環境から `enumerate` 環境をネストすることもできる。
 - ii. 第 3 レベルの ii 番目の要素。
 - No.1 ネスティングは第 4 レベルまで可能。
 - No.2 このとき、`itemize` など、他の環境のネストレベルとは無関係である。
 - (b) 第 2 レベルの II 番目の要素。
- (3) 第 1 ネストレベルの 3 番目の要素。

この出力を得るためのソースが List 4-6 です。

List 4-6 ● `enumerate` 環境のサンプルソース

```

1: %%%▼\enumerate環境をはじめるコントロール・シーケンスが
2: %%%\begin{enumerate}です。
3: \begin{enumerate}
4: %%%▼箇条書きの要素を示すコントロール・シーケンスが\itemです。
5: \item第1ネストレベルの\arabic{enumi}番目の最初の要素。
6: \item第1ネストレベルの\arabic{enumi}番目の要素。
7: %%%▼第2ネストレベルの\begin{enumerate}です。
8: %%\begin{enumerate}
9: %%%\item{\tt\itemize}環境のように、ネスティングも可能。
10: %%%\begin{itemize}
11: %%%\item{\tt\enumerate}環境のなかに\itemize環境をつく
12: %%%ることもできる。
13: %%%▼第3ネストレベルの\begin{enumerate}です。
14: %%%\begin{enumerate}
15: %%%\itemもちろん、逆に\itemize環境から\
16: %%%\tt\enumerate環境をネストすることもできる。
17: %%%\item第3レベルの\roman{enumiii}番目の要素。
18: %%%▼第4ネストレベルの\begin{enumerate}です。
19: %%%\begin{enumerate}
20: %%%\renewcommand{\labelenumiv}{No.{\bf\arabic{enumiv}}}
21: %%%\itemネスティングは第4レベルまで可能。
22: %%%\itemこのとき、\itemizeなど、他の環境のネストレ
23: %%%ベルとは無関係である。
24: %%%\end{enumerate}
25: %%%▲第4ネストレベルの\end{enumerate}です。
26: %%%\end{enumerate}
27: %%%▲第3ネストレベルの\end{enumerate}です。
28: %%%\end{itemize}
29: %%%\item第2レベルの\Roman{enumii}番目の要素。
30: %%%\end{enumerate}
31: %%%▲第2ネストレベルの\end{enumerate}です。
32: \item第1ネストレベルの\arabic{enumi}番目の要素。
33: \end{enumerate}
34: %%%▲\itemize環境を終了するコントロール・シーケンスが
35: %%%\end{enumerate}です。

```

`enumerate` 環境では、要素番号の出力を制御するためにカウンタが用意されていて、そのカウンタ名はネストレベル 1 から順に、`enumi`, `enumii`, `enumiii`, `enumiv` となっています。`enumerate` 環境のラベル部分は、このカウンタの値を出力しているのですが、その出力形式は好みに応じて変更できます。

たとえば、サンプルソース List 4-6 の 20 行目がそれにあたり、ここでは第 4 レベルの `enumerate` 環境のラベルを、ボールドフェイスのアラビア数字 (1, 2, 3, ...) に「No.」という文字列を加えて出力するように指示しているのです。

ここで、もし出力形式を大文字のローマ数字 (I, II, III, ...) に変更したい場合には、以下のようにしてください。

```
\renewcommand{\labelenumiv}{\Roman{enumiv}}
```

コントロール・シーケンス `\Roman` には、実在するカウンタを引数として、当該カウンタに現在登録されている値を大文字ローマ数字で出力する働きがあります。L^AT_EX には、カウンタを任意の形式で出力できるよう、Table 4-8 で示すコントロール・シーケンスが用意されています (List 4-6 の 5, 6, 17, 20, 29, 32 行目で使用)⁵⁾。

Table 4-8 ● カウンタ値の出力を行うコントロール・シーケンス

コントロール・シーケンス	形式	カウンタ (<i>counter</i>) の出力例
<code>\arabic{counter}</code>	アラビア数字	..., -2, -1, 0, 1, 2, ...
<code>\roman{counter}</code>	小文字のローマ数字	i, ii, iii, iv, v, ...
<code>\Roman{counter}</code>	大文字のローマ数字	I, II, III, IV, V, ...
<code>\alph{counter}</code>	小文字のアルファベット	a, b, c, d, e, ...
<code>\Alph{counter}</code>	大文字のアルファベット	A, B, C, D, E, ...
<code>\fnsymbol{counter}</code>	脚注用特殊記号	*, †, ‡, §, ¶, , **, ††, ‡‡

なお、変更したいラベルが第 1 レベルのものである場合には、20 行目の記述において `\labelenumiv` の部分を `\labelenumi` に、第 2 レベルの場合には `\labelenumii` に、第 3 レベルの場合には `\labelenumiii` に変更して、ラベルを変更したいネストレベルの簡条書き要素の前に記述してください。記述以降の当該レベルのラベルが変更されます⁶⁾。

◆ description 環境

`description` 環境は、簡条書きで列挙する各要素に見出し語をつける環境です。見出し部分は、英数記号を太字で、日本語をゴシック体で出力します。`itemize` 環境でも同じような記述が可能です。出力が若干異なりますので、その点に注意して比較してみてください。

5) なお、`\fnsymbol` の引数は 9 以下でなければなりません。

6) この変更指定を `enumerate` 環境に入ってから行った場合、変更はそのネストレベルに対応する `\end{enumerate}` が最初に記述されているところまで有効です。詳しくは第 4.4.7 項「“グルーピング”にかかわる構造」(p.132)で述べます。

第 1 要素 見出し部分には、このように本文とは異なる変化がつけられる。

い これは第 2 ネストレベルの簡条書き要素である。
ろ 同じような出力は、実は `itemize` 環境によっても実現できる。

i-1 このネストは、`itemize` 環境を使用している。

i-2 `description` 環境による出力と比較してほしい。

d-1 `description` 環境の第 3 ネストレベルである。

d-2 直前の `itemize` 環境とは、見出し語の出力位置が異なることに注意。

第 4 ネスト ほかの簡条書き環境と同じく、第 4 ネストレベルまで実現することができる。

は `description` 環境の第 2 ネストレベル。

第 2 要素 これは第 1 ネストレベル。

この出力を得るためのソースを List 4-7 に示します。

List 4-7 ● `description` 環境のサンプルソース

```

1: %%%▼_description_環境をはじめるコントロール・シーケンスが
2: %%%_begin{description}_です。
3: \begin{description}
4: _%%▼_簡条書きの要素を示すコントロール・シーケンスが\_item_です。
5: _\item[第1~要素] 見出し部分には、このように本文とは異なる変化がつけ
6: _られる。
7: %%%▼_第2_ネストレベルの\_begin{description}_です。
8: %%%_begin{description}
9: %%%\_item[い] ここは第2~ネストレベルの簡条書き要素である。
10: %%%\_item[ろ] 同じような出力は、実は\_itemize_環境によって
11: %%%_も実現できる。
12: %%%\_itemize_環境に見出し語をつけてみます。
13: %%%_begin{itemize}
14: %%%\_item[i-1] このネストは、\_itemize_環境を使用している。
15: %%%\_item[i-2]{\_description}_環境による出力と比較してほしい。
16: %%%\_end{itemize}
17: %%%▼_第3_ネストレベルの\_begin{description}_です。
18: %%%_begin{description}
19: %%%\_item[{\_d-1}]\_description_環境の第3~ネスト
20: %%%_レベルである。
21: %%%\_item[{\_d-2}]_直前の\_itemize_環境とは、見出し語の出力位
22: %%%_置が異なることに注意。
23: %%%▼_第4_ネストレベルの\_begin{description}_です。
24: %%%_begin{description}
25: %%%\_item[第4~ネスト] ほかの簡条書き環境と同じく、第4~ネストレベ
26: %%%_ルまで実現することができる。
27: %%%\_end{description}
28: %%%▲_第4_ネストレベルの\_end{description}_です。
29: %%%\_end{description}
30: %%%▲_第3_ネストレベルの\_end{description}_です。
31: %%%\_item[は]{\_description}_環境の第2~ネストレベル。

```

```

32: \end{description}
33: \item[第2~要素] これは第1~ネストレベル。
34: \end{description}
35: \end{description}
36: \end{description}
37: \end{description}

```

description 環境の見出し語は省略することもできますが、見かけは必ずしもよくありません。

4.4.4 その他の文章構造

前項までに述べてきた以外にも、 \LaTeX には文章構造を表すさまざまなコントロール・シーケンスや環境が用意されています。本項では、特に使用頻度が高いと思われるものに絞って概説を行います。

◆ 強調: `\em`

文章の強調を行うためのコントロール・シーケンスです。

このコントロール・シーケンス以下に続く文章は、強調の扱いを受けます。具体的にどう「強調」されるかというと、標準の環境でこのコントロール・シーケンスを使用すると、通常はローマン体で出力される英数字および記号がイタリック体に変更されます。

なお、強調部分を特定したい場合には、以下のようにグルーピング (後述⁷⁾) を利用することで、*text* だけを強調の扱いにすることができます。

```
{\em text}
```

しかし、ここで注意しなければならないことは、`\em` は使用するフォントをイタリック体に変更するコントロール・シーケンスではないということです。たとえば、次のソースおよびその出力⁸⁾を見れば、理解してもらえると思います。

List 4-8 ● 強調を行う構造のサンプル

```

1: By contrast, \em our General Public License is intended to
2: \em guarantee your freedom to share and change \} free
3: software--\em to make sure the software is
4: \em free for all its users \} \} \} .

```

以上のソース List 4-8 の出力サンプルが次に示すものです。

By contrast, *our General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users.*

7) 第 4.4.7 項「“グルーピング”にかかわる構造」(p.132)です。

8) 文章自体は『GNU GENERAL PUBLIC LICENSE Version 1, February 1989』の Preamble から引用しています。いうまでもないことですが、ここで「強調」を行っている部分は本章の筆者の独断によるもので、正式なものではありません。

一見すればわかると思いますが、「強調」のなかで「強調」を行うことも可能になっています。つまり、端的にはこのコントロール・シーケンスが使用されるたびに、英数字および記号がローマン体とイタリック体とトグルスイッチ式に「今、使用されているものとは異なるフォント」へと切り替わっているのです。

強調したい場合には、今まで利用していたフォントとは異なるフォントを使用すればよいのですが、WYSIWYG 方式のツールに慣れた発想から考えるとおもしろい考えといえます。この例などは、 \LaTeX が文章の論理構造をデザインするためのものであることを示す典型的な事例といえるかもしれません。

◆ 脚注：\footnote

脚注を付記するためのコントロール・シーケンスです。

脚注は、本文中の用語や文章に対して、ページの下などの余白部分を用いて補足説明を行うときに使用するもので、本文中の用語や文章と余白欄の説明文に同じ記号や番号をつけることで、その対応を明らかにします。

L^AT_EX では、脚注を入れたい部分に以下に示す書式でコントロール・シーケンスを埋め込むと、*text* が脚注となります。

`\footnote{text}`

使用例は List 4-1 (p.81) の 32 行目にありますから参考になさってください。

なお、基本的に `\footnote` は、文章を出力するパラグラフモードでしか使用できません⁹⁾。そこでもし、たとえば LR モードで脚注を利用したい場合には、LR モード中で `\footnotemark` により脚注記号 (Table 4-9 における “12) ” および “13) ”) だけを出力¹⁰⁾しておき、パラグラフモードに戻ってから `\footnotetext{text}` によって脚注文章 *text* を出力¹¹⁾しなければなりません。以下に、表中で脚注を使用するサンプルのソース List 4-9 と、その出力 Table 4-9 を示してみましょう。

List 4-9 ● LR モードでの脚注出力のサンプルソース

```

1: {\small
2: \addtocounter{footnote}{-1}%
3: \footnotetext{日本やアメリカで主に採用されているアメリカ式ポイン
4: ト制の基準単位。}%
5: \addtocounter{footnote}{-1}%
6: \footnotetext{ヨーロッパで主に採用されているディドポイント製の基準単位。}%
7: \addtocounter{footnote}{-2}%
8: \begin{tabular}[c]{l}\hline
9: \multicolumn{3}{|c|}{《日本語\text{\tiny TeX}における単位記号》} \\ \hline\hline
10: ptとポイント\footnotemark、1pt=0.3515mm=65536sp\\
11: spとスケールドポイント、1sp=0.5362\times$10$\%^{-5}$mm\\
12: mmとミリメートル、1mm=0.1cm=0.2834pt\\
13: cmとセンチメートル、1cm=10mm=28.34pt\\
14: inとインチ、1in=25.4mm=72.27pt=72bp\\
15: bpとビッグポイント、1bp=0.139in\\
16: pcとパイカ、1pc=12pt=4.218mm\\
17: ccとセシロ\footnotemark、1cc=12.835pt=4.511mm=12dd\\
18: ddとディドポイント、1dd=0.0833cc=0.376mm\\
19: emとmultiplecolumn{2}{1}|現在の有効なフォントの大文字Mの幅}\\
```

9) 例外として、`minipage` 環境があります。この環境内では LR モードや数式モードでも `\footnot` が利用できます。なお、`minipage` 環境は、通常のページ出力中に、あたかも別の小さなページがあるかのようにページをすることができず環境です。たとえば、この傍注を見てください。傍注の部分は、ページ内に存在する独立した 1 ページのように見えます。`minipage` 環境は、このようにページと同じような動作をする、任意の大きさのボックス (空間) を作成する環境なのです。

10) このとき、オプション引数を省略（通常時）すれば、脚注の参照番号を管理するカウンタ `footnote` が 1 増えます。

11) この場合、カウンタ
footnote は変化しま
せん。


```

20: \ex\&\multicolumn{2}{l}{現在有効なフォントの小文字xの高さ}\
21: \zw\&\multicolumn{2}{l}{現在有効な日本語フォントの文字の幅}\
22: \zh\&\multicolumn{2}{l}{現在有効な日本語フォントの文字の高さ}\hline
23: \end{tabular}
24: }

```

このソースの出力が Table 4-9 です。

Table 4-9 ● LR モードにおける脚注のサンプル

《 日本語 T _E X における単位記号 》		
pt	ポイント ¹²⁾	1pt = 0.3515mm = 65536sp
sp	スケールドポイント	1sp = 0.5362×10 ⁻⁵ mm
mm	ミリメートル	1mm = 0.1cm = 2.834pt
cm	センチメートル	1cm = 10mm = 28.34pt
in	インチ	1in = 25.4mm = 72.27pt = 72bp
bp	ビッグポイント	1bp = 0.139in
pc	パイカ	1pc = 12pt = 4.218mm
cc	シセロ ¹³⁾	1cc = 12.835pt = 4.511mm = 12dd
dd	ディドポイント	1dd = 0.0833cc = 0.376mm
em	現在有効な欧文フォントの大文字 M の幅	
ex	現在有効な欧文フォントの小文字 x の高さ	
zw	現在有効な日本語フォントの文字の幅	
zh	現在有効な日本語フォントの文字の高さ	

12) 日本やアメリカで主に採用されているアメリカ式ポイント制の基準単位。

13) ヨーロッパで主に採用されているディドポイント制の基準単位。

脚注についてもっと詳しく知りたい人は、『文書処理システム L^AT_EX』(アスキー刊) など、他の文献をあたってください。

4.4.5 ソースに忠実な出力をする構造

ところで、L^AT_EX は理工系の人々が特に好んで使用する傾向がありますから、たとえば文章中にプログラムのソースリストを掲載したいというような場合も当然考えられます。

L^AT_EX のソースを記述する場合に気をつけなければならない文字については、すでに「特殊な機能を持つ文字」(p.98)で説明しましたから、本項に至るまでの知識があればソースリストを L^AT_EX で出力することは必ずしも不可能ではないでしょう。

たとえば(プログラムソースではありませんが)、L^AT_EX のコントロール・シーケンス「\documentstyle[b5j]{jarticle}」を出力したい場合、文字列の途中には「注意すべき文字」である、\, {, } が交ざっています。しかし、先に述べたように、これらの文字を出力する場合は、すでに「特殊な機能を持つ文字」(p.98)で解説したとおりです。具体的には、以下の記述で「\documentstyle[b5j]{jarticle}」という出力が得られます。

```
\tt\symbol{'134}\documentstyle[b5j]{jarticle\}
```

しかし、ここで使用したソース例は 1 行に収まるような短いものだったのでたやすかったのですが、複数行に及ぶようなある程度長いソースファイルを考え

ると課題が残ります。なぜなら \LaTeX は、独自の改行処理、たとえばソースの〈改行〉が出力の改行であるとはかぎらなかったり、〈空行〉によって改行（正確には改段落）すれば行頭でインデント（字下げ）を行ったりという処理を自動的に行うからです。このことから、本項までの知識だけで「ソースリストを、ソースリストらしい文書体裁で出力すべく \LaTeX ソース化する」には、その修正がひと苦勞だということがわかるでしょう。

このような文書体裁にかける苦勞をユーザが背負うことは再三述べている \LaTeX の哲学に反しますから、 \LaTeX には与えられた文字列を記述されたまま出力するための構造が用意されています。本項では、この構造を紹介しましょう。

ただ気をつけなければならないことは、「記述されたまま」とはいても、

- ソース中の半角の英数字および記号は、タイプライタ体 (\tt) で出力される。
- ソース中の全角文字は、出力では明朝体 (\mc) で出力する。

ということです。

したがって、たとえばソースに含まれた半角英数字および記号が、全角文字の半分の幅で出力されるというわけではありませんし、半角カナ文字を出力することができるといってもありません。

しかしながら、その他の文字、たとえば特殊な機能を持つためにソース中にそのまま記述できない文字 \backslash , $\#$, $\$$, $\%$, $\&$, \wedge , $_$, \sim , $\{$, $\}$ もソースに忠実に出力されます。これが、本項における「記述されたまま」あるいは「ソースに忠実」という言葉の意味です。

◆ \verb

先に示した「 $\text{\documentstyle[b5j]{jarticle}}$ 」のように、〈改行〉を含まない文字列を、ソースに出力するための構造が、このコントロール・シーケンス \verb です。たとえば「 $\text{\documentstyle[b5j]{jarticle}}$ 」という出力は、「 $\text{\verb+\documentstyle[b5j]{jarticle}+}$ 」というソースから得られます。

前段落のソースを List 4-10 に示しましょう。

List 4-10 ● コントロール・シーケンス \verb のサンプルソース

- 1: 先に示した「 $\text{\verb+\documentstyle[b5j]{jarticle}+}$ 」のように、
- 2: $\text{\mbox{〈改行〉}}$ を含まない文字列を、ソースに忠実に出力するための構造が、
- 3: このコントロール・シーケンス $\text{\verb+\documentstyle[b5j]{jarticle}+}$ です。
- 4: たとえば「 $\text{\verb+\documentstyle[b5j]{jarticle}+}$ 」という出力は、
- 5: 「 $\text{\verb+\documentstyle[b5j]{jarticle}+}$ 」というソース
- 6: から得られます。

以下の書式を指定すれば、〈改行〉を含まない *word* を、記述されたままに出力します。このコントロール・シーケンスを使用した場合には、出力される文字列 *word* を、そのまま文章の一部として使用することができます。

\verb+word+

\verb*+word+

◎ L^AT_EX について その 2 ～L^AT_EX の使い方～

Lamport 氏が L^AT_EX にこめた設計哲学を考えれば、「L^AT_EX を使える」ということは、L^AT_EX のマクロやスタイルファイルを自在に作成でき、望むがままの体裁の文書を作り出せることではないことに気づきます。

それはむしろ逆で、「L^AT_EX を使える」ということは、文章の内容を論理的に構成し、その論理的な構成が反映されるように的確にコントロール・シーケンスを埋め込むことができるということにほかなりません。いいかえれば、「L^AT_EX を使う」場合、ユーザは論理的な文章作成と、その反映としてのコントロール・シーケンスの打ち込みに専念し、体裁にかかわる作業のほとんどすべてを L^AT_EX に任せてしまう方向に

そが、正しい使い方といえるのです。

もちろん、L^AT_EX のマクロやスタイルファイルを自由に作成することができるなら、それはそれですばらしいことであり、素敵なことだといえます。しかし、あくまでそれは二次的なものであって、L^AT_EX を使用するための決定的な要請ではないことに留意しなければなりません。

その意味では、一般に敷居が高いといわれる T_EX のうち、L^AT_EX に関しては、むしろコンピュータについての知識がない人々にとってこそ敷居が低く、理数系の人のみならず、およそ論理的な文章を書こうとするすべての人にとってこそ最も効率的に用いられるべきものといえるでしょう。

(p.136 へ続く)

なお、*word* の前後に記述する “+” は、必ずしも + 記号である必要はありません。*word* の前後に “*” 以外で、かつ *word* に含まれない文字を置けばよいのです。ここで、* 以外でなければならないという理由は、`\verb*` との区別をつけるためです。一般に、“+” 以外には “|” が多用されています (List 4-10)。

ところで、`\verb` と `\verb*` の違いは、*word* に含まれる〈空白〉に対する処理の違いによるものです。`\verb` では「 」のように単にスペースを空けますが、`\verb*` では「□」のように表現されます。

また、コントロール・シーケンス `\verb` および `\verb*` は、他のコントロール・シーケンスの引数として利用することはできない¹⁴⁾ ので、注意してください。

14) 他の環境のなかでは使用することができます。

◆ verbatim 環境

コントロール・シーケンス `\verb` と異なり、複数行にわたるテキストを忠実に出力することに適した環境が `verbatim` 環境です。`\verb` は引数 *word* に〈改行〉を含むことができませんでしたが、`verbatim` 環境では〈改行〉も忠実に処理されます。

たとえば、「L^AT_EX ソースの作成」に挙げた L^AT_EX のサンプルソース List 4-1 (p.81) の冒頭部分を何行か、以下に記述してみます。


```
% 文書のスタイルを指定します。
\documentstyle[b5j]{jarticle}

% タイトルを定義します
\title{X680x0 について}
\author{Chikumi Yoshino}
\date{1993/12/27}

% ここから実際に文書が始まります
\begin{document}
% タイトルを出力します。
\maketitle
\section{X680x0 の現状}
% 参照用のラベル
\label{1st_section}
1993 年には、待望久しかった X68030 が発売されました。X68030 は
X68000 シリーズの最高峰として存在するマシンですが、基本的な設計においては
従来機種との違いはなく、単に CPU をより高速な MC680EC30 に置き換えて
バス幅を 32bit に拡張しただけのものとして考えることもできます。
そういった観点からは、この X68030 はなんら新しい部分がない魅力のない
機械であるといった批判を免れないのは事実です。
```

この出力は、以下のソースによって得られました。

List 4-11 • verbatim 環境のサンプルソース

```
1: \begin{verbatim}
2: % 文書のスタイルを指定します。
3: \documentstyle[b5j]{jarticle}
4:
5: % タイトルを定義します
6: \title{X680x0について}
7: \author{Chikumi Yoshino}
8: \date{1993/12/27}
9:
10: % ここから実際に文書が始まります
11: \begin{document}
12: % タイトルを出力します。
13: \maketitle
14: \section{X680x0の現状}
15: % 参照用のラベル
16: \label{1st_section}
17: 1993年には、待望久しかったX68030が発売されました。X68030は
18: X68000シリーズの最高峰として存在するマシンですが、基本的な設計においては
19: 従来機種との違いはなく、単にCPUをより高速なMC680EC30に置き換えて
20: バス幅を32bitに拡張しただけのものとして考えることもできます。
21: そういった観点からは、このX68030はなんら新しい部分がない魅力のない
22: 機械であるといった批判を免れないのは事実です。
23: \end{verbatim}
```

なお、コントロール・シーケンス `\verb` に〈空白〉を「」で表現するタイプのもの (`\verb`) と「」で表現するタイプのもの (`\verb*`) があったように、`verbatim` 環境にも `verbatim*` 環境があります。使い方は、`verbatim` 環境の場合

15) たとえば、脚注 `\footnote` の引数として利用することはできません。

16) たとえば、アスキーが開発したマクロ集 `ascmac.sty` で定義された `screen` 環境のなかで使ったのが、List 4-11 の出力サンプルです。

17) 添付ディスク 7 の `\sample` ディレクトリ配下にある各種アーカイブです。

合と同じです。

コントロール・シーケンス `\verb` および `\verb*` と同様に、`verbatim` 環境および `verbatim*` 環境もほかのコントロール・シーケンスの引数¹⁵⁾ として利用することはできません。ただし、他の環境のなかでは使用することができます¹⁶⁾。

4.4.6 数式にかかわる構造

本節冒頭、第 4.4 節「論理構造を表す環境」(p.112) で述べたように、 \LaTeX において、数式の表現は「数式モード」と呼ばれる特殊な動作環境において行われます。数式モードは、文字どおり数式の表現を行うための動作環境です。それにとまって、パラグラフモードなどでは出力できない多彩な数学記号や特殊記号を利用することができるようになっていきますから、特に数式を扱わない人でも使用することは多いでしょう。

本項ではごく簡単に、数式モードへの移行のしかたについてのみ概説を行います。実際の数式の記述法については、その他の文献や、第 6 章「 \TeXfamily 」(p.229) で使用されている出力サンプルのソース¹⁷⁾ のうち数式を使用しているものを参照してください。

◆ インライン数式

インライン数式とは、文章中あるいは表中で比較的簡単な数式を記述する場合に使用する形式です。たとえば、「 $\Gamma \sin(2\pi + \theta) = \Gamma \sin \theta$ 」の、「」のなかがインライン数式に該当します。

前の段落を出力するためのソースを、以下の List 4-12 に示してみましょう。

List 4-12 ● インライン数式のサンプルソース

- 1: インライン数式とは、文章中あるいは表中で比較的簡単な数式を記述する場合に
- 2: 使用する形式です。
- 3: たとえば、「`\Gamma\sin(2\pi+\theta)=\Gamma\sin\theta`」の、
- 4: 「」のなかがインライン数式に該当します。

\LaTeX において、インライン数式モードを使用するための書式は以下に示すとおりです。

```
$numerical_formula$
```

```
\(numerical_formula\)
```

```
\begin{math}numerical_formula\end{math}
```

`numerical_formula` に数式を記述しますが、紙面の都合から、本書では説明しません。先に述べたように、その記述方法については各参考文献を参照してください。

なお、インライン数式モードを利用する各環境の違いは、 \LaTeX の内部的な処理の違いにすぎないと思ってもらってかまいません。ただ、 \LaTeX の初級者には

\$\cdots\$ の使用を勧めます。紙面の余裕がありませんので説明は省略しますが、一言でいえば、「\$ \cdots \$ 以外の書式には使用できない場面がある」からです。たとえば、`\section` の引数のなかで `\(\cdots \)` を使用するとエラーが起こります。

◆ ディスプレイ数式

インライン数式が文章中あるいは表中で簡単な数式を記述するとき使用する形式であったのに対して、ディスプレイ数式環境は、比較的長い数式を文章から独立させて出力する場合に使用するものです。したがって、ディスプレイ数式環境はパラグラフモードからしか使用できません。

具体的には、インライン数式のサンプル $\Gamma \sin(2\pi + \theta) = \Gamma \sin \theta$ をディスプレイ数式環境で出力してみると、以下のようになります。

$$\Gamma \sin(2\pi + \theta) = \Gamma \sin \theta$$

また、数式番号をともなった形式の出力も可能です。

$$\sum_{k=1}^n n = \frac{n(n+1)}{2} \quad (1)$$

このように、ディスプレイ数式は文章から独立して数式が出力されます。

前段落を出力するためのソースを、以下の List 4-13 に示しておきましょう。

List 4-13 ● ディスプレイ数式のサンプルソース

- 1: 具体的には、インライン数式のサンプル
- 2: `\Gamma\sin(2\pi+\theta)=\Gamma\sin\theta` を
- 3: ディスプレイ数式環境で出力してみると、以下のようになります。
- 4: `$$\Gamma\sin(2\pi+\theta)=\Gamma\sin\theta$$`
- 5:
- 6: また、数式番号をともなった形式の出力も可能です。
- 7: `\begin{equation}`
- 8: `\sum_{k=1}^n n=\frac{n(n+1)}{2}`
- 9: `\end{equation}`
- 10: このように、ディスプレイ数式は文章から独立して数式が出力されます。

L^AT_EX において、ディスプレイ数式モードを使用するための書式は以下のとおりです。

```
$$numerical_formula$$
```

```
\[numerical_formula\]
```

```
\begin{displaymath}numerical_formula\end{displaymath}
```

```
\begin{equation}numerical_formula\end{equation}
```

```
\begin{eqnarray}numerical_formula\end{eqnarray}
```

`$$\cdots$$`, `\[\cdots \]`, `displaymath` 環境については、ほとんど違いはありません¹⁸⁾。`equation` 環境は、数式を参照番号付きで出力できるもので、これは先にサンプルを示したとおりです。`eqnarray` 環境は、数式を縦揃えて出力することができます。

18) ディスプレイ数式の上下に空ける空白の大きさが異なります。詳しく知りたい人は試してみてください。

4.4.7 “グルーピング”にかかわる構造

L^AT_EX では、コントロール・シーケンスを使用する場合や、独自のコントロール・シーケンスや変数を定義する場合に、その作用範囲や存在範囲を限定することができます。

ここでは、「itemize 環境」(p.119)で紹介した箇条書きの構造においてラベル記号を変更する例を用いて説明してみましょう。

まず、List 4-14 とその出力をご覧ください。

List 4-14 ● グルーピングの例 その1

```

1: \begin{itemize}
2:   \item 標準の状態
3:   %%ラベル記号の変更
4:   \renewcommand{\labelitemi}{$\bigcirc$}
5:   \item コントロール・シーケンスの作用範囲内
6: \end{itemize}
7: \begin{itemize}
8:   \item コントロール・シーケンスの作用範囲外
9: \end{itemize}
```

以下に示すのが List 4-14 の出力サンプルです。

- 標準の状態
- コントロール・シーケンスの作用範囲内
- コントロール・シーケンスの作用範囲外

次に、以下に示すソースおよび出力サンプルを、今見ていただいた List 4-14 およびその出力と比較してみてください。

List 4-15 ● グルーピングの例 その2

```

1: \begin{itemize}
2:   \item 標準の状態
3: \end{itemize}
4: %%ラベル記号の変更
5: \renewcommand{\labelitemi}{$\bigcirc$}
6: \begin{itemize}
7:   \item コントロール・シーケンスの作用範囲内
8: \end{itemize}
9: \begin{itemize}
10:   \item コントロール・シーケンスの作用範囲内
11: \end{itemize}
```

- 標準の状態

- コントロール・シーケンスの作用範囲内
- コントロール・シーケンスの作用範囲内

違いがおわかりでしょうか？ 環境内で使用されたコントロール・シーケンスは、環境内にしか作用していないはずで、具体的には、以下のことがわかります。

- ラベルの変更を `itemize` 環境に入ってから行っている List 4-14 の場合、この変更は利用している環境内で一時的に作用する。
したがって、いったん `itemize` 環境から抜け出して、もう一度 `itemize` 環境に入っても、先ほどの変更はされない。
- ラベルの変更を `itemize` 環境に入る前に (親の環境で) 行っている List 4-15 の場合、この変更は親の環境下にあるかぎり有効なものである。
したがって、いったん `itemize` 環境から抜け出して、もう一度 `itemize` 環境に入っても、先ほどの変更は有効なものとして機能する。

このような特性は、Human68k における子プロセスと環境変数との関係を考えてみるとわかりやすいかもしれません。

まず、`COMMAND.X` から、さらに `COMMAND.X` を実行して子プロセスに入ってしまったでしょう。このとき、親プロセスで定義されている環境変数は子プロセスでも受け継がれています (Fig. 4-5-(1))。そこで、子プロセスで親プロセスから受け継がれた環境変数を変更 (Fig. 4-5-(2)) したあと、`EXIT` コマンドでプロセスを抜け出した場合、親プロセスでは先に変更したはずの環境変数が元に戻っています (Fig. 4-5-(3))。さらに再び `COMMAND.X` を実行して子プロセスに入っても、親プロセスの環境変数がそのまま受け継がれ、先の変更が再現されるようなことはありません (Fig. 4-5-(4))。

Fig. 4-5 ● Human68k のプロセスと環境変数 〜その1〜

(1) 子プロセスに入る

```
A> set
TEXHOME=f:/tex
A> command.x
Command version 3.00
```

```
A> set
TEXHOME=f:/tex
```

(2) 子プロセスで環境変数を変更する

```
A> set TEXHOME=c:/usr/local/tex
A> set
TEXHOME=c:/usr/local/tex
```

(3) 子プロセスを抜ける

```
A> exit
A> set
TEXHOME=f:/tex
```

(4) 子プロセスに入る

```
A> command.x
Command version 3.00

A> set
TEXHOME=f:/tex
A> exit
```

しかし、親プロセスで先ほどの変更を行ってしまえば (Fig. 4-6-(1))、その後、子プロセスに何度出入りしても、親プロセスで変更した環境変数が受け継がれることになるのです (Fig. 4-6-(2), (3))。

Fig. 4-6 ● Human68k のプロセスと環境変数 〜その 2〜

```

(1) 親プロセスで環境変数を変更する
A> set TEXHOME=c:/usr/local/tex
A> set
TEXHOME=c:/usr/local/tex

(2) 子プロセスにおける環境変数の設定状況 〜その 1
A> command.x
Command version 3.00

A> set
TEXHOME=c:/usr/local/tex
A> exit

(3) 子プロセスにおける環境変数の設定状況 〜その 2
A> command.x
Command version 3.00

A> set
TEXHOME=c:/usr/local/tex
A> exit

```

L^AT_EX におけるコントロール・シーケンスも、この例ときわめて似た作用範囲を持っているといえます。

このように、L^AT_EX では各コントロール・シーケンスの作用範囲を特定することができます。ここでは環境による作用範囲の特定の例を示しましたが、これ以外に `{ }` で囲むことによって、コントロール・シーケンスの作用範囲を制御することができます。このような制御を行う典型例としては、フォントの種類やサイズを特定部分だけ変更させる場合が挙げられます。

たとえば、以下のサンプルソースにおいて `{\it···\}` および `{\bf···}` の部分がそれにあたります。

List 4-16 ● グルーピングの例 その 3

```

1: 1993年、ついに{\itX68000}シリーズの最高峰である{\bfX68030}が登場
2: した。また、それにともなつて{Human68k\}も{Ver.3}がリリースされている。

```

ここで“`\it`”はフォントをイタリック体に、“`\bf`”はフォントをボールド体(太字)に変更するためのコントロール・シーケンスのことです¹⁹⁾。本来、`\bf` や `\it` のようなフォントの変更を行うコントロール・シーケンスは後続のフォントをすべて変更しますが、このサンプルでは `{ }` で囲むことでコントロール・シーケンスの作用範囲を特定することに成功しています。次に示すのが List 4-16 の出力です。

19) フォントの変更を行うためのコントロール・シーケンスについては、まとめて後述します。

 ◎ L^AT_EX について その 3 ～コントロール・シーケンスの埋め込み～

L^AT_EX について、2 回にわたってコラムで述べてきました。しかし、「複雑きわまるコントロール・シーケンスの埋め込みを要求することは、論理的な文章作成にライターを集中させるどころか、逆にそれを疎外するのではないか」という疑問を持つ人もいるのではないかと思います。

たしかに、この疑問は一面において事実といえるでしょう。特に、WYSIWYG になじんでいた場合にはなおのこと、バッチ処理特有の「まどろっこしさ」に苛立ちすら覚えるに違いありません。

しかし、Lamport 氏は『文書処理システム L^AT_EX』において、こう述べています。

「自分のテキストの論理的な構成を L^AT_EX に知らせなければならないので、必然的に構成のしっかりしたテキストを作ろうとするようになる。」

L^AT_EX のコントロール・シーケンスは、文章の論理構造を出力に反映させるための

ものがほとんどですから、コントロール・シーケンスの埋め込みそのものが、論理的な文章作成のための推敲となるというわけです。

ただし、L^AT_EX のコントロール・シーケンスは、可読性が高くわかりやすいものが多い半面、長さが長いためにタイプミスの可能性が高くなります。そこで、巷には L^AT_EX のソース作成のためにカスタマイズを加えたエディタが多数存在 (第 6 章「T_EXfamily」で紹介) しており、Lamport 氏もこれらを利用することを勧めています。

ところで、ここまでの L^AT_EX に関する説明で「L^AT_EX はえらく“かたい”」あるいは「L^AT_EX では論文しか書けないのか」という印象をもった方、Lamport 氏は同書でこうもいっているのです。

「言葉というものはつねに論理的とはかぎらない。」

1993 年、ついに X68000 シリーズの最高峰である **X68030** が登場した。また、それにとまって *Human68k* も Ver.3 がリリースされている。

以上のようなグルーピングの性質については、正直なところ「習うより慣れよ」の言葉どおり、実際に経験を積み重ねたほうがわかりやすいでしょうから、いろいろ自分で試してみてください。

4.5 視覚構造を制御する

第 4.4 節「論理構造を表す環境」で、 \LaTeX は「論理構造を表現する」と述べました。基本的に文章の論理構造を表すコントロール・シーケンスだけを使用していく方向性が「より \LaTeX らしい使い方」なのですが、実際のところ、視覚的な問題にまったく手をつけずにすませることができる文章というのは、それほど多くありません。たとえば、「 \LaTeX で処理したところ、思うように出力されなかった」という場合、もっと具体的な例を挙げるなら、「警告 (Warning)」(p.84) で解説した `Overfull \hbox` のようなメッセージが出力される場合などは、その典型といえるでしょう。その場合、やむを得ず¹⁾、視覚構造を制御する類のコントロール・シーケンスを利用することになります。

ある意味では、結果的にこうして文書構造を制御するコントロール・シーケンスと視覚構造を制御するコントロール・シーケンスとが入り乱れてしまうことが、 \LaTeX のソースをわかりにくくしてしまう原因であるのかもしれません。

それはともかくとして、 \LaTeX では、「処理したところ、思うように出力されなかった」という場合のような消極的な要因だけではなく、もっと積極的に視覚構造の制御を行うことも可能になっています。

文中のある部分を特に強調する必要からフォントを変更したい、または、同様の理由から今度は文字を大きくしたい、あるいは、逆に補足的に記述する必要から文字を小さくしたい — \LaTeX を使用していれば、むしろそのような欲求は高まるからです。

まして、 \TeX の特徴のひとつは、その多彩かつ美しいフォントにあります。したがって、 \LaTeX においてもこれを活用して、より読みやすい文書をつくり出そうと考えることは、ごく当然のことでしょう。

本節では、主に文字に着目して、視覚構造を制御するための方法について概説することにします。

4.5.1 文字の大きさを変更する

本項では、まず \LaTeX における文字の大きさの変更について、簡単に触れてみます。

1) このような書き方をすると、「視覚的制御は使わないほうがよい」というニュアンスに取れると思います。その受け取り方は間違いでも誤解でもありません。「視覚に関する制御をできるだけ避ける」というのが、 \LaTeX の「まっとうな」使い方です。

以下、編集部注：このように述べる本章の著者が、「最後に頼るのは力業だよな」とぼやきながら、`plain \TeX` のコントロール・シーケンスで本書の体裁上の最終調整をしていたことは、注目して然るべきであろう。

◆ 文字の大きさについて

活字の世界では、文字の大きさを表すためにポイント (point. pt, ポとも略す) という単位を使用するのが一般的です。1pt = $1/72.27$ in = 0.3514mm で、T_EX で使用する場合には pt という単位記号で表します。これをアメリカ式ポイント制 (Anglo-American point system) といい、日本でもこの方式が採用されています。

また、イギリス以外のヨーロッパでは、1cc (シセロ, cicero) = 12dd (ディドポイント, didot point) = 4.511mm = 12.835pt \doteq 1pc (パイカ, pica) = 12pt として定めるディドポイント制 (Didot point sysytem) が主流のようです。

このほかに、日本では写真植字の指定で使用される 1 級 (Q とも略す) = $1/4$ mm という単位もあります。

T_EX では、Table 4-10 に示すように、長さの単位として pt, dd, cc, pc などを使用することができますが、文字サイズを表すためには pt を使うのが最も一般的のようです。

Table 4-10 ● 日本語 T_EX における単位記号

記号	意味 および ほかの単位との換算	
pt	ポイント	1pt = 0.3515mm = 65536sp
sp	スケールドポイント	1sp = 0.5362×10^{-5} mm
mm	ミリメートル	1mm = 0.1cm = 2.834pt
cm	センチメートル	1cm = 10mm = 28.34pt
in	インチ	1in = 25.4mm = 72.27pt = 72bp
bp	ビッグポイント	1bp = 0.139in
pc	パイカ	1pc = 12pt = 4.218mm
cc	シセロ	1cc = 12.835pt = 4.511mm = 12dd
dd	ディドポイント	1dd = 0.0833cc = 0.376mm
em	現在有効な欧文フォントの大文字 M の幅	
ex	現在有効な欧文フォントの小文字 x の高さ	
zw	現在有効な日本語フォントの文字の幅	
zh	現在有効な日本語フォントの文字の高さ	

Table 4-11 ● “pL_AT_EX” で拡張された単位記号

記号	意味 および ほかの単位との換算	
H	歯	1H = 0.25mm = 0.7085pt
Q	級	1Q = 1H

◆ L_AT_EX における文字の大きさ

L_AT_EX には、文字の大きさを変更するために、いくつかのコントロール・シーケンスが用意されています。しかし、それらのコントロール・シーケンスで実際に出力される文字の大きさは、「つねに一定」というわけではありません。どう

◎ フォントの大きさとデザイン

TeX で表示される「12pt の roman フォント」と「10pt の roman フォントを 12pt に拡大したもの」とでは、同じ出力が得られるわけではありません。これは、METAFONT のフォントが特定のサイズで表示した場合に最も美しく見えるようにデザインされているからです。

つまり、5pt の Computer Modern Roman フォント “cmr5” は 5pt で見たときに最も美しく見えるように、10pt の Computer Modern Roman フォント “cmr10” は 10pt で見たときに最も美しく見えるようにつくられているのです。

しかし、Table 4-13 (p.143) に見られるように、フォントによっては特定のサイズしか用意されていないものもあります。たとえば Computer Modern Roman フォントの場合、5, 6, 7, 8, 9, 10, 12, 17pt だけしか用意されていません。

では、14pt のフォント (10pt をベース

にしたときの \Large) はどうやって出力するのでしょうか？

新たに 14pt 用にフォントをデザインすることが最も望ましいことはいうまでもありません。しかし、これではあまりにも面倒です。

通常は次善の策として、存在する「ある大きさのフォント」を拡大・縮小し、「それらしく」表示することによって対処しています。

このようなフォントの拡大処理を行うコントロール・シーケンスが \magstep です。

ところが、先ほどから述べているように、たとえば 10pt のフォントは、10pt で見たときに最高のバランスとなるようにデザインされていますから、10pt のフォントを拡大して 12pt で見たときには、12pt で見るためにデザインされたフォントのそれと比べると、若干バランスがくずれることになります。

ということかといいますと、スタイルオプションで 11pt あるいは 12pt などを指定している場合と、指定していない場合 (10pt) とでは、実際に使用されるフォントの大きさが変わってくるということです。

L^AT_EX の文字サイズを変更するコントロール・シーケンスによって設定されるフォントの大きさ、および改行幅 (\baselineskip) は Table 4-12 に示すとおりです。この表に示された文字の大きさが実際にどの程度の大きさであるかについては、本書の『Vol.2 — Reference 編』の巻末にフォントの出力サンプルが掲載されていますから、これを参照してください。

Table 4-12 ● L^AT_EX における文字の大きさ

コントロール・ シーケンス	文字の大きさ			改行幅		
	10pt	11pt	12pt	10pt	11pt	12pt
\tiny	5pt	6pt	6pt	6 pt	7 pt	7 pt
\scriptsize	7pt	8pt	8pt	8 pt	9.5pt	9.5pt
\footnotesize	8pt	9pt	10pt	9.5pt	11 pt	12 pt
\small	9pt	10pt	11pt	11 pt	12 pt	13.6pt
\normalsize	10pt	11pt	12pt	12 pt	13.6pt	14.5pt
\large	12pt	12pt	14pt	14 pt	14 pt	18 pt
\Large	14pt	14pt	17pt	18 pt	18 pt	22 pt
\LARGE	17pt	17pt	20pt	22 pt	22 pt	25 pt
\huge	20pt	20pt	25pt	25 pt	25 pt	30 pt
\Huge	25pt	25pt	25pt	30 pt	30 pt	30 pt

なお、これらの文字サイズの変更を行うためのコントロール・シーケンスを利

◎ em と ex

活字の世界にかぎらず、寸法単位には Table 4-12 に示すものがありますが、これらのなかには、およそ活字の世界にかかわる人でなければ使用しない、そしてなにより、必ずしも絶対的ではない寸法単位が含まれています。

ここで「絶対的ではない」という意味は、いうまでもなく「相対的だ」ということで、現在使用されているフォントの種類とサイズとに依存して、その寸法単位の絶対的な寸法を変化させるということです。

もう少し、この点について説明しておきましょう。1cm はどう操っても 1cm であり、同時に 10mm です。今 1cm の長さが、明日 1mm で表されるということは、「世界に存在する辞書のすべてを書き換えてもありえない」といっていいでしょう。

しかし、活字の世界では、「文字幅の半分だけ、出力を右に寄せたい」などという場合が往々にしてあります。そのために、使用中の文字に応じてその絶対的な長さを変える寸法単位がつくられたのであり、その代表例が em と ex なのです。

さて、なぜ「寸法単位が持つ絶対的な値」

が変化するのかというと、本来、em は現在のフォントにおける大文字の“M”の幅（つまり、「クワタ」あるいは「全角の幅」のこと）を、ex は現在のフォントにおける小文字の“x”の高さを表したからです。つまり、現在使用しているフォントを別のフォントに変更すると、今まで 1em だった長さ（仮に xpt としましょう）は、今度使用するフォントの大文字の“M”の幅が xpt でないかぎり、“1em”が示す長さではなくなってしまうのです。

ただし、現実には「“1em”は現在使用するフォントの大文字の“M”の幅を表す」という、相対的な寸法単位が持っていたはずの本来の意味は失われ、em や ex は、各フォントごとに定められた固有の値という以上の意味を持ちません。

さて、我々が T_EX の標準フォントというべき Computer Modern フォントの場合、各フォントは（一部を除いて）、ソース上“---”と書いたときに出力される「全角のダッシュ（—）」が幅 1 em に、数字の 0~9 が幅 0.5em に、小文字の x の高さが 1ex になるようにデザインされています。

用した場合、フォント自体は roman フォントおよび明朝体に戻ってしまいますから注意が必要です²⁾。このような場合、文字サイズの変更を行ったあとに、あらためてフォントの変更を行います。

2) 「フォントの種類」(p.140)のコラムで述べる NFSS では解消されています。

4.5.2 フォントを変更する

本項では、L^AT_EX 上で使用可能な多彩なフォントと、その使用法について概説を行います。

◆ フォントの種類

フォントの種類は、以下のように分類することができます。なお、この分類は、Mittelbach 氏と Schöpf 氏によって提唱され、L^AT_EX の Ver.3 で採用されるというフォント指定の方法“New Font Selection Scheme (NFSS)”をもとにしています。

◎ New Font Selection Scheme (NFSS)

従来の \LaTeX では、文字サイズ変更のコントロール・シーケンスを使用すると、日本語フォントは明朝体に、アルファベットのフォントはローマン体に戻ってしまいました。しかし、NFSS ではサイズ・ファミリー・シェイプ・ウエイト・ウィズが独立して管理されるために、文字サイズを変更しても、使用されていたフォントは継続して利用することが可能になっています。

もともと、NFSS は \LaTeX だけの専売特許ではありません。たとえば、 \AMS-L\TeX

にはすでに実装されています。試したい人は入手してみるとよいでしょう。あるいは、NFSS のパッケージも存在しますから、独自に組み込んでみるのもひとつの方法です。

ただし、これらは今のところ日本語には対応していませんので、現在、PC-VAN の SSCIENCE を中心に、NFSS を日本語に対応させたり、 \LaTeX Ver.3 のプロトタイプともいうべき \LaTeX をアスキーの日本語 \TeX で動作させたり、といったプロジェクトが積極的に進められています。

○ ファミリー (family)

ある特定の思想 (というのは大げさですが) のもとに統一したデザインで作成された複数のフォントのグループのことをいいます。

- シェイプ (shape) あるファミリーに属するフォントの形状の変化のことをいいます。

ローマン体 (roman) をはじめ、イタリック体 (*italic*)、スラント体 (*slant*) などが代表的でしょう。

- シリーズ (series)

あるファミリーのフォントについて、文字そのものの幅の広狭や、文字の線の太さなどの系統的な変化のことをいいます。

NFSS によれば、サイズ (size) もこの配下に位置すると思ってよいかもしれません。

- ウエイト (weight)

文字を描く線の太さを表すシリーズのことをいいます。

細いものから順に、ライト (light)、ミディアム (medium)、ボールド (bold) があり、これより細いもの、あるいは太いものはエクストラ (extra) などの形容詞を冠して表します。

- ウィズ (width)

あるファミリーに属するフォントの文字そのものの幅の広狭を表すシリーズのことをいいます。

狭いものからコンデンス (condensed)、ミディアム (medium)、エクスパンド (expanded, extended) があり、これより細いもの、あるいは太いものはエクストラ (extra) などの形容詞を冠して表します。

◆ CM Font Family

TeX で標準的に使用されているフォントは、TeX や METAFONT の生みの親である D.E.Knuth 博士の手によって作成された、「CM (Computer Modern) フォント・ファミリー」と呼ばれるものです。

CM フォント・ファミリーには Table 4-13 に示すものが用意されていますが、 \LaTeX では、これらのフォントを次の 3 種類に分類しています。

○ ロード済み (preload)

表中に「サンセリフ体」で示したサイズのフォントです。

名前をサンセリフ体で示したフォントは、「 \LaTeX に定義されているフォント」(p.144)で示すコントロール・シーケンスに定義されており、そのうちのサンセリフ体で示したサイズは、フォーマットファイル (fmt ファイル) に tfm ファイルが読み込まれています。

なお、フォントのサイズ欄に書かれている数字は読み込まれるフォントのサイズを表します。したがって、たとえば 17pt のサイズの欄に「10」とある場合には、それは 10pt のサイズのフォントを 17pt に拡大して使用することを示し、8pt のサイズ欄に「10」と書かれている場合には、それは 10pt のサイズのフォントを 8pt に縮小して使用することを示します。

○ 要求時ロード (loaded on demand)

表中に「(サンセリフ体)」で示したサイズのフォントです。

このサイズのフォントは、必要になったときに tfm ファイルが読み込まれるようになっています。

なお、フォントのサイズ欄に書かれている数字が当該フォントのサイズと異なっている場合については、ロード済みのフォントの場合と同じく、適当なサイズを流用することを表します。

○ 利用不可 (unavailable)

上記以外の通常書体で示されたフォントです。

これらのフォントは、「任意のフォントを得る」(p.144)で解説するように、明示的にフォントの使用を定義しなければ利用できません。

Table 4-13 • Computer Modern Font Family

ファイル名 のヘッダ	フォントネーム	サイズ (pt)												
		5	6	7	8	9	10	11	12	14	17	20	25	
cmb	CM bold roman						10							
cmbsty	CM bold math symbols						(10)	(10)	(10)	(10)				
cmbx	CM bold extended roman	(5)	(6)	(7)	(8)	9	10	10	12	10	10	(10)	(10)	
cmbxsl	CM bold extended slanted roman						10							
cmbxti	CM bold extended text italic						10							
cmcsc	CM caps and small caps				(10)	(10)	(10)	(10)	(10)	(10)	(10)	(10)		
cmdunh	CM dunhill roman						10							
cmex	CM math extension						10							
cmff	CM funny roman						10							
cmfi	CM funny italic						10							
cmfib	CM roman fibonacci font				8									
cminch	CM inch-high sans serif													
	bold extended caps and digits						10							
cmitt	CM italic typewriter						10							
cmmi	CM math italic	5	6	7	8	9	10	10	12	10	10	10		
cmmib	CM math italic bold						(10)	(10)	(10)	(10)				
cmr	CM roman	5	6	7	8	9	10	10	12	10	17	10	10	
cmssl	CM slanted roman				(8)	(9)	10	10	12	(10)	(10)	(10)		
cmstlt	CM slanted typewriter						10							
cmss	CM sans serif				(8)	(9)	10	10	12	(10)	(17)	(10)		
cmssbx	CM sans serif bold extended						10							
cmssdc	CM sans serif demibold condensed						10							
cmssi	CM slanted sans serif					9	10		12		17			
cmssq	CM sans serif quotation style				8									
cmssqi	CM sans serif quotation style slanted				8									
cmsy	CM math symbols	5	6	7	8	9	10	10	10	10	10	10		
cmtcsc	CM typewriter caps and small caps						10							
cmtext	CM T _E X extended ASCII characters				8	9	10							
cmti	CM text italic				7	8	9	10	10	12	(10)	(10)	(10)	
cmitt	CM typewriter text				(8)	9	10	10	12	(10)	(10)	(10)		
cmu	CM unslanted italic						10							
cmvtt	CM variable-width typewriter						10							

ところで、L^AT_EX では、CM フォントのほかに簡単な図表を作成するために、Table 4-14 に示すような METAFONT が拡張されています。表の見方は CM フォントの場合と同じです。

Table 4-14 • L^AT_EX Fonts

ファイル名 のヘッダ	フォントネーム	サイズ (pt)												
		5	6	7	8	9	10	11	12	14	17	20	25	
lasy	L ^A T _E X symbols	5	6	7	8	9	10	10	10	10	10	10		
lasyb	L ^A T _E X bold symbols						(10)	(10)	(10)	(10)				
lcircle	L ^A T _E X circle						10							
lcirclew	L ^A T _E X wide circle						10							
line	L ^A T _E X line						10							
linew	L ^A T _E X wide line						10							

また、日本語 L^AT_EX には、以下のような日本語フォントも用意されています。

Table 4-15 • 日本語フォント

ファイル名 のヘッダ	フォントネーム	サイズ (pt)											
		5	6	7	8	9	10	11	12	14	17	20	25
min	横書き用明朝体	5	6	7	8	9	10	10	10	10	10	10	10
tmin	縦書き用明朝体	5	6	7	8	9	10						
goth	横書き用ゴシック体	(5)	(6)	(7)	(8)	9	10	10	10	10	10	(10)	(10)
tgoth	縦書き用ゴシック体	5	6	7	8	9	10						

◆ \LaTeX に定義されているフォント

\LaTeX に定義されているフォントには、以下に示すものがあります。それぞれ先頭に示しているのは、そのフォントに切り替えるためのコントロール・シーケンスです。

- `\rm` : “Roman” font is “CM Roman”.
- `\bf` : “Bold Face” font is “CM Bold Extended Roman”.
- `\it` : “Italic” font is “CM Text Italic”.
- `\sl` : “Slanted” font is “CM Slanted Roman”.
- `\sf` : “Sans Serif” font is “CM Sans Serif”.
- `\tt` : “Type-Face” font is “CM Type Writer”.
- `\sc` : “SMALL CAPS” FONT IS “CM CAPS AND SMALL CAPS”.
- `\cal word$` :
“CALI GRAPH” FONT IS “CM MATH SYMBOLS”.
- `\boldmath$\cal word$\unboldmath` :
“BOLD CALI GRAPH” FONT IS
“CM BOLD MATH SYMBOLS”.
- `$\mit word$` : “Math Italic” font is “CM Math Italic”.
- `\boldmath$\mit word$\unboldmath` :
“Bold Math Italic” font is “CM Bold Math Italic”.

ただし、`\cal word $` および `\boldmath$\cal word $\unboldmath`, `$\mit word $`, `\boldmath$\mit word $\unboldmath` は、`word` の部分が当該フォントに変更されます。

なお、イタリック体やスラント体は右方向に傾いているために、直後に斜体ではない文字がくると文字間が詰まってしまいます。たとえば、`didn't` (`{\it did}n't`) のようにです。そこで、このように斜体になる可能性³⁾ があるフォントの直後には、「イタリック補正 (`\`)」と呼ばれる横方向の空白補正を行います。この補正を行った場合、先ほどの例は `didn't` (`{\it did\}n't`) のように出力されるようになります。

ところで、本書に添付されている日本語 \LaTeX には、日本語フォントとして以下に示すフォントの変更を行うためのコントロール・シーケンスが設定されています。

- `\mc` : 「明朝体」のフォントである。
- `\gt` : 「ゴシック体」のフォントである。

◆ 任意のフォントを得る

これまでに、 \LaTeX の標準の環境に設定されているコントロール・シーケンス

3) 明らかに斜体となるイタリック体 `\it` やスラント体 `\sl` はもとより、状況に応じて斜体となる強調 `\em` の場合にもつけるようにしましょう。

によって、文字サイズや文字フォントを変更する方法を示しました。しかし、場合によっては、 \LaTeX が標準で定義しているよりも大きな文字サイズが必要なこともあるでしょうし、「 \LaTeX に定義されているフォント」(p.144)に示したものと異なるフォントを使用したいという場合もあるでしょう。

このような場合のために、 \LaTeX には新たにフォントを定義して利用するためのコントロール・シーケンスが用意されています。

それが `\newfont` コントロール・シーケンスです。このコントロール・シーケンスは、以下に示すような書式で使います。

```
\newfont{\command}{fontname at ?pt}
```

`\newfont` は、`?pt` のフォント `fontname` を新たなコントロール・シーケンス `\command` として設定します。フォントサイズの指定の部分 (`at ?pt`) を省略した場合、選択したフォントの本来のフォントサイズ (`cmr10` なら `10pt`) が選択されます。

ここでは仮に、Computer Modern Dunhill Roman の `9pt` フォントを得たい場合を例にとり、解説を行うことにしましょう。CM Dunhill Roman 体は、「CM Font Family」(p.142)で示した Table 4-13 (p.143) のように `10pt` サイズしか用意されていません。そこで、ここでは既存の `10pt` のフォントファイルを縮小して `9pt` のフォントを得ることにします。

```
\newfont{\cmdunhnin}{cmdunh10 at 9pt}
```

以上の記述によって、ソース中で `9pt` の CM Dunhill Roman を出力するコントロール・シーケンス `\cmdunhnin` を利用できるようになります⁴⁾。

ただし、このようにして定義された任意のサイズの任意のフォントは、読み込まれたそのサイズでしか利用することができません。どういうことかという、 \LaTeX における文字の大きさ」(p.138)で示したような文字サイズを変更するコマンドによって、文字の大きさを変更することはできないということです。以下のサンプルソースとその出力を見比べてみれば、原因はともかく、その意味はわかるでしょう。

List 4-17 ● 任意のフォントを利用するためのサンプル

```
1: \newfont{\cmdunhnin}{cmdunh10 at 9pt}
2: ABC{\cmdunhten DEF}{\bf GHI}{\tiny\rm JKL}{\bf MNO}
3: {\cmdunhten PQR}STUVWX YZ.}
```

List 4-17 の出力が以下のサンプルです。

ABCDEF GHI JKLMNO PQRSTUVWX YZ.

4) \LaTeX での処理時にフォントのデータが記述された `tfm` ファイルが読み込まれますから、当該フォントの `tfm` ファイルが準備されていなければなりません。

4.5.3 フォントテーブルの利用

\LaTeX では、使用する機会が多い記号についてはコントロール・シーケンスに数学記号として定義されています。たとえば、「 α 」は `\alpha` という具合で

す。したがって、通常はこれらのコントロール・シーケンスを使用すればこと足るはずです。

しかし、すべての記号がコントロール・シーケンスに定義されているというわけではありませんし、定義されていたとしても数学記号として定義されているために数式モードでなければ使用することができないという制約もあります。このような場合には、以下に示すようにフォントテーブルを参照して新たにコントロール・シーケンスとして定義することによって、出力したい任意の文字を自在に使用することができるのです。

たとえば Computer Modern Typewriter Text (cmmt) において “\” を出力する場合、もともとの定義を利用することを考えれば、`\backslash$` で “\” を、あるいは `\verb+\"` で “\” を出力することができます。しかし、どちらもいささか冗長な記述ですし、`\verb` にいたっては他のコントロール・シーケンスの引数として使用できないという使用上の制約もあります。そのため、記号を多用する場合や使用する状況によっては便利とはいえません。

そこでこの場合、本書の『Vol.2 — Reference 編』の最終章に掲載されているフォントテーブルをもとに、新たなコントロール・シーケンスとして設定することで対応します。

まず『Vol.2 — Reference 編』のフォントテーブルを参照すると、“\” という文字が Computer Modern Typewriter Text においては 8 進数コードで '134⁵⁾、16 進数コードで "5c であることがわかります。そこで、コードで指定して文字を出力するためのコントロール・シーケンス `\symbol` あるいは `\char` を使用して、新たなコントロール・シーケンスとして登録するのです。

登録のしかたは、ソース中で `\def\bs{\tt\symbol{'134}}` あるいは `\def\bs{\tt\symbol{"5c}}` を定義することにより、`\bs` というコントロール・シーケンスで “\” を出力できるようになります。

ところで、新たに入手した METAFONT のフォントを出力する場合、フォントとフォントを出力させるためのコードとの対応を知ることが必要となります。つまり、フォントファイルにどのようなフォントが入っているのか、そして、それがどのコードに割り当てられているのかを知ることが肝心です。そのために、先ほどから利用しているようなフォントテーブルを作成しなければなりません。

ありがたいことに T_EX のシステムには、フォントテーブルをはじめ、各種のフォントにかかわる出力サンプルが得られるソースファイルが用意されており、これは本書が構築した T_EX システムでも当然サポートされています。具体的には、`%TEXHOME%\macros\testfont.tex`⁶⁾ が当該ソースファイルにあたります。このソースを plain T_EX で処理し、指示を入力していくことによって、各種のサンプルが得られるようになっています。本書の『Vol.2 — Reference 編』に掲載されているフォントテーブルは、このソースによって作成したものではありませんが、出力の形式は同様ですので、同じように利用することができます。たとえば、cmr10 のフォントテーブルを得る場合、次のようにしてください。

5) T_EX では、数値の表現に 8 進数、10 進数、16 進数を使用することができますが、数値の前に「r」をつけた場合には 8 進数を、「n」をつけた場合には 16 進数を表します。

6) ほかに、`%TEXHOME%\macros\fontbl.tex` も用意されています。


```

A> TeXenv.bat
A> TeX.bat testfont.tex
This is pTeX, C Version 2.99 j1.6 p1.0.9a fix2b (no format preloaded)
(c:/TeX/macros/testfont.tex
Name of the font to test = cmr10
Now type a test command (\help for help):)
*\table

*\bye
[1]
Output written on testfont.dvi (1 page, 5816 bytes).
Transcript written on testfont.log.
A> FMset.bat
:
:
A> print testfont.dvi
:
:
```

このようにして得られたフォントテーブルと、Table 4-16 に示す $\text{T}_{\text{E}}\text{X}$ の内部コードは対応しています⁷⁾ から、たとえば、記号 “j” を出力する場合には、以下に示すようないくつかの方法が考えられます。

- コントロール・シーケンス⁸⁾ !‘ を使用する。
- cmr フォントにおけるコード '074 をもとに、コントロール・シーケンス $\backslash\text{symbol}\{ '74\}$ あるいは $\backslash\text{char}'74$ を利用して新たなコントロール・シーケンスに定義する (先の “\” の出力に使用した方法)。
- cmr フォントにおけるコード '074 をもとに、コントロール・シーケンス $\backslash\text{symbol}\{ '74\}$ あるいは $\backslash\text{char}'74$ を使用して直接出力する。
- cmr フォントにおけるコード '074 に対応する $\text{T}_{\text{E}}\text{X}$ の内部コード “<” を記述する。

ただし、英数文字 (0…9, A…Z, a…z) の ASCII コードと等しい $\text{T}_{\text{E}}\text{X}$ の内部コードを持つ文字を出力する場合でないかぎり、最後の方法は $\text{T}_{\text{E}}\text{X}$ (「 $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ 」ではない) の初級者には必ずしもお勧めできません。その理由についてはここで触れませんが、興味のある人は D. E. Knuth 博士の『 $\text{T}_{\text{E}}\text{X}$ book』を参照してください⁹⁾。

7) 正確には、「対応するようにデバイスドライバが作成されている」ということです。 $\text{T}_{\text{E}}\text{X}$ のソースに含まれているすべての文字情報は、 $\text{T}_{\text{E}}\text{X}$ が処理する際に $\text{T}_{\text{E}}\text{X}$ の内部コードへと展開されます。出力する場合には、デバイスドライバがこの内部コードとフォントコードを対応させて出力するのです。このような処理のしかたゆえに、 $\text{T}_{\text{E}}\text{X}$ は異なる機種のコピュータの間でもデータの互換性が保証されているのです。

8) 「コントロール・シーケンスは \ から始まる」という定義からすると、この記述は正確さを欠きますが、ほかの表現を思いつかないので、コントロール・シーケンスということにしています。

9) 逆にいえば、『 $\text{T}_{\text{E}}\text{X}$ book』の内容についていける人ならば、最後の方法を利用しても問題を生じさせるようなことはないでしょう。

Table 4 - 16 • T_EX の内部処理コード

		8 進数 / 16 進数 (上段) 指定下位コード									
		'0	'1	'2	'3	'4	'5	'6	'7		
8 進 数 指 定 上 位 コ ー ド	'00x	^^@	^^A	^^B	^^C	^^D	^^E	^^F	^^G	"0x	16 進 数 指 定 上 位 コ ー ド
	'01x	^^H	^^I	^^J	^^K	^^L	^^M	^^N	^^O		
	'02x	^^P	^^Q	^^R	^^S	^^T	^^U	^^V	^^W	"1x	
	'03x	^^X	^^Y	^^Z	^^[^^\	^^]	^^^	^^_		
	'04x		!	"	#	\$	%	&	'	"2x	
	'05x	()	*	+	,	-	.	/		
	'06x	0	1	2	3	4	5	6	7	"3x	
	'07x	8	9	:	;	<	=	>	?		
	'10x	@	A	B	C	D	E	F	G	"4x	
	'11x	H	I	J	K	L	M	N	O		
	'12x	P	Q	R	S	T	U	V	W	"5x	
	'13x	X	Y	Z	[\]	^	_		
	'14x	`	a	b	c	d	e	f	g	"6x	
	'15x	h	i	j	k	l	m	n	o		
	'16x	p	q	r	s	t	u	v	w	"7x	
	'17x	x	y	z	{		}	~	^^?		
		"8	"9	"A	"B	"C	"D	"E	"F		
16 進数指定下位コード (下段)											

なお、先に紹介した `testfont.tex` では、フォントテーブル以外の出力形式も可能で、その出力形式は次に示すように調べることができます。これらの出力形式で出力する場合には、先にフォントテーブルを得た例で `\table` と入力したかわりに、それぞれの出力形式に対応した指示を入力してやればよいのです。いくつかのフォントで、いくつかの出力形式のサンプル出力を試してみてください。

```

A> TeXenv.bat
A> TeX.bat testfont.tex
This is pTeX, C Version 2.99 j1.6 p1.0.9a fix2b (no format preloaded)
(c:/TeX/macros/testfont.tex
Name of the font to test = cmr10
Now type a test command (\help for help):)
*\help

\init switches to another font;
\end or \bye finishes the run;
\table prints the font layout in tabular format;
\text prints a sample text, assuming TeX text font conventions;
\sample combines \table and \text;
\mixture mixes a background character with a series of others;
\alternation interleaves a background character with a series;
\alphabet prints all lowercase letters within a given background;
\ALPHABET prints all uppercase letters within a given background;
\series prints a series of letters within a given background;
\lowsers prints a comprehensive test of lowercase;
\uppers prints a comprehensive test of uppercase;
\digits prints a comprehensive test of numerals;
\math prints a comprehensive test of TeX math italic;
\names prints a text that mixes upper and lower case;
\punct prints a punctuation test;
\bigtest combines many of the above routines;
\help repeats this message;
and you can use ordinary TeX commands (e.g., to \input a file).
*
```

4.5.4 その他の視覚構造を制御する

前項までは、文字にかぎって L^AT_EX 上で視覚構造の制御を行う方法について述べてきました。本項では、特に文字にこだわらずに視覚構造の制御について解説することにしましょう。

◆ 改行と改ページ

L^AT_EX では、基本的に改行処理や改ページ処理を自動的に行ってくれますから、必ずしも使用する側が設定しなければならないというわけではありませんが、いくつかの理由によってそれらの処理を明示的に行いたい場合があります。

たとえば、「警告 (Warning)」(p.84) で解説した `Overfull \hbox` のようなメッセージが出力される場合があります。この場合、L^AT_EX が適当な改行位置を

見つけれずに出力が右マージン領域にはみ出してしまっていますから、出力の状態によってはなんらかの手段をもってこれを解消しなければなりません。

L^AT_EX には、このような場合に使用する改行の制御を行うコントロール・シーケンスがいくつか用意されていますので、以下に挙げてみることにします。

○ 改行を制御するコントロール・シーケンス

● `\linebreak[level]`

このコントロール・シーケンスが記述された部分では、*level* (0~4) の大きさに応じて改行がされやすくなり、*level* が 4 のとき必ず改行されます。

ただし、*[level]* の指定は省略可能で、省略した場合は `\linebreak[4]` と同等の強制力を持ちます。

なお、改行された行は右マージンに届くように単語間の空白が調節され、行の長さが隣接する他の行に揃えられます。

● `\nolinebreak[level]`

このコントロール・シーケンスが記述された部分では、*level* (0~4) の大きさに応じて改行が抑制され、*level* が 4 のとき最も改行されにくくなります。

ただし、*[level]* の指定は省略可能で、省略した場合は `\nolinebreak[4]` と同等の抑制力を持ちます。

なお、`\nolinebreak[0]` のときは、改行してもしなくてもよいということになり、これは `\linebreak[0]` と同値になります。

● `\newline`

行を終了して改行を行います。改行した行頭でインデントは行われません。

単純な改行処理を行うのみですから、行長を前後の行に揃えるなどの処理は行いません。

● `\\`

行を終了して改行を行います。改行した行頭でインデントは行われません。

単純な改行処理を行いますから、行長を前後の行に揃えるなどの処理は行いません。

`\newline` よりも `\\` のほうがよく利用されていますが、機能的な違いはありません。

以上、解説した改行を制御するコントロール・シーケンスのうちのいくつかについて、出力サンプルとそのソースを示してみることにします。

List 4-18 ● 改行を制御するコントロール・シーケンスのサンプルソース

- ```

1: \verb+\linebreak+の場合、改行された行は\linebreak
2: このように右マージンに届くように単語間の空白が調節され、
3: 行の長さが隣接する他の行に揃えられます。
4:
5: これに対して\verb+\newline+および\verb+\\+の場合は、\\
6: このように単純な改行処理のみを行いますから、\newline

```

7: 行長を前後の行に揃えたりはしません。

List 4-18 の出力を以下に示します。

`\linebreak` の場合、改行された行はこのように右マージンに届くように単語間の空白が調節され、行の長さが隣接する他の行に揃えられます。

これに対して `\newline` および `\\` の場合は、このように単純な改行処理のみを行いますから、行長を前後の行に揃えたりはしません。

さて、これまで説明してきたことは横方向、つまり行に対する制御でした。これに対して `Overfull \vbox` のようなメッセージが出力される場合は、縦方向、つまり、ページに対する制御を行ったほうがよいということがあります。L<sup>A</sup>T<sub>E</sub>X には、このような場合に使用することができる改ページ制御を行うコントロール・シーケンスもいくつか用意されていますので、以下に挙げてみることにしましょう。

#### ○ 改ページを制御するコントロール・シーケンス

- `\pagebreak[level]`

改行制御のコントロール・シーケンス `\linebreak` に相当する、改ページ制御のコントロール・シーケンスです。このコントロール・シーケンスが記述された部分では、*level* (0~4) の大きさに応じて改ページがされやすくなり、*level* が 4 のときは必ず改ページされます。

ただし、*[level]* の指定は省略可能で、省略した場合は `\pagebreak[4]` と同等の強制力を持ちます。

なお、改ページされたページは、下マージンに届くように行間の空白が調節されるので、出力はページ全体を占めます。

複数の空白ページを作成したい場合、このコントロール・シーケンスを単純に列挙しても、空白ページを作成することはできません。このようなときには、このコントロール・シーケンスを列挙する際に、全角の空白文字や `\mbox{ }{}` によって見えない出力を作成するか、あるいは何もしないコントロール・シーケンス `\null` をはさんでやることで望む結果が得られます。

また、スタイルオプション `jtwocolumn` を設定している場合には、改ページではなく、改段が行われます。

- `\nopagebreak`

改行制御のコントロール・シーケンス `\nolinebreak` に相当する、改ページ制御のコントロール・シーケンスです。このコントロール・シーケンスが記述された部分では、*level* (0~4) の大きさに応じて改ページが抑制され、*level* が 4 のとき、最も改ページされにくくなります。

ただし、`[level]` の指定は省略可能で、省略した場合は `\nopagebreak[4]` と同等の強制力を持ちます。

なお、`\nopagebreak[0]` のときは、改ページしてもしなくてもよいということになり、この指定は `\pagebreak[0]` と同値になります。

- `\samepage`

すべての改ページを抑制するコントロール・シーケンスです。

これと上記 2 つのコントロール・シーケンスとを併用することで、ほとんどの場合の改ページ動作を制御できるでしょう。

- `\newpage`

改行制御のコントロール・シーケンス `\newline` に相当する、改ページ制御のコントロール・シーケンスです。単純な改ページを行うため、ページ下方に余白をつくります。

`\pagebreak` と同様に、複数の空白ページを作成したい場合、このコントロール・シーケンスを単純に列挙しても複数の空白ページを作成することはできませんが、`\pagebreak` の場合と同様の解決策で解消することができます。

また、スタイルオプション `jtwocolumn` を設定している場合には、改ページではなく、改段が行われます。

- `\clearpage`

改行制御のコントロール・シーケンス `\newline` に相当する、改ページ制御のコントロール・シーケンスです。

`\newpage` との違いは、コントロール・シーケンス `\clearpage` が現れた時点でまだ出力しないでためてある図表<sup>10)</sup>を、改ページのあとで出力し、その後さらに改ページする（出力されていない図表を本文とは異なった独立したページに出力する）ことにあります。

また、スタイルオプション `jtwocolumn` を設定している場合には、単なる改ページが行われます。

- `\cleardoublepage`

`jarticle` スタイルでオプション `twoside` を設定している場合や、`jbook` スタイルを設定している場合には、改ページを行ったあとの記述を右ページから始めたいことがあります。このような場合に `\cleardoublepage` を使用すれば、次の記述開始ページが右ページになるように、空白ページを挿入してくれます。

その他の点では `\clearpage` と同じです。

10) `table` 環境や `figure` 環境のように、実際に出力する位置を  $\text{\LaTeX}$  の設定にまかせている環境があります。このような環境を「フロート」といい、本文中の「まだ出力しないでためてある図表」とは、出力を待っているフロートのことをいいます。

#### ◆ 横方向の空白を制御する

$\text{\LaTeX}$  ではソース上の複数の連続する〈空白〉がひとつとみなされることについては、すでに何度か説明してきました。それでは、実際に横方向の任意の長さの



空白を出力させたい場合には、どのようにするのでしょうか。

これまでに得た知識によれば、以下に示すような方法が考えられるでしょう。

- (1) コントロール・シンボル `\_` を使用する。
- (2) 改行を抑制して単語間空白を得る `~` を使用する。
- (3) 全角の空白文字を使用する。

もちろん、これらの方法は誤りではありませんし、実際これらの手段で横方向の空白を制御できないわけではありませんが、 $\text{\LaTeX}$  にはほかにも横方向の空白を制御するための手段が用意されています。

## ○ 横方向の空白を制御するコントロール・シーケンス

- `\hspace{length}`

長さ *length* を持つ横方向の空白を出力するコントロール・シーケンスです。*length* が負の値をとる場合、バックスペースキーを押した場合のように後退させることができます (ただし、存在する文字を消去したりはせず、重ね書きされます)。また、`\hspace` は、空白からなる「単語」を出力するコントロール・シーケンスとして考えられているため、このコントロール・シーケンスの前後に〈空白〉があった場合、この〈空白〉も単語間スペースを構成しますから、注意しなければなりません。

ただし、`\hspace` が出力行の行頭および行末に配置された場合 (ソース上の行頭および行末ではありません)、行頭および行末では空白を作成しません。もし、行頭および行末に配置されても空白を作成したい場合には、`\hspace*` を使用してください。

- `\hfill`

`\hspace` の引数に、無限に広がりうる長さ `\fill` を指定した場合 (`\hspace{\fill}`)、これは可能なかぎり伸びる空白となります。この設定はテキストの位置合わせなどに多用されるので、`\hfill` というコントロール・シーケンスとして定義されているのです。

- `\hfil`

`\hspace` の引数に、`\fill` より伸びる強さが弱い `\fil` を指定した場合 (`\hspace{\fil}`) を定義したコントロール・シーケンスです。`\hfill` より伸び方が弱くなります。

- `\_`

「独特な働きをするコード」(p.99) で解説したとおり、単語間空白を作成するためのコントロール・シンボルです。

- `\quad`

活字組版で主に行末の空間部分を埋めるために使用される込めもの、「クワタ」に相当する空白をつくります。具体的には、そのときに使用されている全角文字の大きさと等しい空白<sup>11)</sup>をつくるためのコントロール・シーケンスで、たとえば「 」のような空白を作成します。

11) 欧文組版の場合、現在使用中のフォントにおける大文字の M の幅 (1em) を表すこともある。

- `\quad`

`\quad` によってつくられる空白の 2 倍の長さの空白を作成するコントロール・シーケンスで、たとえば「     」のような空白を作成します。

- `\,`

ごく微小な空白を作成するためのコントロール・シンボルです。

典型的な使用例としては、ダブルクォーテーションマーク「`“`」とシングルクォーテーションマーク「`‘`」を連続して出力する場合があります。

この説明をするためには、 $\text{\LaTeX}$  でダブルクォーテーションマーク「`”`」を出力する方法について解説しておかなければなりません。 $\text{\LaTeX}$  では、ダブルクォーテーションマーク「`”`」を出力するとき、ソース上ではダブルクォーテーションマークではなく、シングルクォーテーションマークを 2 つ記述しなければならないのです。具体的には、「`‘ ‘ ’ ’`」という記述で「`”`」を得ることができます。

ところが、ダブルクォーテーションマーク「`“`」とシングルクォーテーションマーク「`‘`」を連続して出力したい場合には、もしソース上で「`‘ ‘ ‘`」と記述してしまうと、「`““`」と出力されてしまい、どちらがダブルクォーテーションマークなのか区別ができません。そこで、この場合には、ダブルクォーテーション「`“`」とシングルクォーテーションマーク「`‘`」との間に微小な空白を作成することで問題を解消します。具体的には、「`‘ ‘ \, ‘`」とすることで「`““`」という出力を得ることができます。

以上、解説した横方向の空白を制御するコントロール・シーケンスのうちのいくつかについて、出力サンプルとそのソースを示してみることになります。

List 4-19 ● 横方向の空白を制御するコントロール・シーケンスのサンプルソース

```

1: \verb+\hspace+□の出力例は、以下に示すとおりです。
2: \begin{quote}
3: ○以下に続く、\hspace{6zw}この空白である。\\
4: ○以下に続く、□\hspace{6zw}この空白である。\\
5: ○以下に続く、□\hspace{6zw}□この空白である。\\
6: ○以下に続く、\hspace{-6zw}_____この負の「空白」である。
7: \end{quote}
8:
9: ただし、\verb+\hspace+□は出力行の行頭および行末に配置された場合、\\
10: \hspace{3zw}このように行頭および行末では空白を作成しません。
11: もし、行頭および行末に位置しても空白を作成し
12: たい場合には、\verb+\hspace*+□を使用してください
13: い。 \verb+\hspace*+□を使用した場合には、\\
14: \hspace*{3zw}このように行頭および行末でも空白を作成します。
15:
16: また、\verb+\hfill+□を使用した場合、\\
17: \begin{quote}
18: ○たとえば\hfill□このような\hfill□出力\\
19: ○たとえば\hfill\hfill□このような\hfill□出力\\
20: ○たとえば\hfill□このような\hfill\hfill□出力
21: \end{quote}
22: \noindent
23: を得ることができるのです。
24:
25: \verb+\hfil+□の場合には、\\
26: たとえば\hfill□このような\hfill□出力を\\
27: 得ることができますが、\verb+\hfill+□と併用した場合、その「伸びる

```

28: 強さ」の違いから\\  
 29: たとえば\hfillこのような\hfil出力に\\  
 30: なるのです。

List 4-19 の出力を以下に示します。

\hspace の出力例は、以下に示すとおりです。

- 以下に続く、                      この空白である。
- 以下に続く、                      この空白である。
- 以下に続く、                      この空白である。
- 以下に続く、この負の「空白」である。

ただし、\hspace は出力行の行頭および行末に配置された場合、  
 このように行頭および行末では空白を作成しません。もし、行頭および行末  
 に位置しても空白を作成したい場合には、\hspace\* を使用してください。

\hspace\* を使用した場合には、

このように行頭および行末でも空白を作成します。

また、\hfill を使用した場合、

- |       |       |    |
|-------|-------|----|
| ○たとえば | このような | 出力 |
| ○たとえば | このような | 出力 |
| ○たとえば | このような | 出力 |

を得ることができるのです。

\hfil の場合には、

たとえば                      このような                      出力を  
 得ることができますが、\hfill と併用した場合、その「伸びる-強さ」の違  
 いから

たとえば                      このような出力に  
 なるのです。

#### ◆ 縦方向の空白を制御する

複数の連続する<空白>がひとつとみなされたように、 $\text{\LaTeX}$  では複数の連続する<改行>や、複数の連続する改行制御のコントロール・シーケンスは無意味な改行として Warning の対象となり、無視した場合には結果的にひとつとして扱われてしまいます。

横方向の空白の制御がそうであったように、 $\text{\LaTeX}$  では縦方向の空白についても明示的に空白の高さを指定することによって制御されます。ここでは、この機能を有するコントロール・シーケンスのうち、代表的なものについて概説してみることしましょう。



## ○ 縦方向の空白を制御するコントロール・シーケンス

● `\vspace{length}`

長さ *length* を持つ縦方向の空白を出力するコントロール・シーケンスです。

`\hspace` の場合がそうであったように、`\vspace` が出力ページのページ頭およびページ末に配置された場合、`\vspace` は空白を作成しません。もし、ページ頭およびページ末に配置された場合も縦方向の空白を作成したいのであれば、`\vspace*` を使用してください。

● `\vfill`

`\vspace` の引数に、無限に広がりうる長さ `\fill` を指定した場合 (`\vspace{\fill}`)、これは可能な限り伸びる縦方向の空白となります。この設定はテキストの位置合わせなどに多用されるので、`\vfill` というコントロール・シーケンスとして定義されているのです。

● `\vfil`

`\vspace` の引数に、`\fill` より伸びる強さが弱い `\fil` を指定した場合 (`\vspace{\fil}`) を定義したコントロール・シーケンスです。これは `\vfill` より伸び方が弱い、可能な限り伸びる縦方向の空白となります。

## 4.6 ..... 「始まり」の終わりに

以上で、本章の説明はおしまいです。あなたにとっての  $\text{\LaTeX}$  はこれから始まります。そこで、最終節である本節では、役に立つかどうかはわかりませんが、初級者のためのはなむけとして、いくつかの指針をまとめておきます。

### 4.6.1 日常あるいは平穏な日々

最後に  $\text{\LaTeX}$  のソース作成を行う際に初級者が特によく犯すミスと、 $\text{\LaTeX}$  の処理の過程で出力されるエラーメッセージについて、簡単に触れておきましょう。これらは、 $\text{\LaTeX}$  の処理中に間違いが発見されたときに見直すべき事柄であると同時に、間違える前によく理解しておいてほしい事柄でもあります。

ただし、ここで説明している以外の理由によっても、ここで示したエラーが起こる場合があります。各エラーメッセージの本来の意味を知りたい場合には、 $\text{\TeX}$  あるいは  $\text{\LaTeX}$  関連の文献を参照してください。

- ファイルの指定において、パス区切りに / ではなく、“\” または “\” を使用した。

- ! I can't find file ‘...’.

$\text{\TeX}$  では、“\” または “\” がエスケープ文字であるために、パス区切りとして使用することはできませんから、“/” を使います。

- 特殊な機能を持つ文字 #, \$, &, %, \_, {, }, ~, ^, \ を通常の文字として使用した。

- ! You can't use ‘macro parameter character #’ in ... mode.
  - ! Extra }, or forgotten \$.
  - ! Misplaced alignment tab character &.
  - ! Missing \$ inserted.
  - ... in math mode.

「特殊な機能を持つ文字」で示した Table 4-1 (p.99) を参照して、当該文字を出力するためのコントロール・シーケンスに変更しましょう。

## ○ コントロール・シーケンスのタイプミス。

- ! `\begin{...}` ended by `\end{...}`.
- ! Environment ... undefined.
- ! Undefined control sequence.

L<sup>A</sup>T<sub>E</sub>X を使いはじめたころは、特にこのケースのミスをよく起こします。

防止策としては、第6章「T<sub>E</sub>Xfamily」(p.229)で紹介するような、L<sup>A</sup>T<sub>E</sub>X のソースタイプ用にカスタマイズされたエディタを使用して、L<sup>A</sup>T<sub>E</sub>X ソースを作成することが挙げられます。

実は、この L<sup>A</sup>T<sub>E</sub>X のソースタイプ用にカスタマイズされたエディタを使用するという解決方法には、この手のケアレスミスの防止以外にも、以下のよう

- (1) ショートカットキーによるコントロール・シーケンスの入力がサポートされているため、文章執筆と併行してコントロール・シーケンスを埋め込むことができ、ソース作成が効率的に行える。
- (2) ショートカットキーによるコントロール・シーケンスの埋め込みが無意識にできるようになれば、文章の執筆だけに集中することが可能になる。

## ○ { と } との対応がとれていない。

- ! Extra }, or forgotten \$.
- ! Missing { inserted.
- ! Missing } inserted.
- ! Paragraph ended before ... was complete.
- ! Runaway argument.
- ! TeX capacity exceeded, sorry [...].

これも、L<sup>A</sup>T<sub>E</sub>X のタイプ用にカスタマイズされたエディタを使用して、L<sup>A</sup>T<sub>E</sub>X ソースを作成することで、かなり解消できるでしょう。

## ○ \begin と \end との対応がとれていない。

- ! `\begin{...}` ended by `\end{...}`.
- ! `\end{...}` occurred inside a group at level ... .
- ! Extra }, or forgotten `\endgroup`.
- ! Extra `\endgroup`.

これも、L<sup>A</sup>T<sub>E</sub>X のタイプ用にカスタマイズされたエディタを使用して、L<sup>A</sup>T<sub>E</sub>X ソースを作成することで、かなり解消できるでしょう。

## ○ 数式モードの制御文字の対応 (\$ と \$ など) がとれていない。

- ! Bad math environment delimiter.
- ! Extra }, or forgotten \$.
- ... in math mode.

これも、L<sup>A</sup>T<sub>E</sub>X のタイプ用にカスタマイズされたエディタを使用して、L<sup>A</sup>T<sub>E</sub>X ソースを作成することで、かなり解消できるでしょう。



- 数式モードでしか使用できないコントロール・シーケンスを使用している。

- ! Missing \$ inserted.

L<sup>A</sup>T<sub>E</sub>X では、特殊記号の多くが数学記号として扱われていますから、これらの記号を使用する場合には数式モードへ入らなければなりません。

- コントロール・シーケンスの引数が不足している。
- 動く引数で fragile<sup>1)</sup> なコントロール・シーケンス<sup>2)</sup> を使用している。

「動く引数」とは、他の場所への移動をとまなうようなコントロール・シーケンスの引数のことをいいます。たとえば、`\section` など章節を制御するコントロール・シーケンスの引数は、目次や文書のヘッダなど、コントロール・シーケンスで記述した以外の場所でも利用されることがありますから、動く引数にあたります。

一方、「fragile なコントロール・シーケンス」とは、動く引数のなかで使用すると、うまく動かないコントロール・シーケンスのことで、たとえば `\begin~\end` および `\footnote` などのことです。

このような場合には、fragile なコントロール・シーケンスの直前に、コントロール・シーケンス `\protect` を付加して、fragile なコントロール・シーケンスを保護してやります。

一応説明はしておきましたが、この説明でわからなくても悩む必要はありません。詳しくは L<sup>A</sup>T<sub>E</sub>X 関連の文献を参照してください。

- 数記号文字を使用したコントロール・ワードを定義しようとしている。

第 4.3.1 項「コントロール・シーケンス」のコントロール・シーケンスについてのコラム (p. 98) で詳説したように、数記号文字はコントロール・ワードのデリミタ (区切り文字) として機能してしまいますから、数記号文字が混在しているコントロール・ワードの定義はできません<sup>3)</sup>。

- 意味のない改行指定がある。

- ! There's no line here to end.

縦方向の空白をつくろうとして、改行を制御するコントロール・シーケンス `\newline` あるいは `\\` をソース中で連続使用している場合などが該当します。

もし縦方向に空白をつくりたい場合には、「縦方向の空白を制御する」(p.155) で説明した `\vspace{length}` を利用することで、*length* の空白を出すことが可能です。

1) 本来の意味は「もろい」とか「壊れやすい」などです。

2) 対比されるのは、「robust (「丈夫な」「強い」) なコントロール・シーケンス」です。

3) 一部の文字については定義することも不可能ではありませんが、原則としては「できない」と思ってください。

## 4.6.2 L<sup>A</sup>T<sub>E</sub>Xnician への道

かなりの駆け足でしたが、L<sup>A</sup>T<sub>E</sub>X のソース作成から出力まで、最小限度の説明をしてきたつもりです。理解していただけたでしょうか。

正直なところ、十分な解説ができたとは思いませんが、特にソースの作成の部分において初級者にとってはよけいと思える事柄についても触れてきたのは、読

者の皆さんが本書以外の  $\text{T}_{\text{E}}\text{X}$  および  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  に関連した文献に対したときに、スムーズに理解できるように隙間を埋められたらという希望からでした。

本章の冒頭でも述べたとおり、本章だけで  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  を使いこなせるということにはなりません。しかし、皆さんが他の参考文献を参照し、 $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  を使えるようになる過程で、一度といわず本章のページをめくってくれたなら、それで本章の目的は十分に果たされたといえるでしょう。

ここから先は皆さん次第です。 $\text{T}_{\text{E}}\text{X}$  が、巨大なだけで役に立たないものに終わるのか、それとも究極の芸術品であることを実感するのか、皆さんの自分自身との闘いです。

---

## Customize

---

第2章「Install」で紹介した  $\text{T}_{\text{E}}\text{X}$  インストーラでは、ユーザの持っているフォント・メモリ・ディスク容量等を考えて、最低限のシステムとして  $\text{T}_{\text{E}}\text{X}$  がインストールされます。もし、あなたの X680x0 がたくさんのメモリを増設していたり、ディスクに十分な空き容量があるなら、より快適な利用環境を求めてカスタマイズすることができます。本章では、そのようなカスタマイズ例をいくつか挙げてみることにしましょう。

---



## 5.1 ..... はじめに

第 5.2 節「フォントマネージャの高度な利用」(p.163)では、フォントマネージャのより進んだ利用方法について説明します。たとえば、あなたがツァイトのフォントを利用して、メモリに十分な空きがある場合は、フォントを描く線の太さを細くしたり、使用されたフォントをキャッシュしたりといったチューンナップができます。

また、インストール時のプリンタリストにあなたの所有するプリンタのコンフィギュレーションファイルがない場合、これを自分でつくらなければなりません。第 5.3 節「`print.cfg` の作り方」(p.186)では、キヤノンのプリンタ BJ-10 を例にとり、コンフィギュレーションファイルの作り方を説明します。

さらに、第 5.4 節「METAFONT」(p.214)では、 $\text{\TeX}$  の美しく多彩な英数字および記号のフォントを作成するシステム METAFONT について、そして第 5.4.2 項「`makefont` によるフォント作成」(p.220)では、その METAFONT のユーザーインターフェースとなってフォントをつくるツール `makefont.x` の使い方について説明します。 $\text{\TeX}$  のデバイスドライバを使っていると、英語のフォントが足りない旨を表示されることがありますので、そのような場合にはこの節を活用してください。

## 5.2……………フォントマネージャの高度な利用

フォントマネージャは、起動時に指定するファイル<sup>1)</sup>、およびそのファイルによって読み込まれるファイル (以下、これらのファイルを総称して「コンフィギュレーションファイル」と呼びます) を変更することで、さまざまなチューンナップが可能となります。

たとえば、各種書体<sup>2)</sup>を追加したり、書体を変形して長体・平体・斜体<sup>3)</sup>を生成したり、使用されたフォントをキャッシュして全体の処理を高速化したり、フォントの代用をしたりといったことが可能です。

本節では、フォントマネージャのコンフィギュレーションファイルの設定例をいくつか紹介しながら、このようなカスタム化について説明してみることになります。

X680x0 ユーザに最も普及していると思われる (ROM フォント以外の) フォントである、ツァイトのアウトラインフォントに関しては、ダイエツト<sup>4)</sup>フィルタと組み合わせでよりきれいなフォントを生成したり、キャッシュフィルタと組み合わせで高速化したりすることができます。そこで、第 5.2.2 項「ツァイトフォントとキャッシュフィルタの組み合わせ」(p.165) では、ツァイトフォントとキャッシュフィルタとの組み合わせについて、また、第 5.2.3 項「ツァイトフォントとダイエツトフィルタの組み合わせ」(p.177) では、ツァイトフォントとダイエツトフィルタとの組み合わせについて説明します。そして本節の最後、第 5.2.4 項「FM TOWNS の丸文字フォントを使用する」(p.180) では、特殊フォントの追加を行う例として、富士通の FM TOWNS に付属している CD-ROM から丸文字フォントを持ってきて、これを使ってみます。

なお、これらのチューンナップを行う場合には、メモリに十分な空き容量があることを前提とします。メモリが足りないときにはチューンナップできない場合もありますので、注意してください。

1) myfonts.fm が標準です。

2) 明朝体・角ゴシック体・丸ゴシック体はもとより、毛筆体・教科書体、果ては丸文字に至るまで。

3) このような変形は、従来は写植機やポストスクリプトプリンタを使わなければできませんでした。

4) フォントを細くするフィルタです。

### 5.2.1 フォントマネージャのコンフィギュレーションファイル

さて、フォントマネージャのコンフィギュレーションファイルのチューンナップについて説明する前に、フォントマネージャとコンフィギュレーションファイルとの関係について、簡単にここで説明しておくことにします。

フォントマネージャのコンフィギュレーションファイルは、いわば、フォント

5) 先に第 3.2.2 項「フォントドライバ」(p.51)で、フォントマネージャとフォントドライバの関係が、OS およびシェルとそこで動くツールの関係にあたるといったことと対比してください。

マネージャのためのプログラムファイルです。あるいは「その関係は OS および シェル (Human68k および COMMAND.X) とシェルスクリプト (バッチファイル) の関係にあたる」<sup>5)</sup>といったほうがわかりやすいでしょうか？

ここで、フォントマネージャのコンフィギュレーションファイルについて解説するために、少しだけ遠回りをします。

人間同士のコミュニケーションに日本語や英語といった言語が必要であるのと同じように、コンピュータと人間のコミュニケーションを仲介する“プログラム”を記述する際にも言語が必要です。コンピュータは、そのような言語によって記述されたプログラムを、基本的には記述された順番どおりに解釈し、命令を実行していきます。しかし、ときには状況判断にもとづいて処理の順番を変える必要が生じる場合もあります。さらに、同じ処理内容を繰り返し連続して実行する場合もありますし、似たような処理内容がプログラム全体を通していくつか存在する場合もあるでしょう。

そこで、普通、コンピュータ処理言語といわれるものは、逐次処理・条件判断・繰り返し制御・関数呼び出しといった機構を備えています。

さて、話題をフォントマネージャに戻しましょう。フォントマネージャを、このようなコンピュータ処理言語のひとつとしてとらえるとき、上述の 4 つの機構が備わっているかどうかということは、非常に大事なことです。

言語としてのフォントマネージャは、逐次処理・条件判断の機構は備えていますし、関数呼び出しのかわりにマクロ機構を備えてもいます。ただし、繰り返し制御はサポートしていません。これは、フォントの登録という目的から考えると、繰り返しを指示することはあまりないと考えられたからです。ですから、どうしても繰り返しの処理が必要なら、何行も同じ文を書かなければなりません。

さらに、フォントマネージャのプログラミング言語としての特性、“条件判断”について詳しく見てみます。

先ほどたとえに使用した「シェルスクリプト (バッチファイル)」では、条件判断の材料としては、環境変数と、ユーザからの入力とを利用します。環境変数の内容から作業ディレクトリ名を得たり、メニューでのユーザ入力によって動作を変えたりします。それと同様に、フォントマネージャは環境変数を参照することができますので、ひとつのコンフィギュレーションファイルでさまざまなユーザの、さまざまな環境に対応することができます。たとえば、フォントファイルを収めるディレクトリは、ユーザによってそれぞれ違うはずですが、フォントマネージャは、環境変数を参照することによってその違いを吸収します。

とはいえ、フォントファイル名などを環境変数にセットすると、あまりに環境変数が多くなってしまいます。そのため、本書添付のインストールキットでは、フォントファイル名などをインストールの際にユーザに入力してもらい、フォントマネージャのコンフィギュレーションファイル内に記述してしまうことにしました。そして、必要なら、ユーザが自分の環境にあわせてコンフィギュレーションファイルを書き換えることによって、自分なりにチューンナップができるようにしたのです。

このように見てくると、フォントマネージャも一人前のコンピュータ処理言語



といってよく、そのコンフィギュレーションファイルは、まさにプログラムそのもののものだといえます。

以下の項では、あなたがコンフィギュレーションファイルの「コンピュータ処理言語としての構造 (はっきりいってしまえば文法)」を知らなくても、自分の環境にあわせてチューンナップできるように説明してあります。

とはいうものの、コンフィギュレーションファイルの「コンピュータ処理言語としての構造」を知っていれば、より自由に、フォントマネージャがつくりだす環境をチューンナップすることができるでしょう。そこで、『Vol.2 — Reference 編』には、コンフィギュレーションファイル用に独立した節<sup>6)</sup>を設け、詳しい構文説明をしておきました。

また、 $\text{\TeX}$  システムとフォントマネージャシステムとの間でフォント名を橋渡しする p3m ファイルについては、同じく『Vol.2 — Reference 編』の第 3.2 節「p3m ファイル」(p.129)に説明してあります。コンフィギュレーションファイルを駆使して自分専用のフォントを作成したい方は、これらを参照し、勉強してみてください。

それでは、次項からは、本節の冒頭で述べたように、コンフィギュレーションファイルのチューンナップについて説明していくことにしましょう。

## 5.2.2 ツァイトフォントとキャッシュフィルタの組み合わせ

ツァイトの提供するアウトラインフォントは、フォントエッジ (輪郭) の描画、中塗り<sup>7)</sup>といった、全体的に CPU にとって重い処理を必要とします。

本項では、ある程度メモリの余裕がある人を対象に、キャッシュフィルタのより高度な利用によって、ページ描画速度の高速化を図る例を紹介します。

以下の変更を行うと、場合によってはメモリが足りないと表示されるかもしれませんが、その場合はいろいろ工夫してメモリを捻出するか、思い切って変更をあきらめるか、いっそのこと、メモリを新たに買い足すかしたほうがいいでしょう。

### ◆ キャッシュとは

フォントマネージャのフォントジェネレータ<sup>8)</sup>は、ビットマップデータが必要になるたびにフォント一文字分の描画を繰り返してページデータを生成します。

ここで、フォントファイルについて、少し説明しましょう。

一般的なフォントファイルのフォーマットには、大きく分けて以下に示す 2 種類があります。

#### (1) ビットマップフォント

ビットマップ情報をファイルに収めてあるフォントデータです。ビットマップフォントの描画はファイルから読み出すだけの手間ですむために、データ

6) 『Vol.2 — Reference 編』の第 3.1 節「コンフィギュレーションファイル文法」(p.100)を参照してください。

7) アウトラインフォントという名前からわかるように、普通、フォントファイル内にはフォントエッジに関する情報しか入っていません。そのため、フォントの内側を塗る必要があるわけです。

8) フォントマネージャに登録されるドライバは、基礎となるフォントを生成するジェネレータと、ジェネレータが生成したフォントを加工するフィルタと、いくつかのフォントをミックスするミキサに分類されることは、すでに第 3.2.2 項「フォントドライバ」(p.51)で述べました。

9) 大日本印刷が<sup>9)</sup>開発し、著作権を持っているフォントをビットマップファイルにして、アスキー社が販売している『パーソナル日本語 TeX フォントライブラリ』があります。

の展開は短時間で済みます。半面、ファイルサイズが大きくなります。

ビットマップフォントの例としては、たとえば、JXL4 フォーマット<sup>9)</sup>のフォントのほか、本書の添付ディスクに収録されているフォントがあります。

## (2) アウトラインフォント

描画プロセスをファイルに収めてあるフォントデータです。アウトラインフォントの描画は、ファイルに収められている描画プロセスを実際に実行する必要があるために、データの展開には時間がかかります。半面、ファイルサイズが小さくて済みます。

アウトラインフォントの例としては、ツァイトのフォントが各種あります。

さて、(1) のビットマップフォントはともかく、(2) のアウトラインフォントは、一文字ごとに描画を繰り返していると、データの展開に時間がかかってしまっ、とても使用に堪えられるものではありません。

そこで、一度描画したビットマップデータをメモリにためておき、必要になったときにメモリから展開済みのデータを得るようにすれば、かなりの高速化が期待できるはずです。それを行うのが、「キャッシュフィルタ」です。

フォントマネージャでは、フォントフィルタとしてキャッシュフィルタを登録することにより、どのフォントに対しても、そのビットマップデータをキャッシュすることができます。

では、とにかくキャッシュフィルタを登録すれば処理が速くなるのかというと、そうではありません。

たとえば、先に述べたように、ビットマップフォントはほとんど描画らしいことはしませんので、これに対してキャッシュフィルタを使用してもほとんど効果は期待できず、メモリの無駄遣いになってしまいます。一方、アウトラインフォントは一度描画したフォントをメモリに蓄えておくことにより、すでに描画したことのある文字フォントについてはメモリ上から高速に得ることができますので、キャッシュフィルタを使うと威力を発揮してくれるのです。

このように、キャッシュすべきフォントかどうかは、フォント生成処理が重いかどうかで決まってきます。

キャッシュの対象にするかどうかを決めるときに考慮しなければならないことは、これだけではありません。

ユーザによっては、「明朝体はよく使うが、ゴシック体はあまり使わない」といった使用頻度の違いもあることでしょう。使用頻度の低いフォントをキャッシュの対象にすると、メモリの無駄遣いになることがあります。

要するに、使用頻度が高いフォントかどうかによっても、キャッシュすべきかどうかは左右されるのです。

使用頻度についてももう少し考えてみると、“ある特定の文字”の使用頻度も重要なポイントになってくることがわかります。たとえば、ひらがなはよく使われますが、カタカナや漢字はそれに比べて使用頻度が低い<sup>10)</sup>ことを思い浮かべてもらえば理解はたやすいでしょう。したがって、メモリをうまく使おうと思うなら、ひらがなだけをキャッシュして、カタカナ・漢字はキャッシュしないという方法も有効だと考えられます。

10) 文字単位で考えてください。たとえば、ひらがなの“の”はよく使われます。それに比べて、漢字の“字”が使われる頻度は低いでしょう。

このようにひとつのフォントセットのなかでも、ひらがなか、そうでないかによって、キャッシュすべきかどうかが決まってきます。

最後は、フォントサイズについてです。明朝体を例にとってみると、本文のサイズ<sup>11)</sup>は非常によく使われますが、タイトルのサイズ<sup>12)</sup>はほとんど使われません。サイズによって、キャッシュすべきかどうかとも決まってくるのです。

そこで、フォントマネージャのキャッシュは、

11) 10 ポイントと呼ばれる大きさです。

12) たとえば 14 ポイントです。

- (1) 適切なフォントを選んで登録する
- (2) ひらがなにかぎってキャッシュする
- (3) フォントサイズを選んで登録する

という選択ができるようになっています。

本書添付のコンフィギュレーションファイルでは、キャッシュフィルタの対象となるフォントは、次の 3 つに絞ってあります。

- Z's STAFF または Z's WORD JG のアウトラインフォント
- 書体倶楽部のアウトラインフォント
- Z's WORD JG Ver.3 のベジェフォント

他のフォントはキャッシュフィルタを使わなくても十分高速にデータを展開できますから、キャッシュする対象には含まれません。本書の添付ディスクに付属するインストーラが作成するコンフィギュレーションファイルでは、つねにひらがなだけをキャッシュの対象とし、サイズについては自分で選べる<sup>13)</sup>ようにしてあります。

13) 標準では 10 ポイントを対象としています。しかし、プリンタの解像度が違えば、同じ 10 ポイントでもフォントサイズは異なりますし、ユーザのなかには本文は 12 ポイントを使うという人もいます。したがって、フォントサイズについては自由に換えられるようになっています。

#### ◆ キャッシュフィルタを使用する

キャッシュフィルタを使用する場合は、まず、TeX の HOME ディレクトリ内のファイル `myfonts.fm` をあなたの使い慣れたエディタで編集します。

あとで再修正ができるように、念のため、`myfonts.fm` のバックアップをとっておくとよいでしょう。ここでは、バックアップファイルを `myfonts.org` とします。

```
A> copy myfonts.fm myfonts.org
```

次に、エディタを起動してください。ここでは、エディタとして `ED.X` を使ってみます。

```
A> ed myfonts.fm
```



### ① フォント倉庫キャッシュフィルタ

本書の添付ディスクが構築する  $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  システムでは、キャッシュフィルタとして `fcache.sys` を採用していますが、添付ディスク 8 には、フォントマネージャによるフォント生成速度の向上を目的として、Seirios (許先明) 氏と Chaola (林曜三) 氏によって作成されたフォントキャッシュフィルタ `storage.sys` が添付されています。

この `storage.sys` は、一度作成したフォントをハードディスク等の「倉庫」に貯蔵しておき、必要なときに利用することによって、一種のフォントキャッシュのような働きをするものです。

このとき、メインメモリを「倉庫」の一部または全部として使用すれば、ハードディスクへのアクセスを減らすことも可能で、

この場合、`fcache.sys` の機能を一部内包することになります。

第4章「Exercise」の著者が試してみたところ、フォントデータの展開とフォントフィルタによる処理に時間のかかるベクトルフォントに対して使用すると、実感速度にして数割程度速くなりました。標準のフォントマネージャのコンフィギュレーションファイルである `myfonts.fm` に組み込むもよし、第5.2節「フォントマネージャの高度な利用」(p.163)で紹介するコンフィギュレーションファイル `fontman.fm` に組み込むもよしというわけで、ハードディスクとメモリの容量に余裕のある人は使用してみるとよいでしょう。

- 14) 利用フォントによって、行番号は違います。ご自分でお探してください。

`myfonts.fm` (List 5-1) 内の 129 行目<sup>14)</sup> の `define` の行頭にある “#” を消して、List 5-2 のように変更します。

List 5-1 ● `myfonts.fm` の一部

```
126: #
127: # 明朝体をキャッシュするなら次の # を外します。
128: #
129: #define minchoCache = true
```

List 5-2 ● キャッシュを有効化する

```
126: #
127: # 明朝体をキャッシュするなら次の # を外します。
128: #
129: define minchoCache = true
```

この行は、明朝体をキャッシュする場合に変更します。なお、ここでいう明朝体とは、プリントアウトするときのフォントのことであって、プレビューするときのフォントとは違うことに注意してください。

プレビューするときのフォントは、高速性を要求される一方で、あまり美しさを要求されないという事情もあって、通常は X680x0 本体に内蔵されている ROM フォントを拡大したり縮小したりして使っています。もちろん、プレビューするときのフォントにアウトラインフォントを指定することも可能です。しかし、そのような要求があるとは思えない<sup>15)</sup>ので、本書では触れません。

プレビューは十分高速になるようにつくられているので、プレビュー用のフォントにキャッシュを指定しても速くはなりません。これでも遅いと感じる人は、X680x0 自体の速度をチューンナップするしかないでしょう。

- 15) やはり、プレビューするときにアウトラインフォントを指定すると、描画速度は格段に落ちます。しかし、フォントマネージャの開発時には、案外役に立ちました。

さて、本題に戻ります。myfonts.fm の List 5-1 の下のほうにも同様な行があつて、ゴシック体をキャッシュの対象にするという指定ができます。

しかし、すでに述べたように、メモリが空いているからといって、ゴシック体もキャッシュするというのは考えものです。通常の TeX の文章においてゴシック体は、タイトルや特に強調したい部分くらいにしか使わないからです。つまり、ゴシック体をキャッシュしてもキャッシュのヒット率 (再利用率) はあまりよくないというわけです。

しかし、意図的にゴシック体を多用したソースを書いた場合は別です。この場合はメモリの空き容量と相談して、どんどんキャッシュしましょう。

さて、myfonts.fm を編集し、プリントアウト時の明朝体をキャッシュすることができたら、myfonts.fm をセーブし、エディタを終了します。

次に、コマンドラインから

```
A> fontman -r
```

と実行し、フォントマネージャが常駐していないことを確かめてください。

```
A> fontman -r
X68k Font Manager Ver 3.00c Copyright (C) 1989,90,91,92
by E x t (T.Kawamoto)
フォントマネージャは登録されていません。
```

と、表示されれば大丈夫です。

もし、フォントマネージャが常駐していたら、このコマンドでフォントマネージャの常駐は (ldepth だけ) 解除されます<sup>16)</sup>。フォントマネージャの ldepth の常駐が解除されるときは、

```
A> fontman -r
X68k Font Manager Ver 3.00c Copyright (C) 1989,90,91,92
by E x t (T.Kawamoto)
フォントマネージャは解除されました。
```

のような表示が出ます。こういう場合は、「fontman -r」を何度も繰り返してください。そのうち、

```
A> fontman -r
X68k Font Manager Ver 3.00c Copyright (C) 1989,90,91,92
by E x t (T.Kawamoto)
フォントマネージャは登録されていません。
```

16) ここで述べていることについては、すでに第 3.2.3 項「フォントマネージャの常駐」(p.54)でも触れたとおりです。

と表示されるようになります。これで、いっさいのフォントマネージャの常駐が解除されました。

それでは、

```
A> fontman myfonts.fm
```

でフォントマネージャを常駐させてください。ここで、プリンタドライバを使用して、印刷速度がどの程度高速化したか、計ってみるとよいでしょう。

なお、プレビューアには、このキャッシュの設定が効きませんので、注意してください。前述のとおり、すでに最高速のフォントを割り当てているので、キャッシュする必要がないからです。

#### ◆ キャッシュバッファを増やす

キャッシュを利用するように変更したら、さらに細かい設定を行います。最も手軽に変更できる設定項目は、キャッシュバッファのサイズです。そこで、次はキャッシュバッファのサイズを増やしてみましょう。

再びエディタを起動して `myfonts.fm` を編集してください。

```
A> ed myfonts.fm
```

List 5-3 ● `myfonts.fm` の一部

```
136: #
137: # キャッシュバッファサイズ (Kバイト単位)
138: #
139: define cacheBufSize = 500
```

List 5-3に示した行 (139 行目) を探し出し、数値 (500) をもっと大きな値に変更してみてください。たとえば、1000 にしてみましょう。

前節と同じく、フォントマネージャを解除し、常駐しなおします。

バッファサイズを倍にしたからといって、速度が倍になるわけではありません。また、バッファサイズを増やせば、それだけメモリの空き容量が少なくなります。効果がわかりにくい部分ですので、ご自分でプリンタドライバを起動して印刷速度を比較してみて、メモリの圧迫度と相談しながら適切なバッファ容量を決定してください。

#### ◆ キャッシュフィルタの、より進んだ利用

`myfonts.fm` のさらに下に、“`define cacheFontSizeList= 22`” という記述



があります。

List 5-4 ● myfonts.fm の一部

```

141: #
142: # キャッシュフォントサイズのリスト
143: # totex.sys の作成する環境ファイルを dump2tty.x を使って覗き、
144: # 良く使用されるフォントサイズを指定します。
145: # これは dpi によって大きく変化します。
146: #
147: # 360dpi 用フォントサイズ
148: #
149: #define cacheFontSizeList= 45 46 65
150: #
151: #
152: # 180dpi 用フォントサイズ
153: #
154: define cacheFontSizeList= 22

```

あなたの myfonts.fm に設定されている内容は、上の例のような表示内容とは異なっているかもしれません。上の例の場合、180dpi プリンタ用として 22 ドットのフォントだけをキャッシュする設定になっています。180dpi では、10 ポイントのフォントのドット数が 22 ドットですから、この指定が 10 ポイントのフォントをキャッシュの対象にするものであることがわかるでしょう。

これを、「12 ポイントもキャッシュしたい (10 ポイントのほか、\large も非常によく使う場合)」という用途に変更してみましょう。

はじめに、12 ポイントのフォントを使う  $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  のソースをつくり、これをプリンタドライバにかけてみます。その後、「フォントの使用履歴表示」ともいえるコマンド dump2tty.x で、何ドットのフォントが実際に使われたのかを確かめ、そのサイズを myfonts.fm に書き込みます。

以下、実際にどうすればいいかを示してみましょう。

- (1) 適当な作業ディレクトリ (たとえば A:\WORK) に移動します。

```
A> cd A:\WORK
```

- (2) 12 ポイントのフォントを使用する  $\mathrm{IAT}_{\mathrm{E}}\mathrm{X}$  のサンプルソースをつくり、ED.X を起動して、簡単なサンプル sample12pt.tex を作業ディレクトリ A:\WORK 内につくり、

List 5-5 ● サンプルソース — sample12pt.tex

```

1: \documentstyle{jreport}
2: \begin{document}
3: \large 12 ポイントキャッシュ試験用
4: \end{document}

```

- (3) 次に、sample12pt.tex を  $\mathrm{IAT}_{\mathrm{E}}\mathrm{X}$  で処理し、sample12pt.dvi を得ます。

```
A> jlatex sample12pt
```

### ◎ ポイント数とドット数

印刷の用語として、フォントの大きさを表すのに“ポイント数”というものがあります。これは mm と同じく、長さを表す単位です。1 ポイントは約 0.35 mm です。この単位を使って、フォントの大きさを表します。具体的には、英字“M”の横幅をポイント数で表したときの数値を、フォントの大きさとして使用します。たとえば 10 ポイントのフォントとは、そのフォントの“M”の幅が約 3.5 mm であるということです。

印刷では、通常書籍などの本文には 9 ポイント、論文などの本文には 10 ポイントのフォントを使用します。

これとは別に、寸法を表すとき、インチという単位が英語圏ではよく使われています。1 インチは 25.4 mm です。また、1 インチが 72.27 ポイントであることも知っておいたほうが、インチをポイントに換算する場合に（あるいは逆の場合にも）便利でしょう。

フォントマネージャでは、フォントのサイズを表すために「ドット」という単位を使います。フォントを実際にコンピュータ、あるいはプリンタ上で扱う場合には、文字は小さなドットの集まりとして表現されます。たとえば、X680x0 のディスプレイに表示される漢字フォントは 16 ドットの大きさです。

おおざっぱに言えば、ドット数が多ければ多いほど大きい文字だといえるのですが、正確にはそうともいえません。これは、表示デバイス（たとえば、ディスプレイであったり、プリンタであったり）によって文字の大きさが変わってくるのです。たとえば、プリンタを例にとると、よく 24 ドットプリンタだの、48 ドットプリンタだのといった、プリンタのきれいさを表す数値を見か

けるでしょう。これらは、正しくはプリンタの解像度を表します。解像度を表す単位としては dpi が使われます。

24 ドットプリンタは、この単位でいうと 180 dpi プリンタと呼ばれます。同様に 48 ドットプリンタは、360 dpi プリンタと呼ばれます。dpi は dots per inch の略で、そのプリンタが 1 インチを何ドットの点で表せるか（横 1 インチの長さに並べることのできる点の数）を示すものです。このため、縦横それぞれ 24 ドットからなるフォントを、180 dpi プリンタで印字した場合と 360 dpi プリンタで印字した場合とで比べてみると、出力されたフォントの大きさは、180 dpi プリンタで出力した場合のほうが大きくなります。

逆に、180 dpi プリンタで印字した 24 ドットのフォントと同じ大きさのフォントを 360 dpi プリンタで印字しようとする、と、48 ドットのフォントが必要です。

ポイント数とドット数との関係について、計算式を知っておくと役に立ちます。上述のように、ポイント数は実際に目で見えた大きさで確認できますが、ドット数というのは表示デバイスの解像度に左右されますから、表示デバイスの解像度を考慮しないと、ポイント数とドット数間の変換計算はできません。

表示デバイスの解像度が  $m$  dpi であったとすると、 $x$  ポイントのフォントのドット数は、 $x \times m / 72.27$  となります。

ただし、ポイント数とは、先ほども述べたように英語の文字“M”を基準にして表しますが、フォントマネージャではドット数というのは、漢字の高さを基準にして表すことになっているので、この計算式での計算結果がそのまま使えるわけではないことに注意してください。

- (4) `dump2tty.x` の表示をわかりやすくするための前処理として、次の一連の操作を行ってください。`dump2tty.x` がよけいな情報を出力しないように、`dump2tty.x` が参照するディレクトリ内をきれいに掃除します。

- (a) フォントマネージャの常駐を解除します。

```
A> fontman -r
```

これまでも述べたように、常駐を解除した旨が表示されるまで、何度も実行する必要があります。

- (b)  $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  の HOME ディレクトリ内にある `dump` サブディレクトリに移動します。

```
A> cd %TEXHOME%
A> cd dump
```

- (c) ディレクトリの中身をすべて消去します。

```
A> del *.*
```

- (d) フォントマネージャを起動します。

```
A> fontman myfonts.fm
```

- (5) 次に、実際にフォントを使用するために、プリンタドライバを使用します。元の作業ディレクトリ `A:\WORK` に戻って `sample12pt.dvi` をプリンタドライバにかけます。

```
A> cd A:\WORK
A> print sample12pt
```

- (6) もう一度フォントマネージャの常駐を解除します。

```
A> fontman -r
```

- (7) また、`dump` ディレクトリ内に移動します。

```
A> cd %TEXHOME%
A> cd dump
```

- (8) ここでディレクトリの中身を見てください。



```
A> dir
```

```

ボリュームがありません a:/usr/local/TeX/dump
 5 ファイル 4663K Byte 使用中 3480K Byte 使用可能
 ファイル使用量 6K Byte 使用
dumps.m2d 150 93-10-29 7:59:48
font.001 881 93-10-29 8:00:18
font.002 1377 93-10-29 8:00:20
font.003 879 93-10-29 8:00:20
font.004 873 93-10-29 8:00:20

```

dumps.m2d と、font.001 から font.004 までのファイルが見えます。

後者の 4 つのファイルは、myfonts.fm の設定にもとづいてフォントマネージャが使用した 4 つのフォントについての使用履歴です。このなかには、文字の幅の情報や、文字位置の横方向の修正情報が入っています。これはどういうことかということ、ツアイトのフォントを含め、日本語のフォントは多くの場合、いわゆる固定幅フォントとして作成されています。つまり、すべてのキャラクタの幅は揃っているわけですが、これに対して TeX システムではフォントの幅はキャラクタごとに異なるという前提になっています。したがって、固定幅のフォントを出力する場合には、フォントの幅や位置等の修正が必要なのです。フォントマネージャでは、文字が利用された際に、その文字に関する修正情報を計算しています。そして、フォントマネージャの常駐を解除したとき、dump ディレクトリ内にこの修正情報を収めるのです。一見したところでは、どれが目的のフォントなのかわかりません（実は、ファイルサイズが最も大きいものが目的のフォントなのですが）ので、dumps.m2d をのぞいてみましょう。

```

A> type dumps.m2d
zeit-ベジエ-ゴシック-tex[goth] font.001
zeit-ベジエ-明朝-tex[min] font.002
sharp-rom-平滑 (16)-tex[goth] font.003
sharp-rom-平滑-tex[min] font.004

```

dumps.m2d は、フォントマネージャによって使用された 4 つのフォントと、今、説明した 4 つのファイルについての対応関係を収めたファイルです。つまり、どのフォント情報をどのファイルに収めたかを記録してあります。皆さんのマシンのインストール状況によって、このファイルの中身は異なるので、キーワードだけ挙げておきますと、“zeit”と“明朝”がこれにあたります。ここで示した例の場合、“zeit”と“明朝”というキーワードが含まれたフォントが font.002 と対応していますから、これが最も利用されているフォントであることがわかります（最もサイズの大きなファイルで間違いありませんでしたか？）。

- (9) 目的のファイルを dump2tty.x にかけます。

```
A> dump2tty font.002
```

画面に次々と、たくさんの表示が現れたことでしょう。

```
fontman dump file displayer Copyright (C) 1991,92
by E x t (T.Kawamoto)
totex dump format Ver 1.10a(Ext)
font name = sharp-rom-平滑-tex[min]
dump file path = A:/usr/local/TeX/dump
jfm file name = A:/usr/local/TeX/fonts/min10.tfm
chars num = 121
fonts num = 32
design size = 00A00000
height = 000C7100
depth = 000238C0
italic = 00000000
```

#### char informations

```
dots per design 19.5932 29.8879
y dots(font size) 18 27
y offset 15 23
italic x offset 0 0
```

|   | code | width    | xdot | xoff | xdot | xoff |
|---|------|----------|------|------|------|------|
| ※ | 0000 | 000F653D | 19   | 0    | 29   | -1   |
| 、 | 2122 | 00081070 | 10   | **   | 15   | **   |
| 。 | 2123 | 00081070 | 10   | **   | 15   | **   |
| , | 2124 | 0005A89D | 7    | **   | 11   | **   |
| . | 2125 | 0005A89D | 7    | **   | 11   | **   |
| ・ | 2126 | 0005A89D | 7    | **   | 11   | **   |

:

:

|   |      |          |    |    |    |    |
|---|------|----------|----|----|----|----|
| ワ | 256F | 000F653D | 19 | ** | 29 | ** |
| ヲ | 2572 | 000F653D | 19 | ** | 29 | ** |
| カ | 2575 | 000BF57D | 15 | ** | 22 | ** |
| ケ | 2576 | 000BF57D | 15 | ** | 22 | ** |
|   | 2577 | 000BF57D | 15 | ** | 22 | ** |

- (10) その表示のなかから、

```
y dots(font size) 18 27
```

という行を探し出してください。この行に、プレビューアやプリンタドライバに使われたフォントのドット数が表示されています。

「y dots(font size)」の行に、2つの数字が見えます。この例では18と27ですが、18はプレビュー用のフォントのドット数を表します。18でないほうがあなたのプリンタのドット数です。皆さんのプリンタのdpi値によっては27という数字ではないことがあります（異なるからこそ、調べてもらっているわけです）。

いくつでしたか？ 覚えておいてください。

- (11)  $\text{\TeX}$  の HOME ディレクトリ内に戻って、myfonts.fm の編集を続けます。

```
A> cd %TEXHOME%
A> ed myfonts.fm
```

List 5-4 に戻ってください。前に見ていた行 “define cacheFontSizeList=22” に、先ほど調べておいた数値を追加します。ここでの例の場合、27でしたから、以下ようになります。

List 5-6 ● キャッシュフォントサイズを追加する (180dpi)

```
141: #
142: # キャッシュフォントサイズのリスト
143: # totex.sys の作成する環境ファイルを dump2tty.x を使って覗き、
144: # 良く使用されるフォントサイズを指定します。
145: # これは dpi によって大きく変化します。
146: #
147: # 360 dpi 用フォントサイズ
148: #
149: #define cacheFontSizeList= 45 46 65
150:
151: #
152: # 180 dpi 用フォントサイズ
153: #
154: define cacheFontSizeList= 22 27
```

360dpi プリンタの人は、実はすでに数値が書き込まれています。「65」がそうです。「360 dpi 用フォントサイズ」と書いてある行の2行下の行頭の“#”を外し、「180 dpi 用フォントサイズ」と書いてある行の2行下の行頭に“#”をつけてください。

List 5-7 ● キャッシュフォントサイズを追加する (360dpi)

```
141: #
142: # キャッシュフォントサイズのリスト
143: # totex.sys の作成する環境ファイルを dump2tty.x を使って覗き、
144: # 良く使用されるフォントサイズを指定します。
145: # これは dpi によって大きく変化します。
146: #
147: # 360 dpi 用フォントサイズ
148: #
149: define cacheFontSizeList= 45 46 65
150:
151: #
152: # 180 dpi 用フォントサイズ
153: #
154: #define cacheFontSizeList= 22 27
```



- (12) あなたの使っているプリンタの解像度が 180dpi でも 360dpi でもない場合は、自分のプリンタの解像度専用の行を新たにつくって、今、述べてきたような作業を行ってください。

以上で説明したことの応用として、「11 ポイントのフォントだけをキャッシュしたい (スタイルオプション 11pt<sup>17)</sup> をよく使う) 場合」などが考えられます。試してみてください。

17) 「*style\_option*」 (p.105) を参照してください。

### 5.2.3 ツァイトフォントとダイエツフィルタの組み合わせ

ツァイトの提供するアウトラインフォントは、太すぎると思われる方もいるのではないのでしょうか。

ビットマップフォントは、それぞれのサイズごとに適切な太さでフォント全体がデザインされているので、異なるサイズでもフォントの太さが気になることはまずありません。たとえば、12 ポイントのフォントは、10 ポイントのフォントに比べて単に 1.2 倍に拡大された形をしているわけではなく、12 ポイントで見たときに最も美しく見えるようにデザインされているのです。実際、印刷の組版では、それぞれのサイズごとに縦・横の太さを変えてデザインされたフォントが用意されていますし、これは METAFONT が作成する T<sub>E</sub>X の英数記号フォントについても同様なことがいえます。

しかし、アウトラインフォントは、すべてのサイズについて、同一スケールで太さが決められているので、当然、サイズによってフォントのバランスがとれていないことがあります。具体的にいえば、各フォントは比較的大きなサイズにあわせてデザインされているので、小さなサイズで使うとフォントが太く感じられます<sup>18)</sup>。

そこで、線の太さを調整して細くするフィルタ — ダイエツフィルタ — をツァイトのアウトラインフォントに組み合わせてみましょう。ダイエツフィルタは、アウトラインフォント描画の際に、そのフォントを描く線の太さを細く調整するものです。

そのアルゴリズムは、大まかにいうと次のとおりです。

- (1) 必要なフォントの 2 倍のサイズで、アウトラインフォントを描画する
- (2) 縦・横ともに 1 ドットおきにドットを拾い集める
- (3) 目的サイズのフォントとして返す

冒頭で述べたとおり、ビットマップフォントにダイエツフィルタを適用する意味はほとんどありません。また、ROM フォントは 24 ドットでデザインされているので、これも細くする必要はありませんし、プレビューアで利用するため、ダイエツフィルタをかませると速度低下につながるので、これは避けるべきでしょう。そういう意味で、ダイエツの対象とするフォントは、現状では、

○ Z's STAFF または Z's WORD JG のアウトラインフォント

18) 逆にいえば、小さなフォントにあわせてフォントをデザインすると、拡大して使ったときに線が細く見えて弱々しく感じられます。

- 書体倶楽部のアウトラインフォント
- Z's WORD JG Ver.3 のベジェフォント

の 3 つで十分でしょう。

ただし、ダイエットフィルタを使う場合はメモリをかなりたくさん必要としますので、メインメモリが 4M バイト以上ある人でないと、この組み合わせを使うことはできません。

#### ◆ ダイエットフィルタを使用する

ダイエットフィルタを使用する場合は、 $\text{T}_{\text{E}}\text{X}$  の HOME ディレクトリ内のファイル `myfonts.fm` を編集します。念のため、バックアップをとっておくとよいでしょう。

```
A> copy myfonts.fm myfonts.org
```

次に、エディタを起動してください。

```
A> ed myfonts.fm
```

#### List 5-8 ● `myfonts.fm` の一部

```
99: #
100: # diet.sys を使う場合、次の # を外します。更に、diet.sys で使用
101: # するオプションを指定してください。
102: #
103: #define useDiet = true
```

List 5-8 の行を見つけて、コメントに書いてあるように行頭の “#” を外してください。

#### List 5-9 ● ダイエットフィルタ `diet.sys` を有効化する

```
99: #
100: # diet.sys を使う場合、次の # を外します。更に、diet.sys で使用
101: # するオプションを指定してください。
102: #
103: define useDiet = true
```

そして、これをセーブし、エディタを終了します。

`fontman -r` で フォントマネージャが起動していないことを確かめます。

```
A> fontman -r
```

```
A> fontman -r
X68k Font Manager Ver 3.00c Copyright (C) 1989,90,91,92
by E x t (T.Kawamoto)
 フォントマネージャは登録されていません。
```

あるいは、

```
A> fontman -r
X68k Font Manager Ver 3.00c Copyright (C) 1989,90,91,92
by E x t (T.Kawamoto)
 | depth 分のフォントドライバが解除されました。
 フォントマネージャは解除されました。
```

と表示されれば OK です。

そうでなければ、何回か `fontman -r` を繰り返してみてください。

最後に、

```
A> fontman myfonts.fm
```

でフォントマネージャを常駐させます。あとは、普通に使用してください。

#### ◆ ダイエットフィルタのより進んだ利用法

`myfonts.fm` のなかに “define minDietOptions” という行があります。

List 5-10 ● `myfonts.fm` の一部

```
105: #
106: # 明朝体用 diet.sys のオプション
107: #
108: define minDietOptions = -p 7
109: #define minDietOptions = -p 8
110: #define minDietOptions = -d -e -k -h
111: #define minDietOptions = -p 1
112: #define minDietOptions = -p 2
```

T<sub>E</sub>X の HOME ディレクトリ内のマニュアル `fontman\doc\drivers\diet.doc` をよく読んで、どれかの行を有効にして<sup>19)</sup>みてください。

たとえば、“define minDietOptions = -p 1” という行を有効にすると、「高解像度明朝」用の設定となります。

List 5-11 ● ダイエットフィルタのオプションの設定

```
105: #
106: # 明朝体用 diet.sys のオプション
```

19)define から始まっている行の行頭に “#” を付け加えて、使いたいオプション指定の行頭の “#” を消します。



◎newfm\_fix.lzh その 1 ～フォントマネージャの複数登録について～

本文中、「フォントマネージャは複数回数登録できる」あるいは「いっさいのフォントマネージャの常駐が解除される」といった主旨の文章が何度か出てきていますが、このことについてもう少し具体的に説明しておきます。

たとえば、本書の添付ディスクに収録された  $\TeX$  システムには、fontman.fm (汎用フォント登録用)、vfdecoreate.fm ( $\text{\texttt{VF-LAT-EX}}$  の飾り文字登録用)、vftransform.fm ( $\text{\texttt{VF-LAT-EX}}$  の変形フォント登録用) という 3 つのコンフィギュレーションファイルが用意されています。ここで仮りに、コマンドラインから「fontman」  
 $\text{\texttt{C:\%TEXHOME%\fontman\fontman.fm}}$  と入力したとしましょう。このとき登録され、使用可能になるフォントは、fontman.fm が定める汎用フォントだけです。引き続いてコマンドラインから、「fontman

$\text{\texttt{C:\%TEXHOME%\fontman\vfdecoreate.fm}}$  と入力したとします。このとき、使用可能になるフォントは fontman.fm が定める汎用フォントと vfdecoreate.fm が設定する  $\text{\texttt{VF-LAT-EX}}$  (第 6 章「 $\text{\texttt{TeXfamily}}$ 」の p.234 参照) が対応する飾り文字になります。さらに vftransform.fm を指定してフォントマネージャを登録した場合、ここまで使用可能だったフォントに加え、vftransform.fm が設定する  $\text{\texttt{VF-LAT-EX}}$  の変形フォントも使用可能になるわけです。

つまり、フォントマネージャは、ある特定の用途 (あるいは種類) のフォントの定義を最も基本となるフォントの定義とは別のコンフィギュレーションファイルで行うことにより、基本フォントしか使用しない場合のメモリ消費を抑えることに成功しているのです。

(p.185 へ続く)

```
107: #
108: #define minDietOptions = -p 7
109: #define minDietOptions = -p 8
110: #define minDietOptions = -d -e -k -h
111: define minDietOptions = -p 1
112: #define minDietOptions = -p 2
```

あとは、「ダイエットフィルタを使用する」(p.178)と同じくフォントマネージャを操作して試してみてください。

## 5.2.4 FM TOWNS の丸文字フォントを使用する

フォントマネージャ応用の最後に、丸文字フォントを利用してみましょう。丸文字フォントは、 $\text{\texttt{TeX}}$  システムそのものではサポートされていません。

しかし、 $\text{\texttt{TeX}}$  システムには柔軟な拡張機能があるので、独自に拡張して丸文字フォントを  $\text{\texttt{TeX}}$  システムの機能の一部にすることが出来ます。

ここでは、丸文字フォントを扱えるように、本書で構築した  $\text{\texttt{TeX}}$  システムを拡張する例を紹介しましょう。

なお、丸文字フォントファイルとして、FM TOWNS の CD-ROM に添付されている MARU24.FNT というファイルを利用します。丸文字フォントを使う場合には、これを手に入れる必要があります。もし、手に入らない場合は、残念ながら、丸文字フォントの拡張はできません。あらかじめご了承ください。

標準の  $\text{\texttt{TeX}}$  システムには、丸文字フォントに切り替えるマクロすらありません

ので、説明は、フォントマネージャでの丸文字フォントの追加だけでなく、 $\text{T}_{\text{E}}\text{X}$  あるいは  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  で丸文字フォントを使用するために追加するコードについてもいっしょに説明します。

#### ◆ $\text{T}_{\text{E}}\text{X}$ および $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ でのユーザ追加コード

List 5-12 •  $\text{T}_{\text{E}}\text{X}$  および  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  でのユーザ追加コード

```
1: \font\maruten=maru10
```

$\text{T}_{\text{E}}\text{X}$  や  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  で丸文字フォントを使用するためには、たとえば、List 5-12 にあるコードを原稿の適切な箇所に書けばよいのです。このような定義を書く場所は、あなたが使っている処理系が  $\text{T}_{\text{E}}\text{X}$  なのか  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  なのかによって一般的に決まっています。

$\text{T}_{\text{E}}\text{X}$  を使っている場合は、丸文字フォントを使用する場所よりも前なら、どこに書いてもかまいません。普通、マクロ定義は原稿内の 1 ヶ所にまとめて書いておくものですから、そこに書き入れます。 $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  を使っている場合は、プリアンブル<sup>20)</sup>部に置きます。この 1 行は、丸文字フォントを使用することを、 $\text{T}_{\text{E}}\text{X}$  (あるいは  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ ) に知らせるための宣言です。maru10 というのが丸文字フォントファイル名で、\maruten が、「これから丸文字に切り替える」旨を  $\text{T}_{\text{E}}\text{X}$  あるいは  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  に知らせるためのコントロール・シーケンスです。以後、\maruten を使うと、10 ポイントの丸文字が使えるようになります。

20) プリアンブルについては「preamble」(p.109)で述べました。

#### ◆ フォントマネージャでの修正および登録

まず、 $\text{T}_{\text{E}}\text{X}$  の HOME ディレクトリ内にあるサブディレクトリ fontman 内のファイル fontman.fm を、 $\text{T}_{\text{E}}\text{X}$  の HOME ディレクトリ内に maru.fm という名前でコピーします。

```
A> copy fontman\fontman.fm maru.fm
```

このファイルは、myfonts.fm と同様に、フォントマネージャの数あるコンフィギュレーションファイルのうちのひとつです。myfonts.fm は、皆さんのフォントファイル環境にあわせて書かれた専用ファイルであるのに対し、fontman.fm は、各種フォントを混在できるように汎用的に書かれています。

専用ファイルというのは、できることがかなり制限されるかわりに、取り扱いが容易です。一方、汎用ファイルは、何でもできるかわりに、取り扱いが極端に難しくなります。

インストールの章、および前節までは専用ファイル `myfonts.fm` を使っていますが、丸文字を追加するためには、やはり、汎用コンフィギュレーションファイルとして `%TEXHOME%\fontman` 配下に用意されている `fontman.fm` に頼らざるを得ません。

さて、`maru.fm` というファイルにコピーが終わったので、エディタを起動してみましょう。

```
A:\TEX>ed maru.fm
```

`maru.fm` から List 5-13 に示す各行を探し出し、これを設定しなおしてください。変更するのは、行頭の“#”をつけるか外すかです。

List 5-13 ● `maru.fm` の変更箇所

```
1: 95 行目 (# をつける)
2: #define useBitmap = true
3:
4: 104 行目 (# をつける)
5: #define useZeit = true
6:
7: 112 行目 (# をつける)
8: #define useClub = true
9:
10: 120 行目 (# をつける)
11: #define useJg = true
12:
13: 128 行目 (# をつける)
14: #define useJXL4 = true
15:
16: 137 行目 (# を外す)
17: define useMaru = true
18:
19: 145 行目 (# を外す)
20: define useTeX = true
```

すでに上記のような設定になっている行もあるかと思いますが。その行はそのままにしておいて、他の行の設定を確認してってください。

`maru.fm` はもとは `fontman.fm` であったので、どんなフォントでも登録できるように書かれています。しかし、皆さんはすでに `myfonts.fm` という名前で、自分のシステムにあった専用コンフィギュレーションファイルを持っているはずです。それを使わない手はありません。

そこで、フォントマネージャの機能のひとつである

“コンフィギュレーションファイルは、32 個まで同時に利用（登録）できる”

ことを利用し、

- (1) `maru.fm` で丸文字フォントだけを登録する
- (2) `myfonts.fm` で基本フォントを登録する

ようにしましょう。

具体的には、



- (1) まず、丸文字フォントを登録した `maru.fm` を常駐させる。

```
A> fontman maru.fm
```

- (2) 次に、`myfonts.fm` を常駐させる。

```
A> fontman myfonts.fm
```

となります。

### ◆ サンプルソース

最後に、サンプルを挙げてみましょう。

- (1) まず、List 5-14 を `sample.tex` としてエディタで入力してください。

List 5-14 ● 丸文字サンプル — `sample.tex`

```
1: \documentstyle{jreport}
2: \font\maruten=maru10
3: \begin{document}
4: その女子高生はいいました。
5:
6: {\maruten 「べつにいい、それでもおお、いいけどおお。」}
7:
8: {\maruten 「けどおお、やっぱああ、めんどおじゃん。」}
9:
10: {\maruten 「それにいい、こまるひともお、いるとおもうしい。」}
11:
12: 嫌なら嫌ってはつきり言えばいいのに。
13: \end{document}
```

このサンプルでは、「その女子高生は…」の部分と「嫌なら嫌って…」の2カ所が通常の明朝体で、「女子高生」がしゃべった部分はすべて丸文字で書くように指定してあります。

- (2) 次に、「フォントマネージャでの修正および登録」(p.181) で説明したように、フォントマネージャを二重に登録してください。
- (3) `sample.tex` を  $\text{\LaTeX}$  で処理します。

```
A> latex sample
```

ところが、この処理はエラーを出して止まります。どんなエラーが出るかは自分で試して確かめてください。

原因は、`jfm` ファイルがないことが問題です。`jfm` ファイルとは、`tfm` ファ

## 21) フォントの大きさ・配置

などの情報が収められています。TeX はその処理の過程で実際の文字をページに配置しているわけではなくて、tfm ファイルに記述された情報を頼りに、空の箱を並べていくイメージで文字の配置を決めていきます。

## 22) 編集部注：「混乱する

な」というほうが無茶かもしれませんが、「慣れてください」ということにしましょう。

イル<sup>21)</sup>の日本語フォント版で、このファイルがないと、TeX や LaTeX はそのフォントをどのように配置したらよいのかわからないので、困ったあげく止まってしまうのです。そこで、jfm ファイルをつくってやります。「つくる」といっても、別に難しいことをするわけではありませんから安心してください。すでにある明朝体用の jfm ファイルをリネームコピーしてやればよいのです。

ここで、なぜ明朝体用のそれですむかという点、「キャッシュフィルタの、より進んだ利用」(p.170)の途中、p.174 での日本語フォントが固定幅フォントであるということの説明に、その理由が見出せます。多くの場合、日本語フォントは固定幅かつ固定高で作成されているために、同じ大きさのフォントに対する文字の配置情報は、基本的にどのフォントでも変わりがないのです。したがって、丸文字だけではなく、ツアイトの書体倶楽部の教科書体や毛筆体を使用する場合にも、ここでの jfm ファイルの作成は、まったく同じように応用できることになります。

さて、jfm ファイルは、%TEXHOME%\fonts 配下にあります。このディレクトリ内の min\*.tfm が、明朝体用の jfm ファイルです。jfm ファイルとはいっても、拡張子は tfm ファイルと同様に .tfm ですから、混乱しないでください<sup>22)</sup>。

あとはこれを maru\*.tfm にリネームコピーすれば、今度は LaTeX の処理を通るようになるはずです。

```
A> cd %TEXHOME%\fonts
A> copy min5.tfm maru5.tfm
A> copy min6.tfm maru6.tfm
A> copy min7.tfm maru7.tfm
A> copy min8.tfm maru8.tfm
A> copy min9.tfm maru9.tfm
A> copy min10.tfm maru10.tfm
```

縦書きを使用することがある人は、tmin\*.tfm についても同じように tmaru\*.tfm にコピーしておくといでしょう。

## (4) プレビューアで見てみましょう。

その前に、TeXenv.bat で設定している環境変数 PREVIEW.P3M および PRINT.P3M の設定内容を、それぞれ %TEXHOME%\drivers\doc\preview.p3m および %TEXHOME%\drivers\doc\print.p3m に変更し、環境変数を再設定してください。

```
A> set PREVIEW.P3M=%TEXHOME%\drivers\doc\preview.p3m
A> set PRINT.P3M=%TEXHOME%\drivers\doc\print.p3m
A> preview sample
```

① newfm\_fix.lzh その 2 ～ $\text{\textbackslash F-LAT}_{\text{E}}\text{X}$  と newfm\_fix.lzh～

「 $\text{\textbackslash L}_{\text{A}}\text{T}_{\text{E}}\text{X}$  の処理」(p.80)で  $\text{\textbackslash L}_{\text{A}}\text{T}_{\text{E}}\text{X}$  のユーザは、満足のできるプレビュー結果を得られるまで、ソースの修正 →  $\text{\textbackslash L}_{\text{A}}\text{T}_{\text{E}}\text{X}$  による処理 → プレビューでの閲覧という処理を繰り返さなければならない旨について述べました。

ところが、空きメモリが不足ぎみの人が  $\text{\textbackslash L}_{\text{A}}\text{T}_{\text{E}}\text{X}$  を使用する場合には、このような処理の繰り返しがひどく面倒になることがあります。どうということかという、本書添付のインストーラでインストールした場合に作成される myfonts.fm を使用した場合はともかく、汎用のコンフィギュレーションファイルとして作成された fontman.fm においてダイエットフィルタやキャッシュフィルタを使用した場合には、フォントマネージャの常駐領域がかなり大きなものになるため、フォントマネージャの常駐を解除しなければ  $\text{\textbackslash L}_{\text{A}}\text{T}_{\text{E}}\text{X}$  の処理ができないこともあるということです。

実は、このような場合にも、異なるコンフィギュレーションファイルで複数回常駐することができるというフォントマネージャのメリットの一端が顕れます。p.180 のコラム「その 1」に見出されるように、一定程度に用途を特定した複数のコンフィギュレーションファイルによってフォントマネージャを登録しておけば、それぞれのコンフィギュレーションファイルによる ldepth は、すべてのフォント定義をひとつのコンフィ

ギュレーションファイルによって実現している場合と比べて、常駐も解除も短時間で可能になります。このことは、フォントマネージャのすべての常駐部分を解除しなくても、「 $\text{\textbackslash L}_{\text{A}}\text{T}_{\text{E}}\text{X}$  を起動可能にするために必要な、最小限の常駐領域だけを解放する」ということが可能だということを意味しています。

この発想を、添付ディスクが構築する  $\text{\textbackslash T}_{\text{E}}\text{X}$  システムに用意されているコンフィギュレーションファイル以上に突き詰めたのが、第 6 章「 $\text{\textbackslash T}_{\text{E}}\text{Xfamily}$ 」(p.234)で紹介する  $\text{\textbackslash F-LAT}_{\text{E}}\text{X}$  の作者の 1 人である許先明 (Seirios) 氏が作成し、本書の添付ディスク 8 にも収録されている newfm\_fix.lzh です。

newfm\_fix.lzh は、 $\text{\textbackslash F-LAT}_{\text{E}}\text{X}$  の使用に際して、「使うフォントだけ常駐させる」という発想のもと、(1) 明朝体・ゴシック体という基本フォントの登録用、(2) 丸文字・毛筆体・教科書体という追加フォント登録用、(3) 『新書体倶楽部』(ツァイト)のゴシック体(角細・角太・角極太・丸細・丸中・丸太・丸極太)の登録用、(4) 変形書体(斜体・平体・長体)の登録用、(5) 飾り文字(影文字・網文字・立体文字・白抜き文字)の登録用という、5 つのコンフィギュレーションファイルを用意しているので、ユーザがこれらのなかから選択して使用することができるようになっています。

丸文字フォントが使用されているはずですが、 $\text{\textbackslash Large}$  など、文字の大きさを変更するコントロール・シーケンスには、まだ対応していません<sup>23)</sup>。そのような本格的な使用を考えている場合は、スタイルファイル等の修正が必要になります。スタイルファイルの修正のしかたについては本書では扱いませんので、他の  $\text{\textbackslash T}_{\text{E}}\text{X}$  関連の書籍をご覧ください。

23)  $\text{\textbackslash font}$  あるいは  $\text{\textbackslash newfont}$  で登録したフォントは、登録時に指定した大きさでしか使用できません。



## 5.3 .....print.cfg の作り方

X680x0 用のプリンタドライバ `print.x` には、現時点においても各社から発売されているさまざまなプリンタを使用するためのコンフィギュレーションファイルが多数用意されています。しかし、今後発売される製品はいうまでもなく、現在発売されているすべてのプリンタのコンフィギュレーションファイルが揃っているわけではありません。

本節では、`TeX` で作成した文書を、コンフィギュレーションファイルが用意されていないプリンタで出力する場合のために、プリンタのコンフィギュレーションファイルを作成する方法について例を示して解説します。

ここでは、値段も手ごろで、性能もそこそこあり、ユーザも多いと思われるキャノンの BJ-10 を取り上げてみたいと思います<sup>1)</sup>。

ただし、本節はプリンタドライバのリファレンスを熟読していないと理解が困難であるかもしれません。まだプリンタドライバのリファレンスを読んでいない場合には、本節を読み飛ばすか、あるいは少なくともプリンタドライバのリファレンスを適宜参照しながら読むようにしてください。

なお、本節において使用する語句の一部を、ここで定義しておきます。

- 「プリンタドライバ」とは、特に断りがないかぎり、本書がサポートするプリンタドライバ `print.x` を表す。
- 「リファレンス」とは、プリンタドライバのリファレンスマニュアル (『Vol.2 — Reference 編』の第 2.4 節「PRINT オプション」(p.54)) を表す。
- 「マニュアル」とは、特に断りがないかぎり、キャノン『BJ-10v バブルジェットプリンタ 操作説明書』を表し、[ ] 内にその参照ページを示す。

### 5.3.1 プリンタドライバのオプション

BJ-10にかぎらず、一般的にプリンタドライバのオプションのうちで実質的にプリンタの仕様によって左右される性質のものは、以下に示すようなドライバ制御系の一部と汎用プリンタ制御系の全部です。

- ドライバ制御系
  - `-dpi` 解像度指定

1) キャノンの BJ-10 のコンフィギュレーションファイルは、すでにプリンタドライバでサポートされているために、本書の添付ディスクが構築する X680x0 版の `TeX` システムにも含まれています。ここでは、コンフィギュレーションファイルの作成方法のサンプルとして取り上げているにすぎません。

- `-width` バッファ横ドット数指定
  - `-height` バッファ縦ドット数指定
- 汎用プリンタ制御系
- `-MSBisUpper` 印字ヘッド方向指定
  - `-MSBisLeft` 印字ヘッド方向指定
  - `-pinBytes` 印字ヘッドのピン数指定
  - `-pinHeights` 印字ヘッドのピン数指定
  - `-prBufSize` プリンタバッファの大きさ指定
  - `-init` プリンタ初期化のコード列指定
  - `-CRLF` プリンタ改行のコード列指定
  - `-extraCRLF` 空行改行時の追加コード列指定
  - `-FF` プリンタ改ページのコード列指定
  - `-graphic` ドットグラフィック指定のコード列指定
  - `-start` ドット単位印字開始位置指定のコード列指定
  - `-relative` ドット単位印字位置相対指定のコード列指定
  - `-repeat` グラフィックリピート指定のコード列指定

リファレンスに詳しく述べられていますが、ドライバ制御系のオプションはプリンタの制御に直接かかわるわけではありません。あくまでプリンタドライバ内部に対する設定ですから、ドライバ制御系のオプションを設定しないからといって、期待どおりの出力がまったく得られないというようなことはまずないのです。せいぜい、用紙に対してずれて出力されてしまったり、小さい文字で印字されてしまったりする程度のことでしょう。

では、何のためにドライバ制御系のオプションをコンフィギュレーションファイルに設定するのでしょうか。その理由は、プリンタドライバ使用に際しての手間を省くことにあります。たとえば、360dpi のプリンタであれば、普通は 360dpi で出力するでしょう。にもかかわらず、プリンタドライバを使用するたびに、いちいちコマンドラインで `-dpi=360` を指定するのは面倒です。そこで、コンフィギュレーションファイルに「`-dpi=360`」を記述しておくことで、このオプションを常時コマンドラインで指定しなくてもすむようにします。

つまり、ドライバ制御系のオプションは、「出力する原稿あるいは好みに応じて設定すべき値が変わる余地はあるけれども、ふだん使用する値を設定しておくことで、印刷するたびごとにそれを指定する手間を省く」という目的でコンフィギュレーションファイルに記述するのです。

他方、プリンタ制御系のオプションでは、設定が誤っていればまともな出力が得られなくなる可能性が高くなります。たとえば、行間が間延びして想定していた用紙範囲に原稿が入らなかったり、ひどいときには正体不明の文字の羅列が出力されてしまったりすることもあります。

このようになる理由は、「プリンタ制御系」という名のとおり、プリンタ制御系のオプションが直接プリンタの制御にかかわるからにほかなりません。どういうことかという、プリンタ制御系のオプションに指定する引数は、プリンタの仕

2) ラスタプリンタとは、ページプリンタなど、ラスタグラフィックの出力をサポートしているプリンタのことです。ラスタとは、テレビなどに見られる 1 本の走査線 (scanning line) のことで、ラスタグラフィックとは行と列に配列されたドットまたはピクセルによってグラフィックを生成する方法を意味します。具体的には、イメージデータを構成する横方向の 1 ドット列を、縦方向に連続させることで出力可能なプリンタを「ラスタプリンタ」と呼びます。

様を定めるものであったり、プリンタの制御を行うためにプリンタに送るコマンド列であったりしますから、これらのオプションに誤った値を設定した場合、プリンタがエラーや誤動作を起こして、期待したような出力をしてくれないのです。

つまり、プリンタ制御系のオプションは、「プリンタが同じであるかぎり、誰でも、どこでも、どの原稿にも、同じ設定をすることが必要 (あるいは効率的) である」という理由から、コンフィギュレーションファイルとして記述することになります。

リファレンスには、これらのオプション以外にもう一系統、ラスタプリンタ制御系について説明されています。このオプション系は、使用するプリンタがラスタプリンタ<sup>2)</sup>の場合に追加指定することによって印字効率を飛躍的に上げることができます。しかし、バブルジェット方式の BJ-10 には関係ないので、本節では触れません。

### 5.3.2 ドライバ制御系を設定する

#### ◆ -dpi

BJ-10 のマニュアル [VI-2] によれば、BJ-10 の印字精度が 360dpi であることがわかります。

したがって、基本となる解像度として 360 を設定します。

List 5-15 ● 基本解像度の設定

```
1: -remark= <<<<< Canon BJ-10v Configuration File >>>>>
2: -remark= テスト版
3: -remark=
4: -remark= << ドライバ制御系 >>
5: -dpi=360
```

なお、-remark は注釈文を記述するためのオプションです。-remark オプションには注意しなければならないことが一点あります。それは「引数に半角スペースを用いてはならない」ということです。このため、空白を使用する場合には、かなスペースや全角スペースを使用しなければなりません。

#### ◆ -width, -height

大ざっぱな言い方をすれば、プリンタドライバはメモリ上に -height と -width で指定された大きさの「キャンバス」を確保し、ここに出力すべき 1 ページの「絵」を描いては、これをプリンタで「ハードコピー」しています。したがって、ここで用意する「キャンバス」は「絵」を描くために必要な大きさがあれば十分



### ◎ $\text{\TeX}$ が想定するデフォルトサイズ

本文中で「多くの文書が  $\text{\TeX}$  のデフォルトサイズであるレターサイズを使用している」と述べていますが、これは正しくもありませんが、誤りでもあります。

確かに Knuth 博士が作成したオリジナルの  $\text{\TeX}$  が想定するデフォルトサイズはレターサイズでした。しかし、本書がサポートしているアスキー社が開発した 日本語  $\text{\TeX}$  のうち、多くの人が使用するであろう 日本語  $\text{\LaTeX}$  を見ると、スタイルファイルオプションで原稿サイズを指定しなかった

場合のデフォルトサイズは、A4 サイズになっているのです。

とはいえ、通常の使用をする時点で、このことが原因となって問題が生じることはほとんど考えられません。しかし、皆無とはいいきれませんから、もし用紙に余裕があるにもかかわらず出力が欠けてしまったり、用紙に対してずれた位置に出力されてしまったりというような場合があったら、ここで説明しているバッファのサイズについて見なおしてみるとよいかもしれません。

ですから、`-height` と `-width` は、プリンタにではなく、原稿にあわせて設定することになります。

たとえば A4 原稿の場合、A4 用紙の大きさは  $297\text{mm}^3) \times 210\text{mm}$  です。`-height` と `-width` にはドット数を与えることになっていますから、BJ-10 の場合、この長さを 360dpi 時のドット数に換算する必要があります。ここで、360dpi とは 360 dot per inch、すなわち  $1\text{in}^4)$  内が 360 ドットという解像度であることを意味します。さらに  $1\text{in} = 25.4\text{mm}$  ですから、用紙の大きさをドット数に換算すると、`-height` は  $297/25.4 \times 360 \div 4209$  ドット、`-width` は  $210/25.4 \times 360 \div 2976$  ドットになります。ただし、`-height` と `-width` には 8 の倍数を与えることになっていますから、用紙の大きさの値を超えない範囲の 8 の倍数に最も近い値、`-height=4208` および `-width=2976` となります。

このようにして、原稿の大きさに対して用意すべき「キャンバス」の大きさを求めることができます。360dpi のプリンタの場合は以下のとおりです。

### ○ 360dpi における用紙の大きさ

デフォルト : `-height=1980(dot) \times -width=1440(dot)`  
 レターサイズ :  $3960\text{dot} (11\text{in} \div 279\text{mm}) \times 3060\text{dot} (8.5\text{in} \div 216\text{mm})$   
 A4 サイズ :  $4208\text{dot} (297\text{mm}) \times 2976\text{dot} (210\text{mm})$   
 B5 サイズ :  $3640\text{dot} (257\text{mm}) \times 2576\text{dot} (182\text{mm})$

$\text{\TeX}$  の場合、多くの文書が  $\text{\TeX}$  のデフォルトサイズ<sup>5)</sup> であるレターサイズを使用していることもあり、仮にコンフィギュレーションファイル内でバッファのサイズを指定しておく必要があるならば、レターサイズで十分であると推測しました。

ところが、BJ-10 の最大印字可能幅は  $203.2\text{mm} = 8\text{in}$  (マニュアル [VIII-3]) でしかありません。この幅は、いうまでもなく、レターサイズを縦置きにした場合の幅  $216\text{mm} = 8.5\text{in}$  よりも狭いものです。では、BJ-10 ではレターサイズや A4 サイズの原稿は出力できないのでしょうか？

もちろん、そんなことはありません。

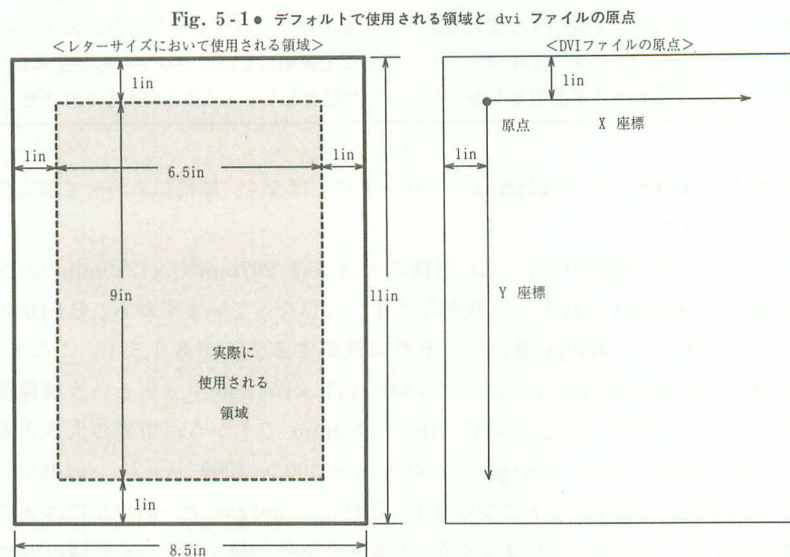
3)mm : ミリメートルの単位記号。 $\text{\TeX}$  で使用できる単位表記のひとつ。

4)in : インチの単位記号。 $\text{\TeX}$  で使用できる単位表記のひとつ。

5) ソースファイル中、たとえばスタイルファイルオプションで、文書サイズを設定したスタイルファイルを読み込んだり (たとえば `a4j` を指定して、A4 用紙の大きさを設定したり)、あるいは独自に文章領域を設定したりしない場合に採用されるサイズのこと。

ただ、その理由の説明には  $\text{T}_{\text{E}}\text{X}$  の内部処理の問題もかかわってきますので、少し遠回りの解説をしますが、我慢しておつきあいください。

$\text{T}_{\text{E}}\text{X}$  はユーザが特別に設定をしないかぎり、上下左右に 1 インチずつの余裕ができるように組版します。たとえば  $11\text{in} \times 8.5\text{in}$  のレターサイズの下稿を作成した場合、実際にはこのサイズの範囲がすべて使用されるわけではありません。 $\text{T}_{\text{E}}\text{X}$  は上下左右に 1 インチの余白をつくるように組版しますから、実際に使用される領域は  $9\text{in} \times 6.5\text{in}$  の範囲でしかないのです (Fig. 5-1)。



$\text{T}_{\text{E}}\text{X}$  によって出力された dvi ファイルは、内部的な座標系の原点を左から 1 インチ、上から 1 インチの位置に設定していますから、 $11\text{in} \times 8.5\text{in}$  のレターサイズの下稿の場合であれば、実際に使用される領域である  $9\text{in} \times 6.5\text{in}$  の左上のポイントが原点に設定されていることになります。仮にここで、その dvi ファイルの情報をレターサイズ紙に出力した場合、BJ-10 の場合にかぎらず、 $\text{T}_{\text{E}}\text{X}$  で実際に使用された領域である  $9\text{in} \times 6.5\text{in}$  が紙の左上から印字されることになってしまいます。つまり、本来想定されていたはずの上下左右 1 インチの余白は出力されず、全体的に左上方に寄った形で出力されてしまうのです。

そこでプリンタドライバは、 $\text{T}_{\text{E}}\text{X}$  が想定しているように上下と左右に 1 インチの余白をつかって原稿が出力されるように補正を行います。つまり、`-height×-width` で与えられたバッファ (「キャンバス」) に対して、印刷する原稿 (「絵」) が中央にくるように、上下および左右に 1 インチ以内の空白を割り付けるのです。

したがって、プリンタドライバが確保するバッファのサイズと  $\text{T}_{\text{E}}\text{X}$  原稿のサイズの関係には、以下の 3 つが考えられます。

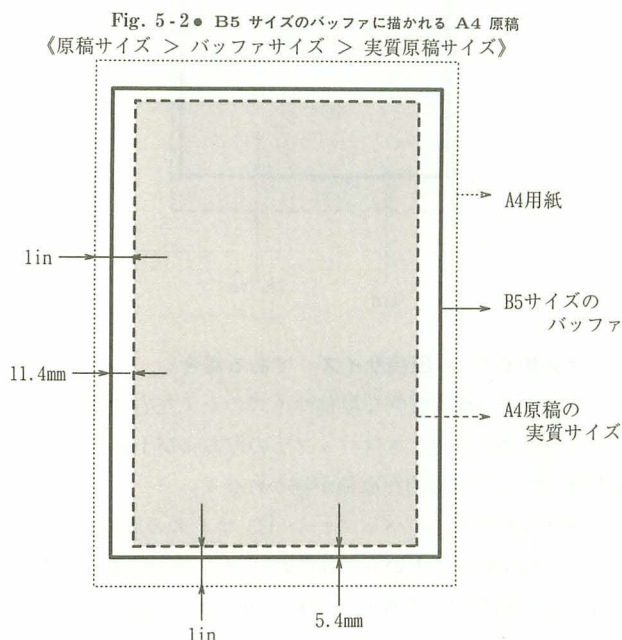
- (1) 「バッファサイズ < 原稿サイズ」であるものの、「バッファサイズ > 実質的な原稿のサイズ」である場合。

このとき、プリンタドライバによってバッファの中央に位置するように上下左右に余白がつけられ、実質的な原稿が完全な形で描かれます。

たとえば「B5サイズのバッファ < A4サイズの実稿」の場合、A4<sup>6)</sup> 原稿の実質的なサイズは、A4 本来の大きさから、 $\text{T}_{\text{E}}\text{X}$  が想定する左右上下の余白各々 1 インチを除いた大きさ、 $(297\text{mm} - 2\text{in}) \times (210\text{mm} - 2\text{in}) = 246.2\text{mm} \times 159.2\text{mm}$  であり、これは B5<sup>7)</sup> サイズより小さいサイズです。よって A4 の実質的原稿は、上下にそれぞれ  $257 - 246.2/2 = 5.4\text{mm}$ 、左右にそれぞれ  $182 - 159.2/2 = 11.4\text{mm}$  の余白を割り付けられ、B5 サイズのバッファの中央に収まるように描かれます (Fig. 5-2)。

6) 297mm×210mm。

7) 257mm×182mm。



- (2) 「バッファサイズ < 原稿サイズ」であり、しかも「バッファサイズ < 実質的な原稿のサイズ」である場合。

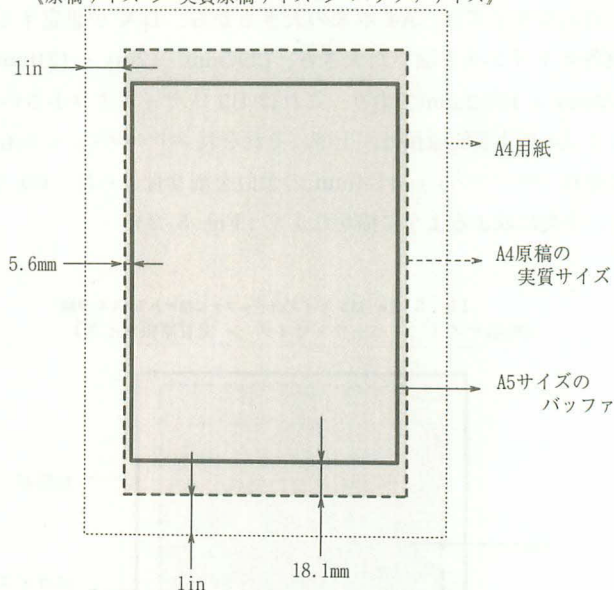
このときは、バッファに入りきらなかった原稿の上下や左右が削られてしまうことになります。

たとえば「A5サイズのバッファ < A4サイズの実稿」の場合、A4 原稿の実質的な原稿のサイズ  $246.2\text{mm} \times 159.2\text{mm}$  は A5<sup>8)</sup> サイズより大きいので、A4 の実質的原稿における上下それぞれ  $246.2 - 210/2 = 18.1\text{mm}$ 、左右それぞれ  $159.2 - 148/2 = 5.6\text{mm}$  が A5 のバッファに入りきらずに消失してしまいます (Fig. 5-3)。

8) 210mm×148mm。



Fig. 5-3 ● A5 サイズのバッファに描かれる A4 原稿  
《原稿サイズ > 実質原稿サイズ > バッファサイズ》

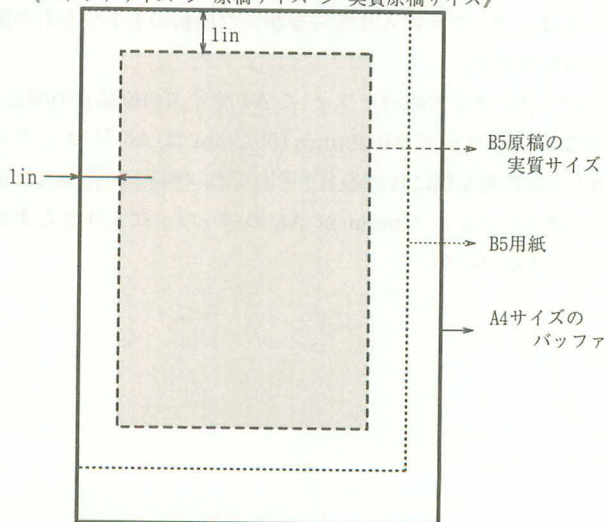


(3) 「バッファサイズ > 原稿サイズ」である場合。

プリンタドライバが実質的な原稿サイズの上下左右に付加できる余白は 1 インチまでのため、このときはバッファの左および上からそれぞれ 1 インチの位置を原点として実質的な原稿が描かれます。

たとえば「A4 サイズのバッファ > B5 サイズの原稿」の場合、A4 サイズのバッファの左および上からそれぞれ 1 インチの位置を原点として B5 原稿の実質的な原稿が出力されることになり、B5 サイズのバッファに B5 サイズの原稿を描き込む場合と同様の位置に描き込まれるのです (Fig. 5-4)。

Fig. 5-4 ● A4 サイズのバッファに描かれる B5 原稿  
《バッファサイズ > 原稿サイズ > 実質原稿サイズ》



以上の結果から、上下左右に 1 インチずつの余白を想定するというフォーマットを守っているかぎり、BJ-10 でもレターサイズや A4 サイズの原稿がバッファの中央に収まるように出力できるようになります。したがって、`-width` には BJ-10 の最大印字可能幅 203.2mm (= 8in = 2880dot) を、`-height` にはレターサイズの縦寸法 279mm (= 11in = 3960dot) を設定することにします。

List 5-16 ● 印字可能領域 (バッファ) の設定

```
6: -width=2880
7: -height=3960
```

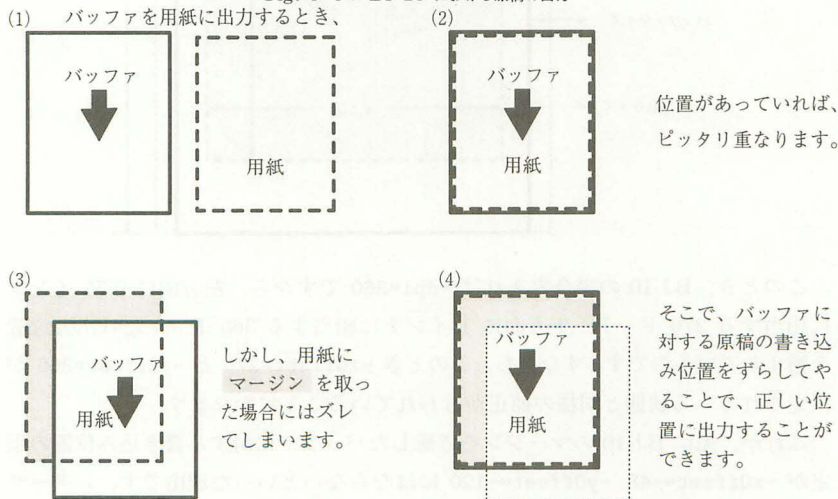
ところで、このようにして用意する「キャンバス」が大きければ大きいほど、より多くのメモリが消費されることはいうまでもありません。確保されるバッファサイズは  $\text{width} \times \text{height} \div 8$  バイトで、この場合  $2880 \times 3960$  ですから、約 1.4M バイトが必要になります。

#### ◆ `-xOffset, -yOffset`

BJ-10 は、紙に対して 3.4mm の左マージン、8.5mm の上マージンをとるという仕様 (マニュアル [VIII-3]) になっています。そのため、原稿を原稿と同じ大きさの紙に出力しようとする (たとえば、A4 原稿を A4 用紙に出力する) 場合、全体として出力が右下にずれてしまいます。そこで、BJ-10 が付加するマージンの分だけ書き込み位置を左上にずらしてやらなければ正しい位置に印字されません (Fig. 5-5)。

このような補正を実現するオプションが `-xOffset, -yOffset` です。`-xOffset` および `-yOffset` は、`-width` と `-height` で与えられたバッファの左上を原点と

Fig. 5-5 ● BJ-10 における原稿の出力



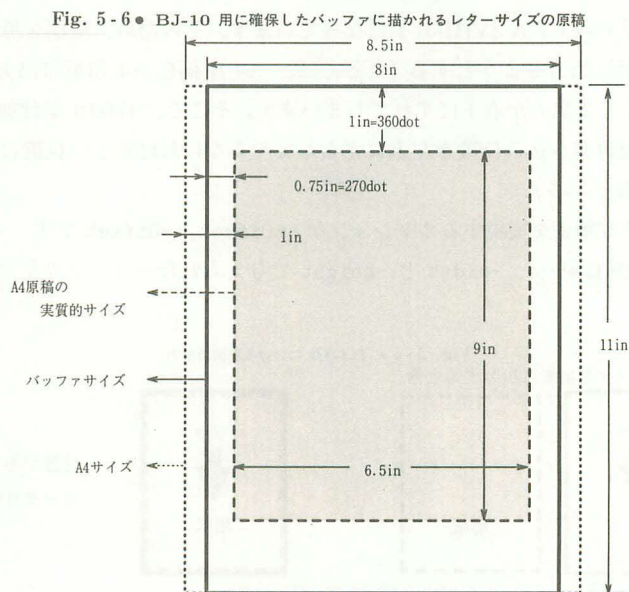
して、右方向が X 軸の正の方向、下方向が Y 軸の正の方向となる座標系における書き込み開始位置（つまり、原稿の左上端の位置）をドット単位で指定します。

したがって、この場合、マージンの長さをドットに換算して、それぞれの座標軸における負の方向に補正をすることで、出力が左上方に移動することになります。BJ-10 は 360dpi で、 $1\text{in} = 25.4\text{mm}$  ですから、左マージンである  $3.4\text{mm}$  をドット数に換算しますと、 $3.4/25.4 \times 360 \doteq 48$  ドット、同様に上マージン  $8.5\text{mm}$  は  $8.5/25.4 \times 360 \doteq 120$  ドットとなります。

では、`-xOffset=-48`、`-yOffset=-120` という設定になるのかというと、そうではありません。

なぜなら、すでに述べたように、プリンタドライバは、`-height` および `-width` によって確保されたバッファの中央に原稿が出力されるように、上下と左右にそれぞれ等しい 1 インチまでの空白を自動的に割り付けているからです。

どういうことかといいますと、 $11\text{in} \times 8.5\text{in}$  のレターサイズ原稿を印字する場合、「`-width, -height`」(p.188) で確保した  $11\text{in} \times 8\text{in}$  のバッファ（「キャンバス」）の中央に、実質  $9\text{in} \times 6.5\text{in}$  の原稿（「絵」）を配置するために、プリンタドライバは自動的に右方向に 0.75 インチ、下方向へ 1 インチだけ印字開始位置を補正していることとなります（Fig. 5-6）。



このとき、BJ-10 の場合であれば `-dpi=360` ですから、左方向に 0.75 インチに相当する 270 ドット、下方向に 1 インチに相当する 360 ドットだけ出力位置を補正しているのです。すなわち、このとき `-xOffset=270` と `-yOffset=360` が指定されている状態と同様の補正が行われていることとなります。

これが、先に BJ-10 のマージンを考慮したバッファに対する書き込み位置の指定が `-xOffset=-48`、`-yOffset=-120` にはならないといった理由です。レターサイズ原稿を「`-width, -height`」(p.188) で確保したバッファに書き込む際、その書き込み開始位置はすでに左に 270 ドット、下に 360 ドット補正されていま



す。ですから、BJ-10 のマージンを考慮して 48 ドット左に出力位置を補正してやるためには、バッファの X 軸に対して  $270 - 48 = 222$ 、同じく 120 ドット上に出力を補正するためにはバッファの Y 軸に対して  $360 - 120 = 240$  のポイントを、原稿の書き込み開始位置に指定してやらなければならないのです。よって、次のように設定できます。

**List 5-17 ● 書き込み位置の設定**

```
8: -xOffset=222
9: -yOffset=240
```

ところで、「-width, -height」(p.188) で -width と -height を決定しましたが、ここで設定したバッファサイズ<sup>9)</sup> に対して、上下にそれぞれ上マージンである 8.5mm、左右にそれぞれ BJ-10 の左マージンである 3.4mm を加えると、A4 サイズ<sup>10)</sup> にほぼ相当します。したがって、-width=2880 および -height=3960 で確保したバッファに対して書き込まれたレターサイズ<sup>11)</sup> の原稿を A4 用紙に出力する場合、BJ-10 ではマージンを考慮した補正は必要なくなることがわかります。それゆえ、レターサイズとはいっても、通常は A4 用紙に出力することが多いだろうと考え、今回は -xOffset, -yOffset を設定しないことにします。

9) 11in×8in  
(279mm×203.2mm).  
10) 297mm×210mm.

### 5.3.3 プリンタ制御系を設定する

第 5.3.1 項「プリンタドライバのオプション」(p.186) で述べたように、プリンタ制御系のオプションの引数はプリンタの仕様であったり、コマンドであったり、直接プリンタの制御にかかわるものです。それだけに、お手元のプリンタが BJ-10 であるかどうかにかかわらず、あなたのプリンタのマニュアルを手元におき、参考にしながら本項を読まれることをお勧めします。

◆ -MSBisUpper, -MSBisLeft

「-width, -height」(p.188) の -width と -height の解説のなかで、プリンタドライバがメモリ上に描かれたページの「絵」を「ハードコピー」しているのだと書きました。これはたとえというには大ざっぱですが、実際プリンタドライバはプリンタでビットイメージによる印字を行っているのです<sup>11)</sup>。

ただ、このとき実際に出力されるイメージデータとプリンタに送られるビットイメージデータとの関係は、プリンタの印字方式の違いによって異なってきます。

ひとつは、従来のシリアルプリンタに代表される印字方式をとるタイプです。シリアルプリンタは、縦の 1 ドット列にあたる印字ヘッドを左から右、あるいは右から左に移動しながら、1 文字ずつ印字していきます。これはイメージデータの出力の際にも変わりません。縦の 1 ドット列を横に連続させることで 1 行分を出力し、さらにこの 1 行を縦に連続させることで実際のイメージデータの出力

11) したがって、本書がサポートしているプリンタドライバではプリンタに内蔵されているフォントはいっさい使用できません。日本語の場合はフォントマネージャによって登録されているもの、英数記号の場合は METAFONT で作成されているものにかぎって TeX 上で使用できます。しかし、どうしてもプリンタの内蔵フォントを使用したい場合、たとえば LIPS3 や ESC/Page のコマンド体系を持つプリンタの内蔵フォントを使用したいというのであれば、第 6 章「TeXfamily」(p.261) で紹介する lips3dvi のような、本書がサポートする以外のプリンタドライバを使用する必要があるでしょう。

を実現していることになります。

もうひとつ、最近のページプリンタに代表されるラスタ出力方式をとるタイプがあります。ページプリンタには、シリアルプリンタでいうところの印字ヘッドがありませんから、事実上、「行」の概念は消失し、行単位で印字しなければ効率が悪くなるという制約も存在しません。そのために、イメージデータの出力の際に、単純にイメージデータの横の1ドット列を縦に連続させていくことで、実際のイメージデータの出力を実現することができるようになります。

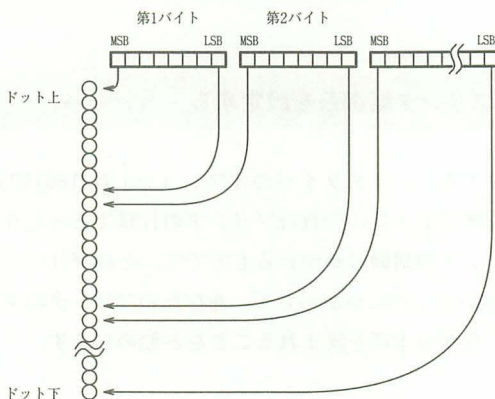
したがって、実際に出力されるイメージデータと、プリンタに送られるビットイメージデータの関係としては、以下の2つ(正確には3つ)の方式があります。

- プリンタに送られるデータが、出力したいビットイメージの縦のドット構成に対応するもの。

- このとき、データのバイトごとの頭が、印字ヘッドのピン配列の上にくるタイプ (Fig. 5-7)。

ESC/P コマンド系のプリンタはこのタイプである。

Fig. 5-7 ● MSB が上方になるプリント  
ビット イメージ データ



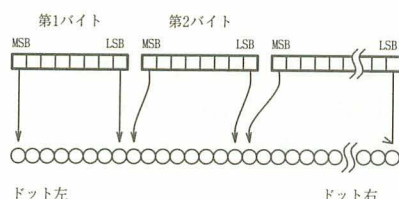
- このとき、データのバイトごとの頭が、印字ヘッドのピン配列の下にくるタイプ。Fig. 5-7 における、印字ヘッドのピンに対応する各データの MSB と LSB とを入れ替えたものをイメージしてほしい。

PC-PR コマンド系のプリンタはこのタイプである。

- プリンタに送られるデータが、出力したいビットイメージの横のドット構成に対応するもの。このとき、データのバイトごとの頭が、出力したいビットイメージデータの左方にくる (Fig. 5-8)。この方式に対応したプリンタとしては、たとえば横河・ヒューレット・パッカートの DJ-505J がある<sup>12)</sup>。

12) DJ-505J は -MSBisUpper の設定も可能なプリンタです。しかし、ラスタグラフィックの出力をサポートしたモードがあるため、高速化のために同モードで出力を行います。このために -MSBisLeft を設定します (HPDJ505J.cfg を参照)。

Fig. 5-8 ● MSB が左方になるプリンタ  
ラスタ グラフィック データ



BJ-10 の場合、マニュアル [IV-46] のビットイメージモードの選択の箇所に説明されるように、プリンタに送られるデータが、出力したいビットイメージの縦のドット構成に対応し、この各データバイトの最上位ビット<sup>13)</sup> が印字ノズルの上方にきていますから、最初に説明したタイプのプリンタに該当することがわかります。そこで、オプションは次のように設定しなければなりません。

13) Most Significant Bit  
: MSB。

List 5-18 ● MSB が上方にくるプリンタの場合

```
8: -remark=
9: -remark= << プリンタ制御系 >>
10: -MSBisUpper
```

もしここで、プリンタに送られるビットイメージデータの作り出すドット構成がプリンタヘッドのピンの縦方向に対応しているものの、各データの MSB は下方になるという場合には、ここではなんのオプションも指定しません。

List 5-19 ● MSB が下方にくるプリンタの場合

```
10: -remark=なにも指定しない
```

また、プリンタに送られるビットイメージデータの作り出すドット構成がプリンタヘッドのピンの横方向の並びに対応した場合には、プリンタに送るデータのバイトごとの最上位ビットは左にきます。このような関係にあるなら、-MSBisUpper でも無指定でもなく、次の設定をしなければなりません。

List 5-20 ● MSB が左方にくるプリンタの場合

```
10: -MSBisLeft
```

つまり、これらのオプションは、いずれかひとつを必ず設定する<sup>14)</sup> 必要があるのです<sup>15)</sup>。

14) 漠然と「指定しない」ではなく、意識的な「無指定の設定」がありうるということです。

BJ-10 以外のプリンタでも、ビットイメージでの印字に関するコマンドの説明のなかで、この仕様についての解説が図入りでなされているのが一般的です。ここまでの説明がよくわからなかった人は、プリンタのマニュアルの該当箇所をもう一度よく読みなおしてください。CZ-8PC4 の場合であれば、取扱説明書の p.149 の図が理解を助けるでしょう。

15) ただし、ラスタプリンタ系のオプション -Raster を設定した場合には、これらの設定は必要ありません。

それでもわからなければ、ご自分で設定を変えて試してみてください。



## ◆ -pinBytes, -pinHeights

「-MSBisUpper, -MSBisLeft」(p.195)で、-MSBisUpper を設定した場合、およびオプションを指定しなかった場合には -pinBytes に 印字ヘッドのピン数/8 を指定し、-MSBisLeft を指定した場合は -pinHeights に印字ヘッドの高さを指定します。

-MSBisUpper を設定する説明で述べたように、BJ-10 はプリンタに送るビットイメージデータの各バイトごとの最上位ビットが、プリンタヘッドのピンの縦方向の並びに対応しています。

BJ-10 の場合、-MSBisUpper を設定しましたので、-pinBytes を設定しなければなりません。BJ-10 は 48 ビットのイメージ印字が可能ですから、ピン数/8 の指示にもとづいて次のように設定します。

## List 5-21 ● MSB が上方にくるプリンタの場合

```
10: -MSBisUpper
11: -pinBytes=6
```

もしここで、先のオプションを無指定にすべきプリンタの場合には、かわりに次のように指定します。“?” には、ピン数/8 にあたる値を指定しなければなりません。

## List 5-22 ● MSB が下方にくるプリンタの場合

```
10: -remark=なにも指定しない
11: -pinBytes=?
```

一方、-MSBisLeft を指定すべきプリンタの場合には、-pinBytes のかわりに次のように設定します。

## List 5-23 ● MSB が左方にくるプリンタの場合

```
10: -MSBisLeft
11: -pinHeights=?
```

“?” には印字ヘッドの高さを入れます。印字ヘッドの高さには 1 を設定するのが効率的です。なぜなら、ラスタグラフィックの出力は横の 1 ドット列を連続させることで実現されているからです。

このオプションは、どちらか一方を必ず設定しなければなりません。

## ◆ -prBufSize

-prBufSize には、プリンタに送るデータのために確保するスプールバッファの大きさを指定します。

具体的にいうと、-prBufSize が指定されているとき、プリンタドライバは -prBufSize に設定されているだけのスプールバッファを確保し、プリンタに送

るべきデータがプリンタ固有のバッファからあふれると、これをスプールバッファにため込みます。このようにプリンタ固有のバッファがあふれるような事態は、たとえば底に穴のあいたビニール袋から (穴を塞がずに) 水をあふれさせるためにはどうすればよいかを考えればわかるように、プリンタがデータを処理する以上のスピードで情報を送り込まなければ起こり得ません。つまり、X680x0 本体が使用するプリンタよりも高速な場合、送られてくるデータをプリンタが取りこぼしてしまうことがあるので、これに対処するためのスプールバッファの大きさを定めるのが本オプションなのです。

したがって、このオプションに設定すべき値は、プリンタがどの機種であるのかだけではなく、X680x0 本体がどの機種であるのかによっても大きく左右されますので注意が必要です。つまり、処理速度の遅い X68000 で使用するとき設定した値が、より高速な X68000XVI, X68030, X68030 + 040turbo<sup>16)</sup> などで使用した場合にも通用するとはかぎらないということです。実際、キヤノンの LASER SHOT B406 を X68030 で使用する場合、`-prBufSize` はデフォルトの 256K バイトで支障ありませんが、X68030 + 040turbo で使用する場合にこの値のままにしておくと、1 ページめを出力したあと、プリンタも本体も止まってしまいます。

BJ-10 は、処理そのものが高速というほどでもないうえに、プリンタ固有のバッファが 8K バイトしかありません。ですから本来ならば若干大きめのバッファを確保したほうがよいのですが、具体的な値が思いつかないことに加え、少なくとも BJ-10 のデータ処理速度が本節の動作確認に使用する無改造の X68000 の処理速度に負けることはないだろうという予測もあって、ここではデフォルトの 256K バイトを採用することにします。

List 5-24 ● プリンタ用のスプールバッファを設定

```
12: -prBufSize=256
```

#### ◆ -init

プリンタをイニシャライズするためのコード列を設定します。`-init` で指定されたコード列は 1 ページの始まりごとにプリンタへ送られますから、ここでは初期化 (電源投入時の状態に戻すという意) のみではなく、初期設定も行う必要があります。具体的には、`-init` オプションに設定されたコマンド列は、「`-graphic`」(p.202) で解説する `-graphic` オプションに設定されたコマンドによって、プリンタドライバがビットイメージ印字をするための下ごしらえをします。したがって、プリンタのコマンドのなかから初期化を行うもの、および `-graphic` オプションで設定するコマンドの使用上、設定が必要と思われるコマンドをチェックします。

BJ-10 の場合、プリンタドライバが行うビットイメージ印字に影響を及ぼするようなコマンドとしては改行制御のコマンドしかありませんでした。

16) BEEPs 氏がパソコン通信のメンバーの協力を得てつくり上げた、X68030 用の 68040 アクセラレータのことです。X68030 の 2~3 倍の高速化を実現しています。

17) 一般のグラフィックのハードコピープログラムでは、この境目を目立たなくするように、行の境を 1, 2 ドット重ねて出力するなどの処理を行うのですが、これはまた別のお話です。

ここで、なぜ改行コマンドがビットイメージ印字に影響を及ぼすかを理解するためには、プリンタドライバが「絵」の「ハードコピー」をしているというたとえを思い出してください。一般的な絵などをハードコピーする際にも、絵を隙間なく出力するために、これから出力する行の最上端が直前の行の最下端に密着するように改行されるはず<sup>17)</sup>。プリンタドライバも同じように、パッファという「キャンバス」に描かれた、原稿という「絵」を「ハードコピー」して出力するので、行間に隙間が空かないように印字するためには、改行量をきっちり 1 行の高さにしなければなりません。

そこで、プリンタのマニュアルから、初期化および改行制御に関するコマンドを探してみましょう。

BJ-10 の場合、マニュアルを見ると、

- ESC @                      プリンタの初期化 [IV-49]
- ESC 3                      改行量を  $n/180$  インチに設定 [IV-14]
- ESC A                      改行量を  $n/60$  インチに設定 [IV-14]

改行量の設定コマンドとしては、“ESC 3”と“ESC A”がありますが、どちらを使用してもかまいません。引数  $n$  の値が変わってくるだけで、同じ結果をもたらします。

さて、360dpi の BJ-10 は 1 行の高さが 48 ドットのビットイメージ印字を行いますから、1 インチあたり 360 ドットあるうちの 48 ドットが 1 行の高さにあたります。実際には  $1/360 \times 48$  インチということになります。“ESC A  $n$ ”は、改行量を  $n/60$  インチに設定するコマンドでした。改行量は  $1/360 \times 48$  インチですから、 $1/360 \times 48 \times 60 = 8$  が  $n$  にあたります。

では、実際に `-init` の文字列をつくってみましょう。

BJ-10 において `-init` には、“ESC @”と“ESC A 8”を設定します。ここで、ESC とはエスケープシーケンスのことで、プリンタドライバのリファレンスにあるように、オプションの文字列引数中で使用できるコードのなかに含まれています。

使用できるコードの記法は以下のようになっています。

- プリンタドライバのオプション引数で使用可能なコード

|                    |                          |
|--------------------|--------------------------|
| <code>\x??</code>  | 16 進数 ?? 指定。必ず 2 桁分記述する。 |
| <code>\????</code> | 8 進数 ??? 指定。必ず 3 桁分記述する。 |
| <code>\b</code>    | ベル (BEL, 0x07)           |
| <code>\t</code>    | 水平タブ (HT, 0x09)          |
| <code>\n</code>    | 改行 (LF, 0x0a)            |
| <code>\v</code>    | 垂直タブ (VT, 0x0b)          |
| <code>\f</code>    | 改ページ (FF, 0x0c)          |
| <code>\r</code>    | 復帰 (CR, 0x0d)            |
| <code>\d</code>    | deselect (DC3, 0x13)     |
| <code>\s</code>    | サブコード (SUB, 0x1a)        |
| <code>\e</code>    | エスケープコード (ESC, 0x1b)     |



|    |                   |
|----|-------------------|
| \\ | \\コード             |
| %% | % コード             |
| %0 | ヌルコード (NUL, 0x00) |

ヌルコードは \000, \x00 等では指定できません。

したがって、プリンタ初期化のコマンド ESC @ は \e@ で、ESC A 8 は \eA\x08 で記述できることになります。ESC A のように定数引数 (この場合は 8) を持つコマンドについては、\x?? で 2 桁の 16 進数を与えるか、\??? で 3 桁の 8 進数を与えることに注意してください。このとき、指定された桁数を満たさない場合には、\x08 のように、不足している桁の数だけ 0 を付加します。

List 5-25 ● プリンタの初期設定

```
13: -init=\e@\eA\x08
14: -init=\e@\eA\010
15: -init=\e@\e3\x18
16: -init=\e@\e3\x30
```

上に示した -init= のコマンド列が、どれも同じ結果になるのだということを理解してください。

◆ -CRLF

-CRLF には、プリンタ改行のコード列を指定します。通常、ここには復帰改行 CR LF (Carriage Return & Line Feed) を指定します。これらのコードは p.200 で示したプリンタドライバのオプションで使用可能なコードのなかに含まれていましたね。

List 5-26 ● 改行コードの設定

```
14: -CRLF=\r\n
```

◆ -extraCRLF

-extraCRLF には、特殊改行コードの設定をします。ここでいう「特殊改行コード」とは、空行時に -CRLF に設定したコードの直前に追加するコード列です。つまり、-extraCRLF を設定した場合、出力行が空行ならば、改行コードとして -extraCRLF および -CRLF に設定されたコマンド列がプリンタに送られることになります。

特殊改行コードの設定は、よほどのことがないかぎり必要ないでしょう。BJ-10 でも設定しないことにします。

List 5-27 ● 特殊改行コードの設定

```
15: -extraCRLF=
```

-extraCRLF の設定が必要な場合としては、1 行分のイメージ出力を終了した際に自動的に改行する仕様になっているプリンタがあります。このようなプリンタを使用する際には、次のように設定することで対応します。

**List 5 - 28 ● 特殊改行コードの設定例**

```
14: -CRLF=\r
15: -extraCRLF=\n
```

この場合は、イメージデータがある行はプリンタ側で改行処理をしてくれますから -CRLF で復帰コードだけを送り、空行時にはプリンタ側の自動改行が行われませんから、-extraCRLF で改行コードを送ります。

◆ -FF

18) シャープ純正のオートシートフィーダが存在しないというだけで、実際のところ、CZ-8PC4 にはエプソン AP-800/550 用のオートカットシートフィーダ、APCSF が使用できます。その際には、CZ-8PC4 のマニュアルには「絶対動かさないで」と書かれているディスプレイの 7 番を ON にします。

-FF には、改ページの際にプリンタに送るコード列を設定します。CZ-8PC4 のように、オートシートフィーダをつけていないプリンタ<sup>18)</sup>の場合には、紙 1 枚の印字が終わった時点で改頁コマンド (FF)、およびプリンタの deselect (データを受信できない状態にする) コマンド (DC3) を送ります。

これらのコードは p.200 で示したプリンタドライバのオプションで使用可能なコードのなかに含まれていましたね。

**List 5 - 29 ● DC3 がある場合の改頁**

```
16: -FF=\f\d
```

その他の場合には、FF のみで OK です。

**List 5 - 30 ● DC3 がない場合の改頁**

```
16: -FF=\f
```

BJ-10 の場合、たとえオートシートフィーダをつけていない場合でも、DC3 コマンド自体がありませんから、必然的にこちらの設定になります。

◆ -graphic

リファレンスによれば、1 ページのはじまりごとに、ここで指定されたコード列を出力してからビットイメージデータの出力が始まる、とのこと。したがって、-init で初期設定されたプリンタに、このオプションで「今からイメージデータを送るから、よろしく頼みます」という命令をすればよいのです。

BJ-10 のマニュアルによれば、イメージ印字に関する命令のうち、48 ドットモードを指示するものは以下のものがあります。

- “ESC | \*”      48 ドットイメージモードの選択 [IV-47]

詳しく見てみると、このコマンドは “ESC | \*  $n_1n_2$  DATA” というフォーマットで、出力すべきビットイメージデータ DATA の総数を  $n_1 + n_2 \times 256$  で与えることになっています。つまり、このとき  $n_1$  がイメージデータの総数を示す下位バイト、 $n_2$  が上位バイトとなっています。このように、データの表記をする際に、そのコードの上位バイトと下位バイトを逆に指定するやり方を INTEL オーダー (little endian) といいます。また、ここでは使用しませんが、上位バイトと下位バイトをそのまま指定するやり方を MOTOROLA オーダー (big endian) といいます。

これで、-graphic に設定するコード列は、ESC | \*  $n_1n_2$  の部分であることがわかりました。では、肝心の  $n_1n_2$  はどう設定するのでしょうか？ リファレンスを調べてみると、プリンタドライバには不定の引数を指定するためのコード列が用意されていました。

#### ○ 不定の引数を指定するためのコード

%?d ? には、“1” から “5” までの ASCII コードを指定します。返り値は、? 桁の十進数 ASCII コードで表されたドット数です。

%?D -graphic でしか使用できません。? には、“1” から “5” までの ASCII コードを指定します。返り値は、? 桁の十進数 ASCII コードで表されたバイト数です。

%?m ? には、“1” または “2” を指定します。返り値は、MOTOROLA オーダーで ? バイトのバイナリデータ (? 桁の 16 進数) で表されたドット数です。

%?M -graphic でしか使用できません。? には、“1” または “2” を指定します。返り値は、MOTOROLA オーダーで ? バイトのバイナリデータ (? 桁の 16 進数) で表されたバイト数です。

%?i ? には、“1” または “2” を指定します。返り値は、INTEL オーダーで ? バイトのバイナリデータ (? 桁の 16 進数) で表されたドット数です。

%?I -graphic でしか使用できません。? には、“1” または “2” を指定します。返り値は、INTEL オーダーで ? バイトのバイナリデータ (? 桁の 16 進数) で表されたバイト数です。

このように、これらのコードは「ドット数」または「バイト数」を返すのですが、返った値の意味は、コードを使用したオプションによって異なってきます。-graphic の場合には、“あとに続くビットイメージデータの総数”を「ドット数」または「バイト数」で返すことになります。

さて、先に述べたように、 $n_1n_2$  は、上位バイトと下位バイトが逆転した 2 バイトの INTEL オーダーで、後続するビットイメージデータの総バイト数を表しますから、 $n_1n_2$  は %2I で与えることができます。

List 5-31 • BJ-10 のイメージモード設定

```
17: -graphic=\e|*%2I
```



参考までに、CZ-8PC4 の場合を示しておくと、CZ-8PC4 では “ESC M  $n_1n_2$  DATA” が 48 ドットのビットイメージ印字を行うコマンドです。 $n_1n_2$  には、あとに続くビットイメージデータ (DATA) のドット数を MOTOROLA オーダーの 2 バイトで記述することになっています。そこで、CZ-8PC4 では以下のよう  
に設定することになります。

List 5-32 ● CZ-8PC4 のイメージモード設定

```
17: -graphic=\eM%2m
```

### 5.3.4 テスト出力してみる

さあ、ここまで設定したら、ためしに出力してみましょう。

第 5.3.1 項「プリンタドライバのオプション」(p.186)で示したオプションのうち、`-start`、`-relative`、`-repeat` はまだ解説をしていますが、これらは必ずしも設定する必要がないオプションです。リファレンスで説明しているように、これらのオプションは「プリンタに送る無駄なデータをできるだけ省くための仕組み」であって、出力スピードが遅くてもいいなら、あえて頭を悩ましてまで設定する必要はないのです。とはいっても、コンフィギュレーションファイルに記述しなくてもいいということではなく、引数を空 (何も指定しないこと。以下、同様の意味で使用する) にして設定することで、それぞれのオプションにデフォルトで設定されているコード列をキャンセルしなければなりません。

List 5-33 ● テスト版コンフィギュレーションファイル

```
1: -remark= <<<<< Canon BJ-10v Configuration File >>>>>
2: -remark= テスト版
3: -remark=
4: -remark= << ドライバ制御系 >>
5: -dpi=360 -remark= 解像度は 360dpi を指定
6: -width=2880 -remark= バッファはレターサイズで確保するが、
7: -height=3960 -remark= A4 サイズの用紙に出力することを想定
8: -remark=
9: -remark= << プリンタ制御系 >>
10: -MSBisUpper -remark= MSB は上方にくる
11: -pinBytes=6 -remark= × 8 = 4 8 ドット
12: -prBufSize=256 -remark= スプールバッファは 256K バイト
13: -init=\e@\eA\x08 -remark= 初期化 (ESC @) と改行幅 (ESC A) を設定
14: -CRLF=\r\n -remark= 改行は CR LF
15: -extraCRLF= -remark= 特殊な改行処理はしない
16: -FF=\f -remark= 改頁は FF
17: -graphic=\eI%2I -remark= 48 ドットイメージ印字
18: -start= -remark= とりあえず
19: -relative= -remark= 出力してみる。
20: -repeat= -remark= うまくいくかな ?
21: -remark=
22: -remark= remark オプション内のスペースはカナスペースです。
```

List 5-33 をコンフィギュレーションファイルとして、ひとまずテスト出力してみます。ソースファイルには、デバイスドライバのアーカイブに添付されてい

るプリンタドライバのマニュアル `print.tex`<sup>19)</sup> の処理結果 `print.dvi` を使用してみました。なお、`print.tex` はデフォルトのレターサイズではなく、A4 サイズの原稿ですから、正確な位置に出力するために、確保するバッファのサイズとそれに対する書き込み位置を調整します<sup>20)</sup>。

```
A> print.x -height=4208 -width=2880 -xOffset=312 -yOffset=240
print.dvi
```

出力結果を見ると、行間が異常に間延びしてしまい、A4 サイズの原稿が A4 用紙に入りません。

ここで、出力すべきデータ自体はすべて出力されていますから、`-prBufSize` によって設定したスプールバッファがあふれた<sup>21)</sup> という症状ではないことがわかります。

BJ-10 のマニュアルを見ると、ディスプレイの 7 番に CR コマンドを CR LF 相当にする機能が割り振られていましたので、これを確認してみましたが、このディスプレイは OFF になっていました。

そこで、縦に間延びするのだから改行コードが悪いのだろうと推測して、`-CRLF=\n` に変えてみます。

```
A> print.x -height=4208 -width=2880 -xOffset=312 -yOffset=240
-CRLF=\n print.dvi
```

今度はちゃんと A4 原稿が A4 用紙に収まりました。

実は、エプソン系のプリンタの一部でも同様な縦スペースの間延び現象が発生します。この抜本的な解決策は、プリンタ側の 14 番ケーブルを絶縁する (エプソン系のプリンタの場合には絶縁のうえ 13 番につなぎなおす) ことです。この処理をすれば、改行コード列として CR LF を送ることができるようになります。なお、この処理をすることによって、他のプリントアウト動作に悪影響が生じることはないようです。

List 5-34 ● とりあえず動くコンフィギュレーションファイル

```
1: -remark= <<<<< Canon BJ-10v Configuration File >>>>>
2: -remark= 「とりあえず動くぞ」版
3: -remark=
4: -remark= << ドライバ制御系 >>
5: -dpi=360 -remark= 解像度は 360dpi を指定。
6: -width=2880 -remark= バッファはレターサイズで確保するが、
7: -height=3960 -remark= A4 サイズの用紙に出力することを想定
8: -remark=
9: -remark= << プリンタ制御系 >>
10: -MSBisUpper -remark= MSB は上方にくる
11: -pinBytes=6 -remark= × 8 = 4 8 ドット
12: -prBufSize=256 -remark= スプールバッファは 256K バイト
13: -init=\e@\eA\x08 -remark= 初期化 (ESC @) と改行幅 (ESC A) を設定
14: -CRLF=\n -remark= 改行は LF だけ。
15: -remark= ケーブル 14 番のカットはしていない。
```

19) 本書の添付ディスクが構築する `TEX` システムの場合、`%TEXHOME%\drivers\doc` の配下にあります。

20) 「`-xOffset`、`-yOffset`」(p.193) を参照。

21) 「`-prBufSize`」(p.198) を参照してください。なお、このとき使用した本体は X68000 EXPERT II(無改造)で、実際には `-prBufSize=8` でも動作することを確認しています。それ以下の値については、試していませんので不明です。

```

16: -extraCRLF= -remark= 特殊な改行処理はしない
17: -FF=\f -remark= 改頁は FF
18: -graphic=\e|*%2I -remark= 48 ドットイメージ印字
19: -start= -remark= とりあえず
20: -relative= -remark= 動くようにはなった
21: -repeat= -remark= しかし、おそ〜い
22: -remark=
23: -remark= remark オプション内のスペースはカナスペースです。

```

これで、ともかく動くコンフィギュレーションファイルができあがりました。しかし、お世辞にも快速とはいいがたい出力スピードですから、もう少しこのコンフィギュレーションファイルをいじってみましょう。

### 5.3.5 高速化を目指して

前項 (p.204) で説明したように、`-start`、`-relative`、`-repeat` は「プリンタに送る無駄なデータをできるだけ省くための仕組み」ですから、これらに有効な指定をしてやることで格段に印字速度が上がります。できるだけ以下のオプションを設定してみましょう。

#### ◆ `-start`

このオプションには、ドット単位で印字開始位置を指定するためのコード列を設定します。`-start` の引数が空の場合、出力する行の横方向の空白はビットイメージで「出力」されますが、`-start` オプションを的確に設定した場合には、この空白をプリンタの印字開始位置を調整することで実現するようになります。こうすることで、プリンタへ転送するデータの量が減りますから、その分、出力が高速になります。

BJ-10 のマニュアルを探してみると、ドット単位での印字開始位置指定の機能を持ったコマンドがありました。

○ “ESC \$”                      印字ヘッドの絶対位置指定 [IV-13]

このコマンドは、“ESC \$  $n_1n_2$ ” という書式で、左マージン位置から次の印字開始位置を  $n_2 \times 256 + n_1$  ドット移します。 $n_1n_2$  には不定のドット数を示すデータを設定しなければなりません。これを設定するために、「`-graphic`」(p.202) の `-graphic` の設定について解説する過程 (p.203) で説明した不定の引数を指定するためのコードを、ここでも使用します。

`-start` オプションは、ドット単位の印字開始位置の指定コードを設定しますから、本オプションでは不定の引数を指定するためのコードのうち、「ドット数」を返すものが使用対象となります。この場合には、これらの不定の引数を指定するコードで返った「ドット数」は、「特定の位置から次に印字を開始する位置までに移動する「ドット数」に該当することになります。



さて、先に述べたように“ESC \$”の引数  $n_1n_2$  は、左マージン位置から次に印字を開始する位置までに移動する「ドット数」に該当します。その移動するドット数は  $n_2 \times 256 + n_1$  です。つまり、このとき  $n_1$  が移動するドット数を表す下位バイト、 $n_2$  が上位バイトとなっており、 $n_1n_2$  は2バイトの INTEL オーダーで指定することがわかります。そして、不定の引数を指定するためのコード列から該当するものを探すと、%2i が該当します。

List 5-35 ● BJ-10 版 印字位置の絶対設定

```
19: -start=\e$%2i
```

参考までに CZ-8PC4 の場合も考えてみましょう。CZ-8PC4 の印字ヘッドの絶対位置指定を行うのは、以下のコマンドです ([ ] 内は CZ-8PC4 のマニュアルのページ数)。

#### ○ ESC POS 印字ヘッドの絶対位置指定 [98]

このコマンドは、“ESC POS  $n_1n_2n_3n_4$ ”の書式で与えられ、左マージンから180dpi 換算で  $n_1 \times 1000 + n_2 \times 100 + n_3 \times 10 + n_4$  ドット移動する命令です。

ここで問題になるのが、「180dpi 換算」での移動ドット数を与えてやらなければならないということです。CZ-8PC4 は 360dpi ですから、360dpi での絶対印字開始位置の指定ができるなら、先ほどから再三使用している不定の引数指定のためのコード列のひとつである %4d をそのまま使用すれば、絶対移動すべきドット数を設定することができます。しかし、CZ-8PC4 の絶対印字開始位置の指定のコマンドは、360dpi ではなく 180dpi 換算でしか機能しません。ゆえに、もし  $n_1n_2n_3n_4$  を不定引数を指定するためのコード %4d で表してしまったら、実際に (360dpi で) 絶対移動すべきドット数の2倍、ヘッドが移動してしまいます。

ここで登場するのが「割り算ファクタ」という概念です。

たとえば、360dpi で15ドットだけ印字ヘッドを移動させたい場合を考えてみましょう。dpi は解像度を表しますから、これが倍になれば1ドットの幅は半分になります。いいかえれば、dpi が倍になれば、同じ幅を表現するために倍のドットが必要になるわけです。したがって、次のようにいえます。

- (1) 180dpi で7ドットだけ印字ヘッドを移動したとすると、それは360dpi で印字ヘッドが14ドット移動したことになる。
- (2) そして、この状態からあと360dpi で1ドット、印字ヘッドを移動させれば、360dpi で印字ヘッドが15ドット移動したことになる。

この機能を果たすのが「割り算ファクタ」なのです。360dpi の CZ-8PC4 において任意のドット数  $x$  が表現する幅は、180dpi で  $x \times 180/360 = x/2$  ドットで表現されます。ここで  $x$  を割った2が、割り算ファクタと呼ばれるものです。

もし、本来の解像度が360dpi であるものの、絶対印字位置指定は120dpi でのみ機能するプリンタの場合であれば、360dpi のドット数  $x$  が表現する幅は、120dpi では  $x \times 120/360 = x/3$  ドットで表現されますから、このときの割り算ファクタは3になります。

CZ-8PC4 に戻りましょう。まず、360dpi のとき  $x$  ドットヘッドを移動するものとします。次に、このドット数  $x$  を割り算ファクタ 2 で割り、その商にあたるドット数を %4d のフォーマットで返します (この返り値を  $y$  とする)。そして “ESC POS  $y$ ” で 180dpi 換算のヘッド移動を行います。この部分が「360dpi で 15 ドットだけヘッドを移動する」という例のなかの (1) にあたる過程です。

その後、 $x$  を割り算ファクタである 2 で割ったときの余りがあるなら、その余りの分だけ空白のビットイメージを送ってやります。この過程が (2) にあたります。これで、360dpi のときに  $x$  で得られるドット数だけヘッドを移動させることができたことになります。

以上のような機能を実現する「割り算ファクタ」は、次のような書式で設定します。

#### ○ 割り算ファクタの書式

%?/\*d 割り算ファクタ \* を指定することで、本来の dpi ÷ 割り算ファクタ で得られる dpi において移動すべきドット数を、%?d のフォーマットで返します。

%?/\*m 割り算ファクタ \* を指定することで、本来の dpi ÷ 割り算ファクタ で得られる dpi において移動すべきドット数を、%?m のフォーマットで返します。

%?/\*i 割り算ファクタ \* を指定することで、本来の dpi ÷ 割り算ファクタ で得られる dpi において移動すべきドット数を、%?i のフォーマットで返します。

したがって、CZ-8PC4 の絶対印字開始位置指定コマンド “ESC POS (\e\x10)” の引数  $n_1n_2n_3n_4$  は %4/2d で与えられることになります。そこで、以下のよう設定します。

List 5-36 ● CZ-8PC4 版 印字位置の絶対位置設定

```
19: -start=\e\x10%4/2d
```

#### ◆ -relative

-start が、左マージン位置という特定位置からドット単位で次の印字開始位置を指定するコマンドを設定するオプションであったのに対して、-relative は、現在のヘッドの位置から次の印字開始位置をドット単位で指定するコマンドを設定するオプションです。設定の目的は -start と同じです。

BJ-10 のマニュアルには、以下のコマンドが示されています。

#### ○ “ESC \” 印字ヘッドの相対位置指定 [IV-13]

このコマンドは、“ESC \  $n_1n_2$ ” の書式で、現在のヘッドの位置から 180dpi に換算して  $n_2 \times 256 + n_1$  ドットだけ印字位置を移します。

「180dpi に換算した」移動ドット数を与えてやるために、「-start」における CZ-8PC4 の設定例を解説する過程 (p.208) で登場した「割り算ファクタ」を使用します。 $n_2 \times 256 + n_1$  に相当するのは  $\%2/2i$  であることがわかります。

List 5-37 ● 印字位置の相対設定

```
20: -relative=\e\\%2/2i
```

ところで、リファレンスには、割り算ファクタの設定例として BJ-130J の場合が紹介されています。よくわからない方のために、一体何をやっているのか、もう少し説明してみましょう。

List 5-38 ● BJ-130J の -relative 設定例

```
20: -relative=\eU%4/2d%0%0%0
```

“ESC U” は、“ESC U  $n_1 n_2 n_3 n_4 d_1 d_2 d_3$ ” の書式で使用します。そして、180dpi モードにおいて縦 1 ドット列にあたるビットイメージデータ  $d_1 d_2 d_3$  を  $n_1 n_2 n_3 n_4$  回繰り返して出力する機能を果たします。

この設定例では、繰り返すべき縦 1 ドット列に相当するイメージデータ  $d_1 d_2 d_3$  として 3 つの NULL を送っています。ここで、BJ-130J は -MSBisUpper を指定すべきプリンタです。また、BJ-130J 自体の解像度は 360dpi ですが、グラフィックリピート機能は 180dpi の解像度でしか動作しません。すなわち、3Bytes=24bits のビットイメージデータ  $d_1 d_2 d_3$  は、24 ドット、つまり 180dpi の縦 1 ドット列に相当するのです。つまり、繰り返されるイメージデータ “%0%0%0” は 180dpi での空の縦 1 ドット列ということになります。既述のとおり、本来 BJ-130J は 360dpi ですから、180dpi で空の縦 1 ドットを出力した場合、360dpi で空の縦 2 ドット列を出力したと同様の結果を生じます。これは同時に、360dpi で 2 ドット印字ヘッドを相対移動したと同様の結果になるのです。

ここで、360dpi で  $x$  ドット印字ヘッドを相対移動したいとしましょう。-start で行った解説の繰り返しになりますが、360dpi での  $x$  ドットが表現する幅は、180dpi では  $x \times 180/360 = x/2$  ドットで表現されます。また、360dpi で  $x$  ドット印字ヘッドを相対移動するということは、現在のヘッドの位置から 360dpi で空の縦 1 ドット列を  $x$  回繰り返したのと同じことです。したがって、180dpi では  $x \times 180/360 = x/2$  回、縦の 1 ドット列を繰り返すことで、同様の結果を生むことができます。これが、BJ-130J でグラフィックリピート機能を使用した -relative オプションの設定例、“\eU%4/2d%0%0%0” が行っていることなのです。

リファレンスには、今、解説したグラフィックリピート機能も使用した場合の例のすぐ次に、キャラクターリピート機能を使用した例も載っています。「設定によって行うこと」は、グラフィックリピートによる相対印字位置指定と同様の発想によります。相対印字位置指定やグラフィックリピートの機能を持つコマンドはないものの、キャラクターリピートの機能はあるというプリンタを使用している人は、これを参考に設定してみてください。



◆ `-repeat`

`-repeat` には、グラフィックリピートを指定するコード列を設定します。しかし、BJ-10 にはグラフィックをリピートするコマンドはありませんから、このオプションは設定のしようがありません。ただし、`-repeat` にはデフォルトでコマンドが登録されていますから、このデフォルトのコマンドをプリンタに送らないように、空を設定します。

List 5-39 ● 印字位置の相対設定

```
21: -repeat=
```

## 5.3.6 再度のテスト出力を試みる

これで高速化を目指した設定ができました。うまく高速化できているか試してみます。

ここでは、コンフィギュレーションファイルに先ほど作成した「とりあえず動くぞ」版 (List 5-34) を使用し、高速化関連のオプションは設定を変更しやすいようにコマンドラインで指定することにします。

また、テスト出力をするためのソースには、第 5.3.4 項「テスト出力してみる」(p.204) のテスト出力で使った `print.dvi` を使用することにします。

```
A> print.x -start=\e$%2i -relative=\e\\%2/2i -repeat= -height=4208
-width=2880 -xOffset=312 -yOffset=240 print.dvi
```

この設定で出力したところ、どうも印字が乱れます。一気に 1 行印字すればよいところを、印字ヘッドがさかんに左右に往き来しています。どうやら横方向に空白がある部分で印字を行ってしまうようで、このとき、横方向のスペースがまったく出力されていません。

そこで、今度は次の設定で出力してみます。

```
A> print.x -start= -relative=\e\\%2/2i -repeat= -height=4208
-width=2880 -xOffset=312 -yOffset=240 print.dvi
```

今度はちゃんと仕上がりました。高速化関連のオプションを何も設定していないときに比べて、格段に速くなっています。ここで使用しているサンプル、`print.dvi` の 1 ページめを出力してみたところ、高速化関連のオプションを設定している場合と設定していない場合とでは、1 分ほどの違いが<sup>22)</sup> 生じます。

`-start` オプションの設定が悪かったのでしょうか。念のため、今度は以下の指定でも試してみます。

22) このとき使用した本体は X68000 EXPERT II (無改造) です。

```
A> print.x -start=\e$%2i -relative= -repeat= -height=4208
-width=2880 -xOffset=312 -yOffset=240 print.dvi
```

この指定で印字を始めると、やはり出力が乱れました。

そこで、何が問題になっているのか、プリンタに送られている実際のデータを「見てみる」ことにします。プリンタドライバには、`-dump` オプションがあり、プリンタに出力すべきバイナリデータをファイルに出力することができるのです。

```
A> print.x -dump=StartSetting.dat -start=\e$%2i -relative=\e\\%2/2i
-repeat= -height=4208 -width=2880 -xOffset=312 -yOffset=240 print.dvi
```

`-start` を設定した場合のデータを `StartSetting.dat` にダンプ出力して、そのファイルを調べます。

```
A> dump StartSetting.dat
00000000 1B 40 1B 41 08 0D 0A 0D 0A 0D 0A 0D 0A 0D 0A 0D .@.A.....
00000010 0A 0D 0A 0D 0A 0D 0A 0D 0A 0D 0A 1B 24 A0 03 1B$..
00000020 7C 2A 4E 03 00 03 FE 00 00 00 0F FF FE 00 00 00 |*N.....
00000030 0F FF 00 00 00 00 0F F0 00 00 00 00 0F C0 00 00
00000040 00 00 0F 80 00 00 00 00 0F 00 00 00 00 00 0E 00
00000050 00 00 00 00 0E 00 00 00 00 00 0E 00 00 00 00 00
:
:
```

横方向の空白の制御には、すべて絶対印字位置指定コード“ESC \$ (\e\$)”が使用されているようです。

そこで“ESC \$”、つまり“1B 24”に続いて出力されている絶対移動のドット数を見てみますと、“A0 03”となっています。`\e$`の引数 $n_1n_2$ は、 $n_1+n_2 \times 256$ で表されるドット数でしたから、この場合、 $160 + 3 \times 256 = 928$ ドットの絶対移動を行おうとしていることがわかります。

もう一度 BJ-10 のマニュアルを調べると、この絶対印字位置指定のできる範囲は、 $0 \leq n_1 + n_2 \times 256 \leq 479^{23)}$  でしかありません。つまり、`print.dvi` の出力時に移動しようとしているドット数 928 は、いうまでもなく、絶対印字位置の指定が可能な範囲の上限を越えてしまっています。キヤノンの「BJ コールセンター」に電話をかけて確認したところ、次のような回答を得ました。

23)BJ-15 での上限は 480 のようです。

- (1) BJ 系の 80 桁プリンタにおける絶対印字位置指定コマンドで設定できる移動ドット数の上限は 480 ドット、というのが仕様である。
- (2) もし、この上限を越える値を使用した場合には、「ちょっとやってみたことがないですけど、無効になるんじゃないでしょうか」。

BJ-10 における絶対印字位置指定の上限である 479 ドットは、実際の長さによれば 1.3in (33.8mm) でしかありませんから、プリンタドライバにおいて印字位置の指定に使用する場合には使用できません。

ゆえに、`-start` には空を設定します。

さて、今回のサンプルの印字では効果的に作用した相対印字位置の指定 `-relative` ですが、BJ-10 では、相対印字位置の指定コマンドにおける移動ドット数にも制約があります。“ESC \” の引数  $n_1n_2$  は、 $-1440 \leq n_1 + n_2 \times 256 \leq 1440$  でなければならないのです。上限となる 1440 ドットは、実際の長さによれば 4in (101.6mm) ですから、これは BJ-10 の最大印字幅 203.2mm (マニュアル [VIII-3]) のちょうど半分の長さにあたります。したがって、もし A4 のページの右隅だけに出力すべき情報があるようなソースを印字する場合には、`-start` を設定した場合と同様に印字が乱れる場合があります。しかし、通常のソースファイルの出力においては問題ないと判断し、`-relative=\e\\%2/2i` の設定を行うことにします。

### 5.3.7 コンフィギュレーションファイルの完成

さあ、これで BJ-10 のコンフィギュレーションファイルができました。

List 5-40 ● BJ-10v 用カスタムコンフィギュレーションファイル

```

1: -remark= <<<<< Canon BJ-10v Configuration File >>>>>
2: -remark=
3: -remark= << ドライバ制御系 >>
4: -dpi=360 -remark= 解像度は 360dpi を指定。
5: -width=2880 -remark= バッファはレターサイズで確保するが、
6: -height=3960 -remark= A4 サイズの用紙に出力することを想定
7: -remark=
8: -remark= << プリンタ制御系 >>
9: -MSBisUpper -remark= MSB は上方にくる
10: -pinBytes=6 -remark= × 8 = 48 ドット
11: -prBufSize=256 -remark= スプールバッファは 256K バイト
12: -init=\e@\eA\x08 -remark= 初期化 (ESC @) と改行幅 (ESC A) を設定
13: -CRLF=\n -remark= 改行は LF だけ。
14: -remark= ケーブル 14 番のカットはしていない。
15: -FF=\f -remark= 改頁は FF
16: -graphic=\e|*%2I -remark= 48 ドットイメージ印字
17: -start= -remark= 指定有効範囲が小さすぎるので使用できず
18: -relative=\e\\%2/2i -remark= 指定範囲を超える場合には空にするべし
19: -repeat= -remark= コマンドがない
20: -remark=
21: -remark= remark オプション内のスペースはカナスペースです。

```

かなりていねいに解説してきたつもりなので、おそらく未知のプリンタに対してコンフィギュレーションファイルを書く際の参考にさせていただけるのではないのでしょうか。コンフィギュレーションファイルを作成するための唯一の助言があるとするれば、それは「マニュアルをよく読むこと」につきます。もちろん、ここでいう「マニュアル」とは、プリンタのそれであり、プリンタドライバのそれでもあります。これらを読み、あとは試行錯誤を繰り返せば、とりあえず動くコン



フィギュレーションファイルを作り出すことはさほど困難ではないはずです。

皆さんも、それぞれの「マニュアル」を参考に、自分自身でのコンフィギュレーションファイルの作成にチャレンジしてみてください。そして、コンフィギュレーションファイルができあがったならば、同じプリンタを使用している、あるいはこれから使用しようとしている人々のためにも、それを発表してくださることを期待します。

## 5.4 ..... METAFONT

本節では、X680x0 版の METAFONT を使うにあたっての環境の整備や起動方法について説明します。この章は、あくまで METAFONT の本体を動かす方法について触れているだけで、METAFONT のソースファイルの記述方法等については説明していませんので、ご注意ください。

### 5.4.1 METAFONT の概要

METAFONT は、T<sub>E</sub>X で印刷や閲覧をするときに使う英文字や数字のフォントを作成するためのプログラムです。T<sub>E</sub>X に比べると使う頻度が低いので、あまり解説書などで大きく紹介されることはありませんが、T<sub>E</sub>X の“美しい文字”を支えている重要なプログラムであることは間違いありません。METAFONT の操作方は T<sub>E</sub>X に非常に似ています。T<sub>E</sub>X を扱えるようになっていれば、METAFONT を操作するのは難しいことはありません。

#### ◆ METAFONT の重要な注意事項

X680x0 版 METAFONT については、重要な注意事項があります。この注意事項は、UNIX と Human68k との違いに起因するものです。種々のフリーソフトウェアによって UNIX の環境に近づいた Human68k では大きな問題にはなりませんが、これらのフリーソフトウェアがインストールされていない場合には、METAFONT を動かすことができませんので、ここで説明していることを十分頭に入れておいてください。

X680x0 版 METAFONT は、UNIX 上で使われている METAFONT をそのまま移植したものです。X680x0 版 METAFONT は、Human68k でのオリジナル環境であるファイル認識文字について、ファイル名 8 文字 + 拡張子 3 文字<sup>1)</sup> といった制限があることを知りません。このため、X680x0 版 METAFONT は、Human68k が通常では許さない名前のファイルを生成しようとしてエラーを起こしたり、本来、UNIX では別々のファイルとして扱われるファイルを混同してエラーを発生させたりします。ファイル名の制限に関連した内容が第 1.2.7 項「T<sub>E</sub>X のための X680x0 のセットアップ」(p.18)、および第 1.2.4 項「より高度な環境構築」

1) Human68k では、ファイル名全体で 21 文字まで使えますが、21 文字すべてを認識しているわけではありません。

(p.14)、第 1.2.8 項「 $\text{\TeX}$  が使うファイルの拡張子」(p.19)に記述されていますので、参考にしてください。

MS-DOS 版の METAFONT では、このような貧弱なファイル環境でも動くように手を加えた形で移植されています。一方、X680x0 の世界では、川本琢二氏がこの制限を事実上 UNIX での環境と同じ程度にまで解除するフリーソフトウェア `TwentyOne.x` を製作しています。METAFONT は、もともと、そのようなファイル名の制限がない UNIX 上で使われていたために、X680x0 上で UNIX の世界で蓄積されてきた資産を利用する場合には大幅な変更をしなければなりませんでした。しかし、`TwentyOne.x` の登場によってこの制限が解除されたおかげで、多くの UNIX 上のファイルを UNIX から X680x0 にそのまま転送するだけで動かすことができるようになりました。X680x0 版 METAFONT は、X680x0 でこのような環境が使えることを考え、あえて MS-DOS 版のようなファイル名 8 文字 + 拡張子 3 文字しか使えない貧弱な環境にあわせることはせずに、そのまま移植しておきました。このため、X680x0 版 METAFONT はオリジナルの Human68k の環境では動作しないソフトウェアになっているので、十分に注意してください。

このような注意は  $\text{\TeX}$  の場合でも同じですが、本書をお届けするにあたって今まで使われてきた X680x0 での METAFONT を Ver.1.7 から Ver.2.7 に一気にバージョンアップしました<sup>2)</sup>。

Ver.1.7 では、MS-DOS と同様な Human68k の環境にあわせ、移植の際にファイル名の変更等を行いました。しかし、Ver.2.7 ではこのような変更を行いませんでした。そのため、今までの METAFONT を使ってこられた人が混乱することが予想されましたので、ここに注意書きを書いております。

### ◆ METAFONT の処理の流れ

METAFONT の処理の流れ自体は、 $\text{\TeX}$  に非常に類似しています。しかし、 $\text{\TeX}$  のように一般のユーザがたびたび操作しなければならないようなプログラムではありません。METAFONT は、 $\text{\TeX}$  で使う英文字や数字のフォント、“METAFONT”に使われているような特殊なフォント等を作成するためのプログラムなのです。

X680x0 では、METAFONT はたいてい第 3.3 節「プレビューア — 画面での閲覧」(p.56)で説明される `preveiw.x` あるいは、第 3.4 節「プリンタドライバ — プリントアウト」(p.64)で説明される `print.x` が「フォントが見つからない」といったエラーを発生した場合<sup>3)</sup>に使います。

実際に 118dpi の画面表示用フォントファイル `cmtt10.118gf`<sup>4)</sup>を作成してみましょう。

#### List 5-41 ● フォントのソース

```
1: % A file to be loaded after "plain.mf" and "cmbase.mf"
2: % for Human68k, TPTc version.
3: % written by Kaoru Maeda, May 6, 1989.
4: % arranged by T.Kawamoto, Oct 13, 1989.
5: base_version:=base_version&"/Human68k(TPTc)";
```

2)METAFONT の Ver.2 では、以下の点が Ver.1 とは異なるようです。ひとつは  $\text{\TeX}$  が導入した「ヴァーチャルフォント」の概念、つまり、複数の文字を組み合わせたものを、あたかもひとつの文字であるかのように扱うという「合字機能」に対応したということです。もうひとつは、メモリ容量が増え、また、グリフ(字形)の生成パターンが変更されたことによって、より複雑なフォントが描画できるようになったということです。

3)preview.x の Ver.2p07 以前は、自動的に代替のフォントを使うといった処理を行ってくれませんでした。

4)先の「METAFONT の重要な注意事項」(p.214)で説明したように、このファイル名はオリジナルの Human68k では使えません。



```

6:
7: screen_rows:=512; screen_cols:=768;
8:
9: mode_def TK =
10: proofing:=0;
11: fontmaking:=1;
12: tracingtitles:=0;
13: pixels_per_inch:=118;
14: blacker:=.0;
15: fillin:=.0;
16: o_correction:=1;
17: enddef;
18:
19: mode = TK;
20: mag := magstep0;
21: input cmtt10;
22: end;

```

List 5-41が METAFONT に入力されるフォントのソースです。一見したところでは、何かのプログラムのようには見えますが、これが METAFONT が使うソースなのです。X680x0 版の T<sub>E</sub>X では、このような METAFONT のソースを自分で一から作成して使うようなことはまずありません。詳細は後述しますが、本書の添付ディスクが構築する T<sub>E</sub>X (および METAFONT) システムには、フォント作成用のツールが用意されているからです。その意味では、List 5-41のような METAFONT のソースを、実際に目にすることはほとんどないでしょう。ここでは、List 5-41を “cmtt10.v0” というファイル名で作成したことにして話を先に進めます。

フォントの作成には inimf.x と virmf.x の2つのファイルを使います。inimf.x と virmf.x は、フォントの作成という観点からは違いはありません。ですが、inimf.x には virmf.x が使う METAFONT の内部情報を作成する機能があります。一度、この内部情報をファイルとして生成すれば、virmf.x でフォントを作成することができます。ですから、inimf.x は最初に一度使ったら、ほとんど使うことはなくなります<sup>5)</sup>。

では、実際にフォントを作成してみましょう。

```

A> virmf cmtt10.v0
This is METAFONT, C Version 2.7
(cmtt10.v0 (A:/MF/MFinputs/cmtt10.mf (A:/MF/MFinputs/roman.mf
(A:/MF/MFinputs/romanu.mf
.....
.....
Font metrics written on cmtt10.tfm.
Output written on cmtt10.118gf (128 characters, 5664 bytes).
Transcript written on cmtt10.log.

```

フォントのソース cmtt10.v0 に virmf.x で処理を行うと、cmtt10.tfm、cmtt10.log、cmtt10.118gf という3つのファイルが作成されます。

5) 毎回 inimf.x を使わないのは、内部情報生成の時間を減らすためです。

- cmtt10.tfm ファイルは、 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  がフォントの大きさなどを知るためのファイル
- cmtt10.log は、METAFONT がフォント作成の過程を出力したファイル
- cmtt10.118gf は、METAFONT が生成したフォントの本体

3 つのファイルのうち、実際に必要なのは cmtt10.tfm, cmtt10.118gf です。cmtt10.log は、フォント生成のときにエラーが発生していない場合には必要ありません。エラーが発生した場合には、ディスプレイ上のエラー情報がスクロールして画面から消えていても、この cmtt10.log を参照すればエラーの原因を知ることができます。

フォントを作成する場合に出力されるファイルの名のうち、cmtt10 の部分はフォントの種別を表します。この場合は、10 ポイントのタイプライタフェースフォントを示しています。

cmtt10.tfm は、cmtt10 という名前のフォントではすべて共通の内容が出力されているので、一度でも cmtt10 というフォントを作成したことがあれば、このファイル cmtt10.tfm はすでに  $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  が参照できるようになっているはずで<sup>6)</sup>。一度も cmtt10 というフォントを作成したことがなければ、 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  が必ずこの cmtt10.tfm を参照できるように所定の場所に<sup>7)</sup>コピーしておく必要があります。

cmtt10.118gf が実際のフォントファイルです。METAFONT は、

[フォント名].[dpi 値]gf

という形でファイル名を作成します。cmtt10.118gf の場合は、cmtt10 がフォントの種類<sup>8)</sup>、118 が [dpi 値] を示します。

この METAFONT が出力するフォントファイル (FontName.numgf) は冗長な部分をたくさん含んでいるので、普通は “gftopk” と呼ばれるプログラムで圧縮します。gftopk は、パソコン通信などで使われる LHA.x のようにファイルを圧縮するプログラム<sup>9)</sup>で、フォントファイルを圧縮してサイズを小さくします。

UNIX のようなファイル名の制限が少ないシステムでは、gftopk プログラムによって圧縮されたファイルを、

[フォント名].[dpi 値]pk

という生成規則で名前をつけて生成します。

これに対し、Human68k の  $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  システムでは、このようなファイル生成規則では拡張子の文字数が 3 文字を超えてしまうのでファイルを生成できません。このために、[dpi 値] 別にファイルをまとめたディレクトリをつくり、その下に同じサイズのフォントを集めて格納する方法を使っています<sup>10)</sup>。

つまり、

[フォントサイズ]\[フォント名].pk

という形でまとめられています。この方法の最大の問題点は、サイズは違っても、フォント名が同じになってしまう点です。一度にたくさんのフォントを作成する場合には十分注意してください。なお、X680x0 の世界には、gftopk は、そのまま gftopk.x という名前で移植されています。

6) 第 1.2.8 項「 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  が使うファイルの拡張子」(p.19) 参照。

7) 第 1.2.6 項「 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  が参照する環境変数」(p.16) 参照。

8) cm は Computer Modern の略、tt はタイプライタフェースの略、10 は 10 ポイントの略です。

9) gftopk は LHA.x のような汎用の圧縮プログラムではありません。

10) Human68k と同様の制限のある MS-DOS でも同様です。

```
A> gftopk cmtt10.118gf cmtt10.pk
This is GFtoPK, Version 1.4
' METAFONT output 1993.11.01:0439'
5664 bytes packed to 2284 bytes.
```

これで 118dpi の画面表示用の `cmtt10.pk` が作成されました。METAFONT では、ユーザがソースを作成することはほとんどありませんので、METAFONT に入力するフォントのソースの作成段階ではエラーが発生することはめったにありません<sup>11)</sup>。

11) ただし、ディスプレイ用の dpi が小さいフォントではエラーが発生します。これは、フォント自体が dpi の小さいフォント用には設計されていないためです。

### ◆ METAFONT のセットアップ

ここでは、X680x0 での METAFONT のセットアップについて説明します。内容的には第 1.2.3 項「 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  の種類と実行ファイル」(p.10)の説明に相当しており、セットアップ手順もほとんど同一です。`initex.x` に相当するのが `inimf.x`, `virtex.x` に相当するのが `virimf.x` です。METAFONT の場合は、 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  の `fmt` ファイルに相当するファイルは、拡張子が `.base` のファイルです。METAFONT にも、 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  と同様にフォントの作成方法のベースになるファイルがあります。これが `plain.mf` です。それでは実際に、`plain.base` を作成してみましょう。

12) この状態から、さらにベースとなるフォントを読み込ませることもできます。たとえば CM フォントは、その生成に `cmbase.mf` というファイルが必要とするので、このファイルも `base` ファイルに読み込んでおくと、CM フォントの生成にしか使用できないものの、フォント作成の時間が短くてすむ `base` ファイルができます。後述の `makefont.x` は、生成対象となるフォントのファイル名の頭 2 文字が「cm」であった場合、すでに `cmbase.mf` を読み込んでいる `cmplain.base` を、そうでなかった場合には `plain.mf` だけを読み込んだ `plain.base` を、それぞれ `base` ファイルとして使用するようになっていきます。

13) \dump でも可能です。

```
A> inimf plain
This is METAFONT, C Version 2.7 (INIMF)
(../inputs/plain.mf
Preloading the plain base, version 2.0: preliminaries,
basic constants and mathematical macros,
macros for converting from device-independent units to pixels,
macros and tables for various modes of operation,
macros for drawing and filling,
macros for proof labels and rules,
macros for character and font administration,
and a few last-minute items.)
*
```

`fmt` ファイルを作成する場合と同様に、最後に「\*」と表示されますので<sup>12)</sup>、これにキーボードから `dump`<sup>13)</sup> と入力します。



```

*\dump
Beginning to dump on file plain.base
(base=plain 93.11.1)
1369 strings of total length 22773
4669 memory locations dumped; current usage is 1320&3224
504 symbolic tokens
Transcript written on plain.log.
A>

```

これで `inimf.x` は、`plain.base` を書き出して終了します。この `plain.base` を  $\text{T}_{\text{E}}\text{X}$  のときと同じように「METAFONT が参照する環境変数」(p.220)で説明する環境変数 `MFBASES` が示すディレクトリにコピーしておきます。`plain.base` の作成が終わっていれば、METAFONT を使ってフォントを作成できます。

実際にフォントを作成する場合には `virmf.x` を用いて作成します。このとき、`base` ファイルを検索させる方法も  $\text{T}_{\text{E}}\text{X}$  のときとまったく同じです。

```

A> virmf &plain
This is METAFONT, C Version 2.7
...

```

この例のように、`virmf.x` は `base` ファイルの指定がない場合には、まず、`virmf.x` が起動されたフルパスに拡張子 `.base` を追加したファイルを検索します。`A:\BIN\virtex.x` が起動時のフルパスだとすれば、`A:\BIN\virtex.x.base` がまず検索されます。このファイル検索規則は、第 1.2.3 項「 $\text{T}_{\text{E}}\text{X}$  の種類と実行ファイル」(p.12)で説明したファイルネーム生成方法と同様です。METAFONT の場合は `.fmt` 拡張子を `.base` 拡張子で置き換えた生成方法となります。

第 1.2.4 項「より高度な環境構築」(p.14)で説明したリンクファイルを上手に設定しておけば、 $\text{T}_{\text{E}}\text{X}$  と同様に `mf` といったコマンドを作成して簡単に METAFONT を使うことができる環境を構築できます。

#### ◆ METAFONT のコマンドラインの書式

METAFONT の実行形式 `inimf.x` および `virmf.x` は、コマンドラインからオプションを指定して起動することはありません。コマンドライン上に指示されるのは、METAFONT の命令か、ソースファイルの指定だけです。

##### ○ `inimf.x` のコマンドラインの書式

```

inimf <ソースファイル>
inimf <コマンド>

```

<ソースファイル> は METAFONT のソースファイル名で、拡張子 .mf は省略可能です。<コマンド> は METAFONT のコマンドです。

- `virmf.x` のコマンドラインの書式

```
virmf [&<base ファイル>] <ソースファイル>
virmf [&<base ファイル>] <コマンド>
```

[&<base ファイル>] は省略可能です。<ソースファイル> は METAFONT のソースファイル名で、拡張子 .mf は省略可能です。<コマンド> は METAFONT のコマンドです。<base ファイル> については「METAFONT のセットアップ」(p.218)を参照してください。

このように、METAFONT にもコマンドラインから指定するようなオプション類はありません。

#### ◆ METAFONT が参照する環境変数

METAFONT は、以下のような環境変数を参照します。これらの環境変数が設定されていなかった場合には、ファイルの検索はカレントディレクトリだけが対象になります。パスを指定する場合は、 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  と同様に複数のパスを“;”で区切って並べて記述できます。

##### (1) MFINPUTS

コマンドラインや `\input` コマンドで指定されたファイルを検索するディレクトリを指定します。

##### (2) MFPOOL

METAFONT 実行ファイルが必要とする `mf.pool` を検索するディレクトリを指定します。`inimf.x` のみがこの指定を必要としますので、`base` ファイルを頻繁につくる方以外は、必要なときだけ指定しても実用上は問題ありません。なお、MFPOOL の環境変数では複数のパスを“;”で区切って並べて記述することはできません。

##### (3) MFBASES

コマンドラインから“&”で指定された `base` ファイルを検索するディレクトリを指定します。

#### 5.4.2 makefont によるフォント作成

ここまで、 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  システムで使用するフォントを作成するためのシステム METAFONT を説明してきました。ここでは、本書の添付ディスクによって構築した  $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  システムに用意されている、METAFONT のソースファイルを使用し

てあなたの環境にあわせたフォントを作成するためのツール `makefont.x` の使い方を中心に説明します。

#### ◆ `makefont.x` とは何か

`makefont.x` はフォント作成を補助するためのツールです。実際にフォントの作成を行うのは METAFONT というシステムですが、使用方法が難しいため、初心者にはちょっと扱いにくいかもしれません。そこで、簡単にフォントをつくれるようにしたツールが `makefont.x` です。このツールは任意サイズのフォントを L<sup>A</sup>T<sub>E</sub>X で使われる拡大率で作成することができます。本書の添付ディスクが構築する T<sub>E</sub>X システムに標準で用意されているフォントは、添付ディスクの枚数の都合で最低限のものしか入れていませんので、プレビューアやプリンタドライバで T<sub>E</sub>X ファイルを見たり、印刷するときにフォントが足りないことがあります。フォントが足りないと、プレビューアやプリンタドライバは次のようなエラーメッセージを出力します。

```
No Font : 118/manfnt.pk が見つかりません。
RETRY ? (y/n)
```

このようなときに、`makefont.x` を使用してフォントを作成してください。

#### ◆ `makefont.x` を使うための準備

`makefont.x` を使用してフォントを作成するためにはいくつかの条件があります。この条件を満たしていないと、フォントの作成を行うことができません。

##### (1) 環境変数の設定

環境変数 `MFHOME` および `TEXPK` が設定されていないと、`makefont.x` は起動しません。必ず設定してください。もちろん、それらの環境変数が示すディレクトリは存在していなければなりません。これらの環境変数は、本書の添付ディスクに付属するインストーラを使ってインストールしているのであれば、`TeXenv.bat` を実行することで適切に設定されます。詳細は第2章「Install」(p.21)を参照してください。

##### (2) パスの設定

`makefont.x` を起動するには、環境変数 `MFHOME` に設定してある `bin` ディレクトリにパスが通っていなければなりません。これも `TeXenv.bat` を実行することにより設定されます。

##### (3) METAFONT のディレクトリ構成

`makefont.x` は、デフォルトでは本書のインストーラでインストールされる



環境で動作します。ディレクトリ構成が異なっている場合は環境変数を指定するか、オプションでディレクトリを指定しないと使用できません。デフォルトでは、環境変数 `MFHOME` で示される以下のディレクトリのいずれかに METAFONT のソースファイルがなければなりません。

- ☐ `cmfonts`
- ☐ `LaTeXfonts`
- ☐ `otherfonts`
- ☐ `utilityfonts\bases`
- ☐ `utilityfonts\black`
- ☐ `utilityfonts\committee`
- ☐ `utilityfonts\gray`
- ☐ `utilityfonts\half`
- ☐ `utilityfonts\logo`
- ☐ `utilityfonts\manualfonts`
- ☐ `utilityfonts\mfbook`
- ☐ `utilityfonts\slant`

METAFONT のソースファイルが上記と異なるディレクトリにある場合は環境変数 `MFINPUTS` か、後述するオプション “`-mf`” でそのディレクトリを指定しなければなりません。

#### ◆ 対話形式によるフォント作成

コマンドライン上で、

```
A>makefont
```

を実行すると、フォント作成プログラム `makefont.x` が起動します。画面上には作成フォントの dpi 値の一覧が表示されます。

作成するフォントの dpi 値を選択して下さい。(1 - 7)

(1)118 (2)180 (3)240 (4)300 (5)360 (6)400 (7)その他

push 1-6

ここで、あなたの作成したいフォントの基本 dpi 値を番号で選択してください。画面表示用のフォントなら 118dpi、プリンタ用のフォントなら使用するプリンタの dpi 値を指定します。一覧のなかにない場合は、「(7) その他」を選択すれば dpi 値を直接指定することもできます。「(7) その他」を選んだ場合、次のように表示されるので、50 から 3000 までの数値を入力してください。

作成するフォントの dpi 値を入力して下さい。:

dpi 値として 118 を指定すると、以下に示すような拡大率の一覧が表示されます。拡大率の一覧のなかから、実際に作成するフォントのサイズを決定します。

作成するフォントの拡大率を選んで下さい。(1 - 13)

1) 0.5( 59dpi) 2) 0.6( 71dpi) 3) 0.7( 83dpi)  
 4) 0.8( 94dpi) 5) 0.9(106dpi) 6) 1.0(118dpi)  
 7)1.095(129dpi) 8) 1.2(142dpi) 9) 1.3(153dpi)  
 10)1.44(170dpi) 11)1.728(204dpi) 12)2.074(245dpi)  
 13)2.488(294dpi)

magstep0 = 1.0 magstephalf = 1.095 magstep1 = 1.2 magstep2 = 1.44  
 magstep3 = 1.728 magstep4 = 2.074 magstep5 = 2.488

input no:

あなたの作成するフォントの拡大率の番号を入力してください<sup>14)</sup>。作成されるフォントのサイズは「dpi 値×拡大率」の小数点以下第 1 位を四捨五入した値になります。

肝心の拡大率の求め方ですが、たとえば、プレビューア preview.x が次のようなエラーメッセージを出力したとしましょう。

No Font : 142/cmbx8.pk が見つかりません。  
 RETRY ? (y/n)

作成するフォントの dpi 値は「(1) 118」を指定し、拡大率は「(8) 1.2」を選びます<sup>15)</sup>。

このとき、なぜ dpi 値として直接 142dpi を指定しないかというと、142dpi の拡大率 1.0 倍と 118dpi の 1.2 倍とは微妙に異なるためです。拡大、縮小用のフォントを作成する場合、指定する dpi 値は基本サイズの dpi 値に揃えて拡大率を変更したほうが全体のバランスがとれます。とはいえ、それほど大きな差はありませんので、こだわらない方なら作成フォントの dpi 値を直接指定して拡大率を 1.0 としてもかまいません。

ここまで終わると、画面に作成フォント一覧が表示されます。

作成フォントをカーソルキーで選択してリターンキーを押して下さい。

|        |        |        |         |        |
|--------|--------|--------|---------|--------|
| accent | aphalf | ascgrp | bigacc  | bigdel |
| bigop  | black  | blaps  | bidov   | blesp  |
| calu   | cmb10  | cmbase | cmbsy10 | cmbx10 |
| cmbx12 | cmbx5  | cmbx6  | cmbx7   | cmbx8  |
|        |        | :      |         |        |
|        |        | :      |         |        |

14)magstep とは TeX で基本となるフォントの拡大率で、1 増えるごとに 1.2 倍になります。一般的な TeX のフォントは magstep で示される倍率に拡大されています。

15)118dpi はプレビューアによる画面表示用フォントの基本サイズです

16) 作成フォントの一覧には、拡張子 `.mf` のファイルがすべて表示されます。しかし、このなかには `accent` のように他の METAFONT ソースから読み込まれるような、直接指定できないソースも含まれていいますので注意してください。

カーソルキーで作成したいフォントを選択してリターンキーを押すと、作成フォントが決定されます<sup>16)</sup>。この段階でフォント複写先のディレクトリがない場合は、以下のようにディレクトリを作成するかどうかを聞いてきますので、作成するなら Y キーを押してください。N キーを押すと、フォント作成を中止します。

フォントの複写先ディレクトリ `c:/tex/fonts/142` は存在しません。  
ディレクトリを作成しますか? (Y/N)

また、作成フォントがすでに存在する場合には作業を中止してもよいか聞いてきます。Y キーを押すと作業を中止し、N キーを押すと処理を続行します。

指定されたフォントは既に存在します。作業を中止しますか? (Y/N)

すべての指定が終わると、フォントの作成を開始します。エラーメッセージが出力されることなくプログラムが終了すれば、フォントの作成は終了です。

#### ◆ オプション指定によるフォント作成

ここまでは、`makefont.x` のメニューから dpi 値や拡大率、作成フォント名を対話的に指定しましたが、`makefont.x` はコマンドラインでオプションを指定して実行することによりフォントを作成することもできます。コマンドの書式は以下のとおりです。

```
makefont <dpi サイズ> <拡大率> <作成フォント名>
[-y] [-mf= <ディレクトリ名>]
```

各パラメータの意味は、

##### <dpi サイズ>

作成フォントの基本 dpi 値のことです。50 から 3000 までの数値を指定してください。値の省略はできません。

##### <拡大率>

作成フォントの拡大率です。以下の表のとおりに指定してください。省略はできません。

```
0.5=m05 0.6=m06 0.7=m07 0.8=m08 0.9=m09 1.0=m0 1.095=mh
1.2=m1 1.3=m13 1.44=m2 1.728=m3 2.074=m4 2.488=m5
```

##### <作成フォント名>

作成するフォントの METAFONT ソースファイル名です。拡張子の `.mf`



は、指定しないでください。指定ソースファイル名の例を以下に示します。省略はできません。

| ソースファイル名 | 作成されるフォント                     |
|----------|-------------------------------|
| cmr10    | Roman face (ローマン体)            |
| cmti10   | <i>Italic face</i> (イタリック体)   |
| cmcsc10  | SMALL CAPS FACE (小文字の大きさの大文字) |
| cmsl10   | <i>Slanted face</i> (斜体)      |
| cmss10   | Sans serif face (サンセリフ体)      |
| cmtt10   | Typewriter face (タイプライタ体)     |

フォントの詳細は『Vol.2 — Reference 編』の第4章「 $\text{\TeX}$  Fonts」(p.133)を参照してください。

-y

このオプションをつけていると、いちいち確認を求めてきません。作成先ディレクトリがない場合は、あなたの確認をとらずにディレクトリを作成し、すでに作成フォントが存在する場合には確認をとらずにフォント作成作業を終了します。また、METAFONT ソースにエラーがあっても、エラーメッセージを出力するだけで作業を続行します。このオプションは、バッチファイル等で連続してフォントを作成するときに指定すると便利です。

-mf= <ディレクトリ名>

指定されたディレクトリから METAFONT ソースファイルを探します。

以上のオプション指定のなかで <dpi サイズ>、<拡大率>、<作成フォント名> の3つは省略できませんし、順番も固定です。あとの2つのオプションは必要に応じて指定してください。順不同です。

参考までに、いくつかの指定例を以下に示します。

- 360dpi の lasy10 フォントを magstep1 で作成する場合

```
makefont 360 m1 lasy10
```

- 180dpi の cmr10 フォントを magstep0 で、あなたへの確認をとることなく作成する場合

```
makefont 180 m0 cmr10 -y
```

- 300dpi の cmbx5 フォントを 0.5 倍で、a:\metafont ディレクトリからソースファイルを探して作成する場合

```
makefont 300 m05 cmbx5 -mf=a:\metafont
```

- 118dpi の cmmi10 フォントを 0.7 倍で、c:\texfonts ディレクトリからソースファイルを探して、あなたへの確認をとることなく作成する場合

```
makefont 118 m07 cmmi10 -mf=c:\texfonts -y
```

#### ◆ フォント作成中のエラー

METAFONT は、エラーが発生すると、たとえば以下のように “?” のプロンプトを表示して処理を中断します。

```
! Strange path (turning number is zero).
<to be read again>

1.776...

?
```

もし、不幸にもフォントの作成中にこのような状態に陥ったら、そのフォントの作成はあきらめるのが無難です。ただし、なかにはエラーを無視してフォントを作成しても、それほど影響のない場合もあります。とりあえずエラーを無視して作成しておいて、そのフォントをテストしてみるのもよいでしょう。ただし、エラーが出るということは、なんらかの問題があるということです。使用するには十分に注意してください。

さて、先ほどのエラーの例は、ある領域を塗りつぶそうとしたとき、その領域の境界をなす線が交差してしまっている場合に起きる、118 dpi 以下の低解像度のフォントを作成するときに起きやすいエラーです。“?” プロンプトを出して中断したとき、フォントの作成を中止するには X キーを入力します。また、エラーを無視して処理を継続するときはリターンキーを押します。以降のエラーをすべて無視してフォントの作成を継続するときには R キーを入力します。このあたりのキー入力は、「エラー (Error)」(p.86) で説明した  $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  でエラーが起こった場合と同じですから、参考にしてください。なお、 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  の場合もそうでしたが、リターンキーによってエラーをスキップした際には、最終的にフォントの作成に失敗するケースもありますので注意してください。

また、makefont.x のオプション “-y” が指定してあると、エラーが発生してもエラーを無視して処理を継続します。この場合、エラーの発生したフォントはあとで削除しておくといよいでしょう。

エラーを無視して強引に作成したフォントが、どの程度まともにできあがっているのかをテストしてみたい方は以下の手順でテストしてみてください。

まず、環境変数等の設定をしてフォントマネージャも組み込んでおきます。そして、

```
A>tex c:/tex/macros/testfont.tex
```

のように入力します<sup>17)</sup>。

すると、

```
Name of the font to test =
```

と表示されますので、cmr10 等テストしたいフォント名を指定します。すると、“\*” プロンプトが出ますので、ここで “\table” と入力してください。しばらくすると、再度 “\*” プロンプトが表示されます。今度は、“\end” と入力すると、testfont.dvi ファイルが作成されます。ここで、

```
A>preview testfont.dvi
```

と入力してリターンキーを押すと、プレビューでフォントテーブルの一覧が見られます。もし、正しく表示されなかったり、汚くて実用に堪えないフォントが生成されていたりしたら、そのフォントの作成はあきらめて削除してしましましょう。

17) 環境変数 TEXHOME で設定されているディレクトリが c:/tex の場合。

#### ◆ makefont.x のエラーメッセージ一覧

- TwentyOne.x が組み込まれていません。処理を中止します。

makefont.x を実行するには、TwentyOne.x が組み込まれていなければなりません。あなたが使用している Human68k のバージョンにあわせて TwentyOne.x を選択し、常駐させてください<sup>18)</sup>。

- 環境変数 MFHOME が設定されていません。処理を中止します。
- 環境変数 TEXFONTS が設定されていません。処理を中止します。
- 環境変数 TEXPK が設定されていません。処理を中止します。

makefont.x は上記 3 つの環境変数の設定が必要です。すでに説明したように、TeXenv.bat を実行することにより、T<sub>E</sub>X 実行に必要な環境変数が設定されますので、これを設定してから makefont.x を実行してください。

- 環境変数 MFHOME が示すディレクトリが存在しません。処理を中止します。
- 環境変数 TEXFONTS が示すディレクトリが存在しません。処理を中止します。
- 環境変数 TEXPK が示すディレクトリが存在しません。処理を中止します。

本書の添付ディスクに付属しているインストーラで構築した環境であれば、本来出ないはずのエラーです。環境変数が示しているディレクトリが存在しないときに表示されます。正しく設定してください。

- パラメータの数が誤っています。
- 第 ? パラメータが異常です。処理を中止します。

18) TwentyOne.x の選択については、第 2.2.2 項「インストールの前に準備しておくこと」(p.30) で詳しく述べています。



- 第3パラメータの示す METAFONT ファイルは存在しません。処理を中止します。

これらはオプションを指定して起動するとき、パラメータが間違っていると出力されるメッセージです。指摘されたパラメータを正しく指定してください。

- ESC キーが押されましたので処理を中止します。

`makefont.x` は、キー入力待ち状態で ESC キーを押すことによって終了することができます。

- 入力値が異常です。正しい値を入力して下さい。

数値入力時に範囲外の値を入力すると出るメッセージです。正しい値を入力してください。

- 指定されたフォントは既に存在しています。作業を中止します。

指定フォントがすでに存在するときに出力されるメッセージです。“-y” オプションが指定されていないと、このメッセージが出力される前に処理を続行するかどうか確認してきます。

- 複写先のディレクトリが存在しないので処理を中止します。

複写先のディレクトリが存在せず、“-y” オプションが指定されていないと、ディレクトリの作成を行うかどうかを確認してきますが、このときに「ディレクトリを作成をしない」を選択すると、このメッセージを出力して終了します。

- 環境下に \*.mf ファイルが存在しません。処理を中断します。

- ファイルが見つかりません。作業を中止します。

これらのエラーは、本書添付ディスク付属のインストーラ `install.x` で正しくインストールされ、`TeXenv.bat` で環境変数等を設定しているなら出るはずのないエラーです。もし、これらのエラーが出るようなら  $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  のインストールに失敗しています。インストールをやりなおしてください。

---

## T<sub>E</sub>Xfamily

ある程度の汎用性を持たせたマクロやスタイルファイルを作成し、これをまとめたものを「マクロパッケージ」といいます。本書の作成に使用している L<sup>A</sup>T<sub>E</sub>X も、このようなマクロパッケージのひとつですが、ほかにもこのようなマクロパッケージはたくさんあります。これらは T<sub>E</sub>X の機能をより直接的に高めようとする試みの成果といえるでしょう。

その一方で、T<sub>E</sub>X の入出力を支援することで、T<sub>E</sub>X の機能を間接的に高めようとする試みも続けられています。T<sub>E</sub>X の強力な組版機能を生かすために、あるいは T<sub>E</sub>X での文書作成効率を上げるために、そしてなにより T<sub>E</sub>X による文書作成をより快適にするために、さまざまなアドオンツールが作成されているのです。

本章では、このような T<sub>E</sub>X のさまざまなマクロパッケージ、スタイルファイル、アドオンツールのうち、主要なものを紹介します。

---

## 書式説明

本章では、TeX オリジナルのパッケージと日本語化されたパッケージがある場合にも、明示的な区別をしていません。たとえば、plain TeX も plain pTeX も同じパッケージとみなして、「plain TeX」として解説しています。これは紙面の都合および説明の煩雑さを避けるためですが、その点を注意してお読みください。

概 略： 当該マクロやスタイルファイル、アドオンツールの簡単な解説です。

所 在： ここに紹介するもののなかには、本書でサポートしていないものもあります。それらについて、日本語版のパッケージがある場合には日本語版パッケージの、ない場合にはオリジナルパッケージの入手先を以下のフォーマットで示します。

*Net / Forum / Library / Number : Filename*

*Net* : nifty は NIFTY-Serve を、pcvan は PC-VAN を表します。X680x0 版の TeX に関する実行ファイルの多くが NIFTY-Serve においてサポートされていることから、所在に関してはできるだけ NIFTY-Serve での所在を示しました。

*Forum* : NIFTY-Serve ではフォーラム、PC-VAN では SIG を表します。

*Library* : NIFTY-Serve の場合、libn でライブラリの  $n$  番を、mesn で会議室の  $n$  番を表します。PC-VAN の場合には、osln で OSL の  $n$  番を、forumn でフォーラムの  $n$  番を表します。

*Number* : 登録番号です。以下のような記述を組み合わせています。番号が前後している場合、その順番で組み合わせなければならないので注意してください。

- $n : n$  番のデータ番号で登録されていることを示します。
- $n, m : n$  番および  $m$  番のデータ番号で登録されていることを示します。このとき、両方の番号のデータをあわせるとひとつのパッケージになります。
- $n-m : n$  番から  $m$  番までのデータ番号で登録されていることを示します。このとき、すべての番号のデータをあわせてひとつのパッケージになります。

*Filename* : 登録されているアーカイブファイルのファイル名です。



本書の初版執筆時点の情報をもとにしていますので、その後変更される可能性があることをお断りしておきます。

使 用 法： 日本語版のパッケージがある場合には日本語パッケージを前提として、インストールの方法を中心に説明します。ただし、ここに示した使用手段がすべてではありません。あくまで一例と考えてください。

マニュアル： ここに示すドキュメントには、以下の 2 つがあります。

- 開発者の手によるドキュメントのうち、パッケージのアーカイブに含まれていないもの。
- パッケージ作成上、規格の決定を左右する役割を果たしたものの。

原著が外国語である場合、邦訳されているものについては邦訳を挙げています。

なお、パソコン通信で入手できるものについては「所在」の箇所で示した形式に準じて入手先を示します。

解 説 書： 開発者以外の手による邦文書のうち、主要なものを挙げておきます。たとえば、市販されている書籍のうち、解説書ないし教科書として使用する場合には便利であろうというものです。

パソコン通信で入手可能なものについては、「所在」で示した形式に準じて入手先を示します。

なお、一部のマクロあるいはアドオンツールについては、当該マクロないしアドオンツールを使用して作成したサンプル出力を掲載しています。このサンプル出力を行うために使用したソースファイルと若干のコメントあるいはドキュメントが、添付ディスク 7 の \sample 配下にありますので、参考にしてください。

また、無茶な日程であるにもかかわらず、心よくサンプル作成にご協力くださった、なべじゅん氏・YODA 氏・彩人氏・仁泉大輔氏 (掲載順) に、紙面を借りてお礼申し上げます。ありがとうございました。

plain T<sub>E</sub>X

概 略： D.E. Knuth 博士が T<sub>E</sub>X を発表したときに添付されていたマクロパッケージです (その後、若干の改良がなされています)。マクロなどは必要最小限しか定義されていませんし、後述する L<sup>A</sup>T<sub>E</sub>X (p.233) などのようなグローバルな自動処理もほとんど行いません。つまり、L<sup>A</sup>T<sub>E</sub>X はユーザが明示していない組版処理を処理系が独自に行いますが、plain T<sub>E</sub>X はこれを行わない分だけ、指示したとおり、そのままの出力が得られます。したがって、結果的に plain T<sub>E</sub>X のほうが L<sup>A</sup>T<sub>E</sub>X より自由な組版処理を実現できるのです。

いいかえれば、L<sup>A</sup>T<sub>E</sub>X と plain T<sub>E</sub>X の違いは、プログラミング言語でいえば C 言語とアセンブラの違いとでもいえるのでしょうか。はじめて使うときは L<sup>A</sup>T<sub>E</sub>X のほうが使いやすいのですが、ある部分を各処理系において標準で設定されている形式と少しだけ変えたい (たとえば、行列や式のなかのある部分を少しだけ動かす) とか、指定された特定のフォーマットで文書を作成しようとかいうときには plain T<sub>E</sub>X のほうが実現しやすくなります。

ただし、それを実現するための煩雑な処理は、文章の作成者自身が明示的にソース中のコマンドで指定しなければなりませんから、T<sub>E</sub>X および組版についてのかなりの知識を要求されることになってきます。

所 在： 添付ディスクでインストールされるファイルに含まれています。

使 用 法： `fmt` ファイルは添付ディスクでインストールされるファイルに含まれています。再作成する場合には、以下のようになります。

```
A> initex.x \input jplain.tex \dump
```

1) 必ずしも読みやすい本とはいえませんが、plain T<sub>E</sub>Xにかぎらず、本格的にマクロを書いてみようという人には必読の書といえます。ただし、T<sub>E</sub>Xの初心者が最初に読むのはつらいでしょう。

マニュアル： 『[改訂新版] T<sub>E</sub>X ブック』<sup>1)</sup>

Donald E. Knuth 著 / 斉藤信男 監修 / 鷺谷好輝 訳

アスキー 1992

解 説 書： 『T<sub>E</sub>X 入門』

大野義夫 著 共立出版 1989

概 略: T<sub>E</sub>X ファミリーにおいて最も代表的ともいえるマクロパッケージ。  
DEC 社の L. Lamport 氏が UNILOGIC 社の文書整形システム  
Scribe を参考に作成したものです。

文章や数式の処理のみならず、表や簡単な図をかくためのフォン  
トやマクロも用意されており、使い勝手が大変よくなっています。  
また、スタイルファイルの概念を導入したことで異なる体裁の文  
書を簡単に作成でき、さらに目次・相互参照・文献表の作成といっ  
たグローバルな処理ができることも特徴といえるでしょう<sup>1)</sup>。

現在、T<sub>E</sub>X のソースファイルといえば、ほとんどが L<sup>A</sup>T<sub>E</sub>X のマ  
クロパッケージで作成されているといってもよいでしょう。

所 在: 添付ディスクでインストールされるファイルに含まれています。

使 用 法: `fnt` ファイルは添付ディスクでインストールされるファイルに含  
まれています。再作成する場合には、以下のようにします。

```
A> initex.x \input jlplain.tex \dump
```

マニュアル: 『文書処理システム L<sup>A</sup>T<sub>E</sub>X』

L. Lamport 著 / Edger Cooke・倉沢良一 監訳 /

大野俊治・小暮博道・藤浦はる美 訳 アスキー 1990

解 説 書: 『楽々 L<sup>A</sup>T<sub>E</sub>X 第 2 版』

野寺隆志 著 共立出版 1994

『L<sup>A</sup>T<sub>E</sub>X トータルガイド』

伊藤和人 著 秀和システムトレーニング 1991

『L<sup>A</sup>T<sub>E</sub>X 美文書作成入門』

奥村晴彦 著 技術評論社 1991

『L<sup>A</sup>T<sub>E</sub>X 自由自在』

磯崎秀樹 著 サイエンス社 1992

『日本語 L<sup>A</sup>T<sub>E</sub>X 定番スタイルファイル集 1, 2』

鷲谷好輝 著 インプレス 1991, 1992

『ひろのぶの L<sup>A</sup>T<sub>E</sub>X 入門』<sup>2)</sup>

鈴木裕信 著 1990

nifty / fprint / lib7 / 189 : LATEXGUD.LZH<sup>3)</sup>

『L<sup>A</sup>T<sub>E</sub>X のマクロやスタイルファイルの利用 Ver.2.15<sup>4)</sup>』

岩熊哲夫・古川徹生 著 1994

nifty / fsnote / lib4 / 114 : STYLEUSE.LZH

1) 文献表の作成のための文  
献データベースおよび当  
該データベースと L<sup>A</sup>T<sub>E</sub>X  
文書との間で橋渡しをす  
るプログラムのセットが  
BibT<sub>E</sub>X であり、参考文  
献の番号の管理なども自  
動的に行ってくれます。

2) 『やさしい L<sup>A</sup>T<sub>E</sub>X のは  
じめかた』(同著 オーム  
社 1991) のもとになっ  
たドキュメント。

3) 添付ディスク 8 に含ま  
れています。

4) アスキーの『UNIX  
MAGAZINE』で 1994  
.3 から連載している『ス  
タイルファイル活用法』  
のもとになったドキュメ  
ントです。



V<sub>F</sub>-L<sup>A</sup>T<sub>E</sub>X

1) 「マニュアル」の箇所を参照してください。

2) 和文フォントの命名による混乱が起きないように、アスキーが日本語 T<sub>E</sub>X 用に定めた日本語フォントの命名規則です。和文フォント名には、メーカー名・ファミリー名・ウェイト・バリエーション(各種文字変形)といった要素を盛り込まなければなりませんと定めています。

3) Z's STAFF に付属しているもの、あるいは書体倶楽部として独自に売られているものと同じフォーマットであれば利用できます。MS-DOS 用のフリーソフトのなかには、ビットマップフォントをツアイトのアウトラインフォントのフォーマットにコンバートするプログラムもありますから、これを用いれば V<sub>F</sub>-L<sup>A</sup>T<sub>E</sub>X をフル活用することも可能です。

4) 以上のファイルは、添付ディスク 8 に含まれています。

5) ファイルの変更箇所へのパッチデータを含んだファイルのこと。一般的に、テキストファイルの差分は diff.x でとられることが多く、作成された差分情報は patch.x を使用することで適用します。

概 略: 『日本語 T<sub>E</sub>X テクニカルブック I』<sup>1)</sup> の「日本語 T<sub>E</sub>X の和文フォントの命名規則」<sup>2)</sup> にもとづき、日本語フォントの多書体 (Various Font) 化に対応させた、X680x0 にだけ存在する L<sup>A</sup>T<sub>E</sub>X。X680x0 での対応は、許先明 (Seirios) 氏の提唱に始まりました。川本琢二 (Ext) 氏による fontman.x と zs.sys の対応、および山崎健 (融) 氏による kage.sys の作成を経て、最終的にマクロおよびフォント関連を許先明氏が V<sub>F</sub>-kit に統合して完成に至っています。

書体としては、今のところ、

明朝体・ゴシック体・教科書体・毛筆体・丸文字

の 5 書体を選択して使用できるようなスタイルオプションファイルが用意されています。

さらに、明朝体とゴシック体については、

- 影付き・白抜き・網かけ・立体の飾り文字
- 10%・20% の平体・長体シリーズ
- +10°・+20° の斜体シリーズ
- 細・普通・太のウェイト

を選択して使用できるスタイルオプションファイルも作成されており、これを他の書体に対応させることも可能でしょう。ただし、変形シリーズはツアイトのアウトラインフォント<sup>3)</sup>を前提としていますので、ビットマップフォントでは飾り文字しか使用できません。

いろいろなフォントを同一文書で使用する場合には BigT<sub>E</sub>X が必要です。

所 在: nifty / fsharp3 / lib9 / 98 : vflatex.lzh  
nifty / fsharp3 / lib9 / 100 : vfkit\_last.lzh  
nifty / fsharp1 / mes18 / 321 : vf\_fix.lzh<sup>4)</sup>

使 用 法: 「所在」の箇所で示したアーカイブに含まれる差分ファイル<sup>5)</sup>を lplain.tex などに適用し、その結果作成される vflplain.tex を initex.x にかけて V<sub>F</sub>-L<sup>A</sup>T<sub>E</sub>X の fmt ファイルを作成します。

```
A> initex.x \input vflplain.tex \dump
```

マニュアル: 『日本語 T<sub>E</sub>X テクニカルブック I』

アスキー出版局責任編集 アスキー 1990

# 多書体対応 L<sup>A</sup>T<sub>E</sub>X

V<sub>F</sub>-L<sup>A</sup>T<sub>E</sub>X

V<sub>F</sub>-L<sup>A</sup>T<sub>E</sub>X は、アスキーの提唱をもとに、次のような変形シリーズをサポートしています。これらは、フォントマネージャの力によって実現された、(現在のパーソナルレベルでは) X680x0 の T<sub>E</sub>X にだけ可能な出力です。ただし、これらはツァイトのベクトルフォントでなければ使用できません。

- 通常体
- 斜体シリーズです。slant.sty は、+10°・+20° の斜体をサポートしています。
- 平体シリーズです。hira.sty は、10%・20% の平体をサポートしています。
- 長体シリーズです。cho.sty は、10%・20% の長体をサポートしています。

この他にも、V<sub>F</sub>-L<sup>A</sup>T<sub>E</sub>X では以下のような字体シリーズがサポートされています。これらのシリーズは、フォントファイルに依存しませんので、ビットマップフォントやベジェフォントでも利用することが可能です。しかも、既述の変形シリーズとの組み合わせも OK。

- 通常字体。通常体に、斜体に、平体に、長体。まさに自由自在。
- 影付き体。通常体に、斜体に、平体に、長体。まさに円転滑腕。
- 白抜き体。通常体に、斜体に、平体に、長体。まさに活潑自在。
- 網かけ体。通常体に、斜体に、平体に、長体。まさに隨機应变。
- 立体体。通常体に、斜体に、平体に、長体。まさに縦横無尽。

以上のような各シリーズは、明朝体だけではなく、異なるフォントでも実現可能です。

- 角ゴシック体。「強調」だったら任せとけ。
- 丸ゴシック体。「強調」だったら任せてね。
- 教科書体。「何でも御座れ」で任せなさい。
- 毛筆体。「年賀シーズン」ならお任せでござる。

あれへ、丸文字はぁ。どうして入れとくれないのぁ、よくおかんばあい。

「あなたも禁断の木の実を食べてみませんか？」

---

SL<sub>A</sub>TeX

---

**概 略：** OHP などのスライド原稿出力用の L<sub>A</sub>TeX です。映写した際の文字などが読みやすいように、フォントサイズの `\normalsize` に 17 ポイント、つまり、通常の L<sub>A</sub>TeX において `\normalsize` が 10 ポイントの場合の `\LARGE` にあたる大きさが設定されています。また、モノクロ、カラー (赤・青・黒) を使い分けてスライドを作成することも可能です。ただし、カラースライドを作成するといっても、スライドシートに対して直接カラーで出力をするという意味ではありません。モノクロで出力した原稿を、加熱するとカラー発色する透明スライドシートにコピーすることを想定しているようです。

なお、OHP などのスライド原稿を作成する場合には、L<sub>A</sub>TeX 用スタイルファイルの `seminar.sty` もあります。こちらは通常の L<sub>A</sub>TeX をベースに利用できますので、便利かもしれません。Timothy Van Zandt 氏によって作成されました。

**所 在：** SL<sub>A</sub>TeX は添付ディスクでインストールされるファイルに含まれています。

nifty / fsnote / lib4 / 131 : SEMINAR.LZH

**使 用 法：** SL<sub>A</sub>TeX の場合、`slitex.tex`, `latex.tex`, `hyphen.tex`, `jsfonts.tex`, `kinsoku.tex` を用意して、以下のようにして `fmt` ファイルを得ます。

```
A> initex.x \input jsplain.tex \dump
```

`seminar.sty` については、付属のサンプルを参照してください。

**マニュアル：** 『文書処理システム L<sub>A</sub>TeX』

L.Lamport 著 / Edger Cooke・倉沢良一 監訳 /

大野俊治・小暮博道・藤浦はる美 訳 アスキー 1990



## seminar.sty とは？

seminar.sty は、スライドや OHP シートへの出力を目的に作成された  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  用のスタイルファイルで、次のような特徴を持っているようです。

- $\mathcal{A}\mathcal{M}\mathcal{S}$ - $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  と同様の数式記述が可能である。
- $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  のフォントはもとより、 $\mathcal{A}\mathcal{M}\mathcal{S}$ - $\text{T}_{\text{E}}\text{X}$  や PostScript のフォントも使用可能である。
- PostScript に通る色を設定可能である。
- 拡大処理を行うコマンドが用意されている。

Slide 1

## $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ がベース

$\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  で使用できるコマンドは、ほとんどそのまま使用できるようです。たとえば表。

| 物理定数の名称 | 記号  | 値                        | 単位                        |
|---------|-----|--------------------------|---------------------------|
| 万有引力定数  | $G$ | $6.6720 \times 10^{-11}$ | $\text{Nm}^2/\text{kg}^2$ |
| 重力加速度   | $g$ | 9.80665                  | $\text{m}/\text{s}^2$     |

そして、数式だってこのとおりです。

$$\int_0^{\frac{\pi}{3}} x \sin 2x dx = \left[ x \left( -\frac{1}{2} \cos 2x \right) \right]_0^{\frac{\pi}{3}} - \int_0^{\frac{\pi}{3}} 1 \cdot \left( -\frac{1}{2} \cos 2x \right) dx$$

Slide 2

---

 $\mathcal{A}\mathcal{M}\mathcal{S}$ -TeX

---

概 略： アメリカ数学会 (American Mathematical Society : AMS) の Michael D. Spivak 氏によって作成された TeX のマクロパッケージ。特に数式の取り扱いに優れており、多くの数学記号や German Fraktur (ドイツ語の旧字体) などのフォントを持っています。

$\mathcal{A}\mathcal{M}\mathcal{S}$ -TeX は、plain TeX に  $\mathcal{A}\mathcal{M}\mathcal{S}$ -TeX のマクロを組み込んだもので、基本的には plain TeX の上位コンパチブルになっています。ただし、定義が変わっている部分もあるので注意が必要です。たとえば、文字 “@” は、plain TeX ではそのまま書いてもかまいませんが、 $\mathcal{A}\mathcal{M}\mathcal{S}$ -TeX では特殊な意味に使われるので、単なる文字として使うときには、“\@” と書かなければなりません。数式などの取り扱いと若干のマクロが増えたこと以外は、plain TeX と大きな違いはありません。LaTeX でいうスタイルファイルのような概念も導入されていますが、今のところ、amspt.sty<sup>1)</sup> しかないようです。

1) AMS PrePrint :  
AMS のプレプリント  
スタイル (AMS への投  
稿形式)。

所 在：  $\mathcal{A}\mathcal{M}\mathcal{S}$ -TeX の Ver.1.1a は添付ディスクでインストールされるファイルに含まれています。

2) Ver.2.1 です。

nifty / flabo / lib11 / 38 : AMSTeX.Lzh<sup>2)</sup>

3)  $\mathcal{A}\mathcal{M}\mathcal{S}$ -TeX Ver.2.1 で  
使用可能な METAFONT  
のソースです。

nifty / flabo / lib11 / 41 : AMSfonts.Lzh<sup>3)</sup>

使 用 法： 以下のようにして  $\mathcal{A}\mathcal{M}\mathcal{S}$ -TeX の fmt ファイルを得ます。

```
A> initex.x \input jplain \input amstex.tex \dump
```

マニュアル： 『The Joy of TeX — A Gourmet Guide to Typesetting with the  $\mathcal{A}\mathcal{M}\mathcal{S}$ -TeX Macro Package』

M. D. Spivak 著 American Mathematical Society 1986

解 説 書： 『もっと<sup>3)</sup> $\mathcal{A}\mathcal{M}\mathcal{S}$ -TeX』

野寺隆志著 共立出版 1991

《CAYLEY-HAMILTON の定理》.

$A$  を正方行列として、 $\Phi_A(\lambda)$  を  $A$  の固有多項式とすれば、

$$\Phi_A(A) = O$$

が成立する。

PROOF:

$A$  の次数を  $n$  とし、固有値を  $\alpha_1, \dots, \alpha_n$  とすると、

$$\Phi_A(\lambda) = (\lambda - \alpha_1)(\lambda - \alpha_2) \cdots (\lambda - \alpha_n)$$

である。ここで、任意の正方行列  $S$  は、対角要素に  $S$  の固有値を持つ三角行列に相似であるから、これに基づいて適当な正方行列  $P$  をとると、

$$P^{-1}AP = \begin{pmatrix} \alpha_1 & & * \\ & \ddots & \\ O & & \alpha_n \end{pmatrix}$$

となる。したがって、

$$\begin{aligned} & \Phi_A(P^{-1}AP) \\ &= (P^{-1}AP - \alpha_1 E) \cdots (P^{-1}AP - \alpha_n E) \\ &= \begin{pmatrix} 0 & & * \\ \alpha_2 - \alpha_1 & & \\ & \ddots & \\ O & & \alpha_n - \alpha_1 \end{pmatrix} \begin{pmatrix} \alpha_1 - \alpha_2 & & * \\ 0 & \alpha_3 - \alpha_2 & \\ & \ddots & \\ O & & \alpha_n - \alpha_2 \end{pmatrix} \\ & \quad \cdots \begin{pmatrix} \alpha_1 - \alpha_n & & * \\ & \ddots & \\ O & & \alpha_{n-1} - \alpha_n \\ & & 0 \end{pmatrix} \\ &= O \end{aligned}$$

となる。このとき、

$$\Phi_A(P^{-1}AP) = P^{-1}\Phi_A(A)P$$

であり、したがって

$$\Phi_A(A) = O$$

が成り立つ。



---

 $\mathcal{A}\mathcal{M}\mathcal{S}$ - $\mathcal{L}\mathcal{A}\mathcal{T}\mathcal{E}\mathcal{X}$ ,  $\mathcal{L}\mathcal{A}\mathcal{M}\mathcal{S}$ - $\mathcal{T}\mathcal{E}\mathcal{X}$ 


---

概 略:  $\mathcal{A}\mathcal{M}\mathcal{S}$ - $\mathcal{T}\mathcal{E}\mathcal{X}$  は数式の表現には優れていますが, plain  $\mathcal{T}\mathcal{E}\mathcal{X}$  を母体としているだけに執筆者に plain  $\mathcal{T}\mathcal{E}\mathcal{X}$  を使いこなせるだけの知識が十分でなければならず, 執筆上の負担がかかることも一面の事実でした。そこで,  $\mathcal{A}\mathcal{M}\mathcal{S}$ - $\mathcal{T}\mathcal{E}\mathcal{X}$  と  $\mathcal{L}\mathcal{A}\mathcal{T}\mathcal{E}\mathcal{X}$  を統合する試みがなされてきました。

$\mathcal{A}\mathcal{M}\mathcal{S}$ - $\mathcal{L}\mathcal{A}\mathcal{T}\mathcal{E}\mathcal{X}$  はその試みののひとつで,  $\mathcal{L}\mathcal{A}\mathcal{T}\mathcal{E}\mathcal{X}$  を母体として  $\mathcal{A}\mathcal{M}\mathcal{S}$ - $\mathcal{T}\mathcal{E}\mathcal{X}$  の数理表現のスタイルを取り入れたものです。1987年に AMS の  $\mathcal{T}\mathcal{E}\mathcal{X}$  技術スタッフ Michael Downes 氏を顧問とし, Romesh Kumar 氏・Frank Mittelbach 氏・Rainer Schoöpf 氏らによって作成されました。

一方の  $\mathcal{L}\mathcal{A}\mathcal{M}\mathcal{S}$ - $\mathcal{T}\mathcal{E}\mathcal{X}$  は,  $\mathcal{A}\mathcal{M}\mathcal{S}$ - $\mathcal{T}\mathcal{E}\mathcal{X}$  の作者である M. D. Spivak 氏が作成し,  $\mathcal{T}\mathcal{E}\mathcal{X}$ plorators 社から発表されたもので,  $\mathcal{A}\mathcal{M}\mathcal{S}$ - $\mathcal{T}\mathcal{E}\mathcal{X}$  に  $\mathcal{L}\mathcal{A}\mathcal{T}\mathcal{E}\mathcal{X}$  の便利な点を取り入れたものといえるでしょう。コマンド体系などに  $\mathcal{A}\mathcal{M}\mathcal{S}$ - $\mathcal{T}\mathcal{E}\mathcal{X}$  のベースになった plain  $\mathcal{T}\mathcal{E}\mathcal{X}$  の影響が色濃く表れています。

所 在: nifty / flabo / lib11 / 39 : AMSLATEX.Lzh  
 nifty / flabo / lib11 / 44 : JAMSPAT.LZH  
 nifty / flabo / lib11 / 41 : AMSfonts.Lzh  
 pcvan / sscience / osl1 / 247 : LAMSTEX.LZH

使 用 法:  $\mathcal{A}\mathcal{M}\mathcal{S}$ - $\mathcal{L}\mathcal{A}\mathcal{T}\mathcal{E}\mathcal{X}$  の場合, まず `jlfonts2.tex` にかえて `lfonts.new` を読み込んだ `jlatex.fmt` ファイルを作成し, さらにソース中で,

```
\documentstyle[amstex]{jarticle}
```

あるいは

```
\documentstyle{amsart}
```

などのようにします。

$\mathcal{L}\mathcal{A}\mathcal{M}\mathcal{S}$ - $\mathcal{T}\mathcal{E}\mathcal{X}$  の場合は, 以下のようにして `fmt` ファイルを得ます。

```
A> initex.x \input jplain \input amstexl.tex \input
lamstex.tex \dump
```

マニュアル: 『 $\mathcal{A}\mathcal{M}\mathcal{S}$ - $\mathcal{L}\mathcal{A}\mathcal{T}\mathcal{E}\mathcal{X}$  Ver1.1 — User's Guide』

AMS Technical Support Group 著

American Mathematical Society 1986

『 $\mathcal{L}\mathcal{A}\mathcal{M}\mathcal{S}$ - $\mathcal{T}\mathcal{E}\mathcal{X}$  — The Synthesis』

M. D. Spivak 著  $\mathcal{T}\mathcal{E}\mathcal{X}$ plorators Corp. 1990

解 説 書: 『今度こそ  $\mathcal{A}\mathcal{M}\mathcal{S}$ - $\mathcal{L}\mathcal{A}\mathcal{T}\mathcal{E}\mathcal{X}$ 』

野寺隆志 著 共立出版 1991

Typeset by  $\mathcal{A}\mathcal{M}\mathcal{S}$ - $\mathcal{L}\mathcal{A}\mathcal{T}\mathcal{E}\mathcal{X}$ <sup>1</sup>

Question :  $f(x)$  は  $[a, b]$  で積分可能、 $g(y)$  は  $[c, d]$  で積分可能とすると、  
 $f(x)g(y)$  は  $D = [a, b] \times [c, d]$  で積分可能で、

$$\iint_D f(x)g(y) dx dy = \int_a^b f(x) dx \int_c^d g(y) dy$$

が成立することを証明せよ。

$D$  の任意の分割  $\Delta$  を

$$\Delta_1 : a = x_0 < x_1 < \cdots < x_{n-1} < x_n = b$$

$$\Delta_2 : c = y_0 < y_1 < \cdots < y_{m-1} < y_m = d$$

また、

$$h(x, y) = f(x)g(y)$$

$$x_{i-1} \leq \xi_i \leq x_i \quad (i = 1, \dots, n)$$

$$y_{j-1} \leq \eta_j \leq y_j \quad (j = 1, \dots, m)$$

とすると、

$$\begin{aligned} & \lim_{|\Delta| \rightarrow 0} \sum_{i=1}^n \sum_{j=1}^m h(\xi_i, \eta_j)(x_i - x_{i-1})(y_j - y_{j-1}) \\ &= \lim_{|\Delta| \rightarrow 0} \sum_{i=1}^n f(\xi_i)(x_i - x_{i-1}) \sum_{j=1}^m g(\eta_j)(y_j - y_{j-1}) \\ &= \lim_{|\Delta_1| \rightarrow 0} \sum_{i=1}^n f(\xi_i)(x_i - x_{i-1}) \lim_{|\Delta_2| \rightarrow 0} \sum_{j=1}^m g(\eta_j)(y_j - y_{j-1}) \\ &= \int_a^b f(x) dx \int_c^d g(y) dy \end{aligned}$$

よって、 $h$  は  $D$  で積分可能となり、以下の式が成立する。

$$\iint_D f(x)g(y) dx dy = \iint_D h(x, y) dx dy = \int_a^b f(x) dx \int_c^d g(y) dy$$

<sup>1</sup>問題および解法は、『別冊数学セミナー・現代応用数学の基礎 / 微分積分』(日本評論社 1993)を参考にしました。

Chem-TeX, X<sub>Y</sub>MT<sub>E</sub>X

概 略： 化学式を表現するパッケージとしては、Roswitha T. Haas 氏・Kevin C. O'Kane 氏によって作成された `chemtex.sty` や、Roswitha Haas 氏・Linda J. Hutchison 氏らによって作成され、構造体の表現を定義した L<sup>A</sup>T<sub>E</sub>X 用のマクロパッケージ `UTILS.TEX` が有名です。

しかし、このほかにも plain TeX 用に Michael Ramek 氏が作成した `chemstruct.sty`、化学反応式を書くことができるという Donald Arseneau 氏の `dchem.sty` などがあります。しかしながら、どれも実用的に利用する場合には不十分な点があるようです。最近、「解説書」の箇所で示す『化学者と生化学者のための L<sup>A</sup>T<sub>E</sub>X』の著者、藤田眞作氏によって、広範囲にわたる化合物の構造式を記述することができる L<sup>A</sup>T<sub>E</sub>X 用のマクロパッケージ X<sub>Y</sub>MT<sub>E</sub>X が発表されました。このパッケージは記述が簡単で、自由度も高いものになっています。

所 在： `pcvan / fprint / forum5 / 1551-1553, 1567, 1555-1556`  
`: chemtex.lzh`

`nifty / fsharp3 / lib9 / 21 : chemMacr.LZH`

`nifty / fsnote / lib4 / 82 : STYL_ADD.LZH`

`nifty / fsnote / lib4 / 74 : STYL_D.LZH`

`nifty / fprint / lib7 / 201 : xymtex.lzh`

使 用 法： `UTILS.TEX` の場合、以下のようにすると `fmt` ファイルを得られます。

```
A> initex.x \input jlplain \input ch_macros \dump
```

X<sub>Y</sub>MT<sub>E</sub>X については、パッケージ内のマニュアルを参照してください。

マニュアル：『X<sub>Y</sub>MT<sub>E</sub>X 例文集』

藤田眞作 著 1994

`nifty / fprint / lib7 / 202 : xymtexj.lzh`

『X<sub>Y</sub>MT<sub>E</sub>X 入門』

藤田眞作 著 1994

`nifty / fprint / lib7 / 204 : xymtexi.lzh`

解 説 書：『化学者と生化学者のための L<sup>A</sup>T<sub>E</sub>X パソコンによる論文作成の手引』<sup>1)</sup>

藤田眞作 著 東京化学同人 1993

1) これは Chem-TeX や X<sub>Y</sub>MT<sub>E</sub>X を解説した本ではありませんが、化学式の処理をするうえでは参考になるかもしれません。また、X<sub>Y</sub>MT<sub>E</sub>X で利用できるマクロ集が、添付されているディスクに含まれているようです。

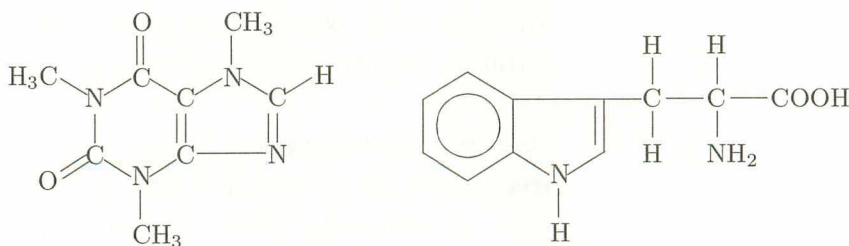


## X<sub>Y</sub>MT<sub>E</sub>X による化学式の表現

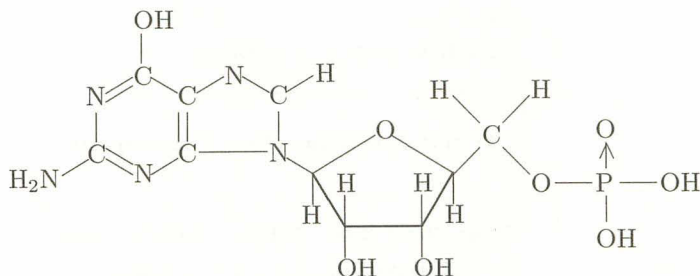
この出力サンプルは X<sub>Y</sub>MT<sub>E</sub>X によって書かれています。

以下の構造式、左側に示したのが皆さんも御存じのカフェインの構造式です。カフェインは、茶葉に 3% 程度、コーヒー豆に 0.7% 程度含まれ、興奮作用や利尿作用を持つことで知られていますね。

右側に示した構造式は、人間が体内で合成できないアミノ酸 (必須アミノ酸) であるトリプトファンです。



次に示すのは、酢酸構成単位であるヌクレオチド構造をもつシイタケの旨味成分 5'-GMP です。この構造式だけを見て「味」が思い浮かぶ方も、世の中にはいらっしゃるのでしょうか。



この構造式は以下のようなソースによって記述されています。

```
\begin{picture}(2500,800)(0,50)
\put(0,0){\nonaheterovi[egj]{1==N;2==C;3==N;4==N;5==C;6==N;7==C;8==C;%
9==C}{2==H;3==;5==H$}{_2}$N;7==OH}}
\put(800,-140){\furanose{1Sa==H;2Sa==OH;2Sb==H;3Sa==OH;3Sb==H;4Sa==H}}
\put(1422,100){\square{0==C;1==H;2==;3==;4==H}}
\put(1605,-80){\tetrahedral{0==O;4==}}
\put(1847,-80){\tetrahedral{0==P;1==O;3==OH;4==OH}}
\put(2117,310){\makebox(0,0)[lb]{\raisebox{0pt}{\shortstack[l]{\wedge$}}}}}
\end{picture}
```

## MuTeX

概 略： 楽譜作成用のマクロパッケージです。

ドイツにある Rheinische Friedrich-Wilhelms 大学の Andrea Steinbach 氏と Angelika Schofer 氏によって作成されました。2 声以下の 1 段譜を扱うことができます。また、五線譜の下に歌詞を書いたり、音符や五線譜の上などにさまざまな種類の発想記号をつけることができますが、複譜表や変則的なサイズの音符などの扱いはできないようです。

MuTeX は、後述の MusicTeX とは異なり、楽譜整形処理を自動的に行いますので、楽譜の作成は容易です。しかし反面、ユーザの自由度は (マクロに手を入れないかぎり) 少ないものになっています。

MS-DOS で動作する MuTeX 用のフィルタプログラムとして、渡邊純一 (なべじゅん) 氏の OMP<sup>1)</sup> があります。MS-DOS ユーザには、この OMP の使い勝手のよさのために、MusicTeX ではなく、MuTeX を使用するという人が大勢いるようです。

所 在： nifty / fsharp3 / lib9 / 20 : MUTEX.LZH

使 用 法： plain TeX ベース、LaTeX ベースのどちらでも使用できるようですが、LaTeX をベースにする場合には BigTeX を用意したほうがいいでしょう。

plain TeX をベースとした場合、

```
A> initex.x \input plain \input mplain \dump
```

で fmt ファイルができます。フォントの作成等、詳しくは「解説書」の箇所です挙げるドキュメントを参照してください。

解 説 書： 『MuTeX User's Guide (Version 1.1)』<sup>2)</sup>

François Jalbert 著 / 渡邊純一 編著 1989

nifty / fprint / lib7 / 187 : MUTEXMAN.LZH

1) 氏のご好意により、OMP のソースを、右ページに掲載されているサンプル出力のソースファイルとともにアーカイブに収録しています。

2) インストール関係などが渡辺氏によって加筆されており、逐語訳と区別する意味から渡辺氏自身「日本語版と言った方がよいのかも知れ」ないとしています。

MTXを使ったサンプル

製作：渡邊純一

Suite II from Six Suites for Violincello Solo

Menuet II

J.S.Bach

tr  
p  
mf  
(tr)  
mf  
p  
mf  
p  
f  
tr



---

MusicT<sub>E</sub>X

---

概 略： 楽譜作成用のマクロパッケージです。

Daniel Taupin 氏によって作成されたもので、オーケストラ譜のように複数パートからなる楽曲の楽譜を作成するためのマクロパッケージです。ただ、MusicT<sub>E</sub>X では、段組・音符・和音・連桁記号・スラー・装飾記号などを指定されたように処理するだけで、楽譜作成上の整形処理を自動的に行うことはしません。それだけに利用者のさまざまな要求に応えることができます。

このような自動整形を行いたい場合には、別に支援マクロや支援ツールを使用することになります。

そうした試みとして、MusicT<sub>E</sub>X の機能を拡張するマクロパッケージ MusicExp (Joker 氏による拡張フォントを含む) が YODA 氏によって、MML<sup>1)</sup> ライクな記述を MusicT<sub>E</sub>X 形式に変換するプリプロセッサプログラム<sup>2)</sup> がごんきち!氏によって制作されています。このほかにも、MS-DOS 用のプリプロセッサプログラムではありますが、MIDI ファイルから MusicT<sub>E</sub>X のソースを得る MIDI2T<sub>E</sub>X などもあるようです。

所 在： nifty / fprint / lib7 / 227 : MUSICTEX.ZIP  
nifty / fprint / lib7 / 224 : MusicExp.Lzh  
nifty / fprint / lib7 / 174 : MIDI2TEX.LZH  
nifty / fprint / lib7 / 175 : M2T\_DIFF.LZH  
nifty / fprint / lib7 / 190 : M2T\_DIF2.LZH

使 用 法： MusicT<sub>E</sub>X は、plain T<sub>E</sub>X ベース、L<sup>A</sup>T<sub>E</sub>X ベースのどちらでも使用できますが、L<sup>A</sup>T<sub>E</sub>X をベースにする場合には BigT<sub>E</sub>X を用意したほうがよいでしょう。

plain T<sub>E</sub>X をベースとした場合、

```
A> initex.x \input plain \input musicnft \input
musictex \dump
```

とすることで MusicT<sub>E</sub>X の fmt ファイルができます。フォントの作成等、詳しくは「マニュアル」の箇所に挙げたドキュメントを参照してください。

マニュアル： 『MusicT<sub>E</sub>X T<sub>E</sub>X による複数パート用の楽譜の作成 Version 5.01』<sup>3)</sup>

D. Taupin 著 / YODA 編著 1994

nifty / fprint / lib7 / 218 : MusicJdc.LZH

1)Music Macro Language。

2)MML2MusicTeX (仮称)  
は、本書初版刊行時点  
('94 年 7 月現在) において正式発表に至っておりません。

3) 原文は「所在」の箇所で示した MUSICTEX.ZIP に同封されています。YODA 氏の手によって、かなり加筆修正されており、逐語訳と区別する意味から「日本語版」とされています。

# Wedding March

from "A Midsummer Night's Dream"

*F. Mendelssohn (1809–1847)*

Arranged for men's voices by J. Hayashi  
Transcribed for **MusicT<sub>E</sub>X** & **MusicExp** by M. Moriwaki  
(YODA:HFD00546@niftyserve.or.jp)

8

3

3

3

3

3

3

3

3

8

8

8

8

to Coda

友人の結婚式用に作成した楽譜より抜粋 (YODA)

---

**multicol.sty**


---

**概 略:** Frank Mittelbach 氏が作成したスタイルオプションファイルで、 $\text{\LaTeX}$  使用時に自由な多段組による出力を可能にします。このスタイルオプションファイルを使用すれば、同一ページ内であっても段組を自在に変更できるうえ、`\section` のような見出し部分だけを特別に 1 段組にすることも可能になっています。

また、多段組をした際の最終部分は、各段の行数がほぼ同じになるように揃えてくれますから、体裁も美しいものに仕上がります。

**所 在:** Ver.1.2a は添付ディスク 8 に含まれています。

nifty / flabo / lib11 / 45 : MLTCOL13.LZH

nifty / fsnote / lib4 / 35 : DOC-1\_7K.LZH

nifty / fsnote / lib4 / 36 : MCOL1\_4M.LZH

**使 用 法:** 「所在」で挙げたすべてのアーカイブを、新規の (日付が新しい) ファイルが残るように展開して、 $\text{\LaTeX}$  で `install.mz0`, `install.mz1` を順番に処理すれば `multicol.sty` ができています。なお、 $\text{\LaTeX}$  で `install.*` を処理する途中で「既にファイルが存在する」旨のメッセージを返してきたときには、上書きを指示してください。スタイルオプションで、

`\documentstyle[multicol]{jarticle}`

のようにします。



# multicol.sty による多段組

## 1 multicol.sty によって実現できること

### 1.1 twocolumn.sty

通常の  $\text{\LaTeX}$  にも二段組をおこなう `twocolumn.sty` の `\twocolumn` コマンドがありますが、このコマンドは設定すると改ページをおこなってしまいますから、同一のページで段組みを変えることができません。このため、たとえば二段組の文章の途中に、段組みの横二段に相当するような幅の広い図表を挿入したい場合には、あきらめて一段組にするか、`minipage` 環境で擬似的に二段組を作

るしかありません。

### 1.2 multicol.s 環境

しかし、`multicol.sty` によって拡張される `multicol.s` 環境を使用すれば、 $\text{\LaTeX}$  において自由な多段組を実現することができます。たとえば、本 section のタイトルは一段組ですし、subsection 以下の文章部分は二段組になっているのです。

## 2 自由自在な段組み

先に多段組といいました。`twocolumn.sty` では二段組までしかできませんが、`multicol.sty` では組数に制限がありませんから、状況に応じて使い分けることができます。

しかも、多段組をはじめる先頭部分については、簡単な記述で一段抜きにすることが可能です。たとえば、このサンプルにおける各セクションタイトルの一段抜きは、この

機能を使用して実現しています。以下に示すように [ ] で囲うだけ。簡単ですね。このような組版をすれば、セクションタイトルを一段抜きにするだけでなく、セクシ

ョンの冒頭部分 (この例では「先に～できます。」) を、セクションの導入部分として強調することもできるかもしれませんね?

```
\begin{multicols}{3}[\section{自由自在な段組み}]
先に多段組といいました。{\tt twocolumn.sty} では二段組までしかできませんが、
{\tt multicol.sty} では組数に制限がないのです。
```

しかも、多段組をはじめる先頭部分については、簡単な記述で一段抜き…

## 3 It is “multicol.sty” !!

以上に示したように、`multicol.sty` は  $\text{\LaTeX}$  で多段組をおこなう場合にたいへん便利なスタイルファイルです。その便利さゆえ

に愛用者も多く、 $\text{\LaTeX}$  のスタイルファイルの中で、世界的にもっともポピュラーになったもののひとつとして数えることができます。

もし  $\text{\LaTeX}$  で多段組をする必要が生じた場合には、ぜひ、このスタイルファイル “`multicol.sty`” を入手しましょう。きっと役に立つはずです。

---

Tarticle.sty

---

概 略： 川村信郎氏が、自作の `tate.sty` と吉澤康介氏の `tatex.sty` をもとに、アスキーの `jarticle.sty` および `jart10.sty` のパラメータを縦組用に変更したものです。

- `\Large` や `\small` のような文字サイズの指定
- `\bf` のような文字種の指定
- `\section` などの漢数字による出力
- ヘッダおよびフッタの上下への配置
- `minipage` 環境内での横書き使用

などが実現されています。簡単な文章であれば、ほとんどそのまま縦書きにすることができ、数式や図表もあまり複雑でないものについては満足のできる仕上がりが期待できます。

同氏は、葉書の宛名と本文を縦書き出力するための `tcard.sty` も作成されています。

なお、アスキーおよびインプレスから 縦組対応版日本語 T<sub>E</sub>X “pT<sub>E</sub>X” 用の縦組対応マクロパッケージ (縦組対応 L<sup>A</sup>T<sub>E</sub>X のマクロ集 `plplain.tex`) がリリースされました。これにともなって、縦組用のスタイルファイル `tarticle.sty`, `treport.sty`, `tbook.sty` も公開されています<sup>1)</sup>。これらは、前から順に、従来の日本語 T<sub>E</sub>X における `jarticle.sty`, `jreport.sty`, `jbook.sty` に対応する縦組用スタイルファイルです。

1) これらの縦組対応マクロは添付ディスクでインストールされるファイルに含まれています。ただし、本書に添付されているデバイスドライバは、本書の執筆時点で、縦書きの際に漢字を時計周りの方向に 90 度回転させる機能をサポートしていないので、この機能を使用する出力はできません。

所 在： `nifty / fprint / lib7 / 139 : TARTIC02.LZH`  
`nifty / fprint / lib7 / 124 : TATEX01.LZH`  
`nifty / fprint / lib4 / 123 : TATE004.LZH`  
`nifty / fprint / lib7 / 143 : TCARD06.LZH`

使 用 法： 本書に添付されている pT<sub>E</sub>X の縦書きマクロを使用するためには、`plplain.tex` を `initex.x` にかけて作成した `fmt` ファイルが必要です。そのうえで、文書スタイルに、

`\documentstyle[b5j]{tarticle}`

のようにします。

マニュアル： 『T<sub>E</sub>X の出版への応用 — 縦組機能の組み込み —』

濱野尚人・田村 明史・倉沢 良一 著 アスキー 1990

『入門基本セット 縦組対応版パーソナル日本語 T<sub>E</sub>X』

インプレス・ラボ 監修 アスキー書籍編集部 編

アスキー 1994

解 説 書： 『日本語 L<sup>A</sup>T<sub>E</sub>X 定番スタイルファイル集 3』

鷺谷好輝 著 インプレス 1994

# Tarticle.styによる縦書の世界

Tarticle.sty によって縦組にするからといっても、ソースに特別の仕掛けをするわけではありません。スタイルファイルに Tarticle を指定するだけです。

数式にせよ図表にせよ、通常の横書  $\text{\LaTeX}$  とまったく同じように記述することができます。

$$A = \int_a^b f(x) dx = \lim_{n \rightarrow \infty} \sum_{i=1}^n f(x_i) \Delta x_i$$

これは、 $\text{\pTeX}$  本体が縦組に対応しているからに他なりません。Tarticle.sty は、 $\text{\pTeX}$  の縦組の機能を手軽に引き出すためのスタイルファイルなのです。本書に添付されているディスクが構築する  $\text{\TeX}$  システムにも、一連の縦組マクロが含まれています。したがって、あなたも縦組の世界を体感することができます。

Tarticle.sty は、 $\text{\pTeX}$  が構想している縦組と横組の混在文書に対応しています。このサンプルがそうであるように、単純に縦組の文章の途中に横組の文章を挿入するだけではなく、それ以上の処理が可能になっているのです。

詳しい説明は『縦組対応パーソナル日本語  $\text{\TeX}$ 』（アスキー刊、一九九四年）の「縦組リファレンスガイド」にありますので、参照してください。

| 《 $\text{\pTeX}$ における相対単位 》 |                       |
|-----------------------------|-----------------------|
| em                          | 現在有効な欧文フォントの大文字 M の幅  |
| ex                          | 現在有効な欧文フォントの小文字 x の高さ |
| zw                          | 現在有効な日本語フォントの文字の幅     |
| zh                          | 現在有効な日本語フォントの文字の高さ    |

図表などは、とくに構造上の理由から横組で出力したい場合があるでしょう。そのときには minipage 環境で確保した領域に横組で出力してください。また、今回は紙面の都合で掲載できませんでした。が、アルファベットや数字も縦にして並べるといって「純粋な縦書き」の組版をすることもできますので、興味のある人は試してみてください。

| 《 $\text{\pTeX}$ における絶対単位 》 |                                |
|-----------------------------|--------------------------------|
| pt                          | ポイント                           |
| sp                          | スケールポイント                       |
| mm                          | ミリメートル                         |
| cm                          | センチメートル                        |
| bp                          | ビッグポイント                        |
| in                          | インチ                            |
| pc                          | パイカ                            |
| dd                          | デイドーポイント                       |
| cc                          | シセロ                            |
| H                           | 級                              |
| q                           | 歯                              |
| 1pt                         | = 0.3515mm = 65536sp           |
| 1sp                         | = 0.5362 × 10 <sup>-5</sup> mm |
| 1mm                         | = 0.1cm = 2.834pt              |
| 1cm                         | = 10mm = 28.34pt               |
| 72bp                        | = 1in                          |
| 1in                         | = 25.4mm = 72.27pt             |
| 1pc                         | = 12pt = 4.218mm               |
| 1157dd                      | = 1238pt                       |
| 1cc                         | = 1dd                          |
| 1H                          | = 0.25mm                       |
| 1Q                          | = 1H                           |



---

## PiCTeX

---

概 略： plain TeX や LaTeX に図形処理機能を追加するためのマクロパッケージです。シカゴ大学の Michael J. Wichura 氏によって開発されました。

plain TeX は図を描く際には無力ですし、LaTeX の図形処理は特殊フォントによって実現されるために、任意の大きさの円や傾きの線を描くことができるというわけではありません。そこで、点(ピリオド)の集合で図形を表現する拡張マクロとして PiCTeX が作成されました。PiCTeX では、円や楕円、直線、二次補間曲線などによって自由な図形描写が可能です。また、座標回転や網かけなどの機能も持っています。

PiCTeX を使用する際には、図形を点の集合で表現するために情報量が多くなります。したがって、PiCTeX を使用するには BigTeX が必要となります<sup>1)</sup>。

1) 図形が複雑である場合には BigTeX を使用しても、なお容量不足になることがありますから注意してください。また、PiCTeX は処理にかなりの時間がかかります。何のメッセージも出さずに黙りこんでしまっても、心配する必要はありません。

所 在： 添付ディスクでインストールされるファイルに含まれています。  
使 用 法： plain TeX ベースの場合には、

```
A> initex.x \input jplain \input prepictex \input
pictex \input latexpicobjs \dump
```

LaTeX での使用であれば、

```
A> initex.x \input jlplain \input prepictex \input
pictex \input postpictex \dump
```

として PiCTeX の `fmt` ファイルを得ることができます。

LaTeX の場合、ソース中のスタイルオプションで以下のような設定をすることでマクロの利用が可能になります。

```
\documentstyle[piclatex]{jarticle}
```

マニュアル： 『The PiCTeX Manual』  
M.J.Wichura 著 TeX User's Group

解 説 書： 『PiCTeX 解析資料』  
河浦淳一 著 1990  
nifty / fsharp3 / lib9 / 36 : PiTeXman.Lzh

『LaTeX 美文書作成入門』  
奥村晴彦 著 技術評論社 1991

概 略: epic.sty, eepic.sty, ecleepic.sty と  $\text{\LaTeX}$  の picture 環境を拡張するためのマクロ集です。 $\text{\LaTeX}$  の picture 環境には、以下のような制限があります。

- (1) 始点と終点を指定して直線を描くことができない。
- (2) 任意の傾きを持つ直線を描くことができない。
- (3) 点線や破線を描くことができない。
- (4) 短い直線を描くことができない。
- (5) 任意の直径を持つ円・楕円・円弧を描くことができない。
- (6) スプライン曲線を描くことができない。
- (7) 指定の領域をタイリングすることができない。
- (8) 図形の回転等ができない。

epic.sty は Sunil Podar 氏が作成したもので、(1)~(3) を組版の範囲内 (マクロと METAFONT だけで解決するため、出力するデバイスドライバに依存しない) で解消するマクロです。

eepic.sty は Conrad Kwok 氏が作成したもので、(4)~(8) を解決します。ただし、後述する Tpic specials (p.255) によって解決を図るために、これに対応するデバイスドライバでなければ出力できません<sup>1)</sup>。

ecleepic.sty は磯崎秀樹氏によるもので、組版の範囲内で (4) と (5) を解決します。

所 在: epic.sty は添付ディスクでインストールされるファイルに含まれています。

nifty / fprint / lib11 / 146 : ECLSTYLE.TAZ<sup>2)</sup>

nifty / fsnote / lib4 / 75 : STYL\_E.LZH<sup>3)</sup>

使 用 法: ソース中のスタイルオプションに以下のように設定することで、上の設定例から順に epic, eepic, ecleepic の使用が可能になります。

```
\documentstyle[epic]{jarticle}
```

```
\documentstyle[epic,eepic]{jarticle}
```

```
\documentstyle[epic,ecleepic]{jarticle}
```

eepic.sty および ecleepic.sty は epic.sty のマクロを再定義していますから、スタイルオプションでの指定は epic.sty のあとに記述しなければなりません。

マニュアル: 『 $\text{\TeX}$  by Tpic, A  $\text{\TeX}$ nician's Reference』

V.Eijkhout 著 Addison-Wesley 1992

解 説 書: 『 $\text{\LaTeX}$  自由自在』

磯崎秀樹 著 サイエンス社 1992

1) 本書に付属のプリントドライバは tpic specials に対応していますから出力可能ですが、他機種あるいはその他のプリントドライバで出力する際にはドライバプログラムに依存することになります。

2) アーカイブに含まれている epic.sty と eepic.sty は、オリジナルの epic.sty および eepic.sty の不具合を解消したものです。

3) ECLSTYLE.TAZ に含まれている不具合の解消版に加え、ドキュメントおよびオリジナルの epic.sty と eepic.sty が添付されています。

---

 epsf.sty, eclepsyf.sty
 

---

1) EPSF とは、カプセル化された POSTSCRIPT ファイルのことで、プログラムなどから POSTSCRIPT のデータを取り出しやすい形式になっています。

2) Tomas Rokicki 氏が作成した同名で同等機能のスタイルファイルも存在しています。本文中で述べた epsf.sty が名前を変えたのは、この同名ファイルが存在していたことに加え、NTT がリリースするスタイルファイルのファイル名に ecl の 3 文字を冠するという方針を決めたことによるようです。

3) 本書中の図の多くはこの方法で作成、挿入しています。また、GNUPLOT の出力サンプルも、この手法によって図を挿入しており、ソースには仁泉氏がコメントを加えていますので、参考にしてください。

概 略: epsf.sty および eclepsyf.sty は、 $\text{\LaTeX}$  に POSTSCRIPT の EPSF (Encapsulated POSTSCRIPT File)<sup>1)</sup> 形式で出力されたデータを取り込むためのスタイルオプションファイルです。

これらのスタイルオプションファイルを使用すれば、文書に取り込む EPSF データの大きさや縦横比率を  $\text{\LaTeX}$  側で任意に設定できるので、POSTSCRIPT のソースを変更する必要がありません。Trevor J. Darrell 氏が作成した psfig.tex をもとに、NTT の風間一洋氏と磯崎秀樹氏が改良を加えて epsf.sty<sup>2)</sup> が整えられ、さらに epsf.sty がバージョンアップして eclepsyf.sty になりました。

所 在: nifty / fprint / lib11 / 146 : ECLSTYLE.TAZ

使 用 法: ソース中のスタイルオプションに以下のように設定します。

```
\documentstyle[epsf]{jarticle}
```

```
\documentstyle[eclepsyf]{jarticle}
```

これらのスタイルオプションファイルを使用した場合にも、dvi ファイル自体には EPSF の情報は書き込まれていません。このとき、本書でサポートしているようなビットマップ展開型のデバイスドライバでは、実際に出力しようとしている EPSF のデータは空白で出力されます。したがって、一般的には、この空白部分に別途 POSTSCRIPT で EPSF を出力して貼り付ける<sup>3)</sup> か、dvi ファイルそのものを後述する dvi2ps (p.260) で処理して POSTSCRIPT 出力することになります。

マニュアル: 『“epsf.sty” マニュアル』

風間一洋 著 1990

nifty / fsharp3 / lib9 / 70 : epsf.lzh

解 説 書: 『 $\text{\LaTeX}$  自由自在』

磯崎秀樹 著 サイエンス社 1992



## Tpic, Gpic

概 略: Tpic は、Brian Kernighan 氏の Troff 用の図形プリプロセッサ pic をもとにして、Tim Morgan 氏が作成した T<sub>E</sub>X 用の図形プリプロセッサ、およびこれが出力する special コマンド<sup>1)</sup> (Tpic specials) のセットです。後述する xfig などが、Tpic のソース出力をサポートしています。

これに対して、GNU プロダクトである GNU roff と T<sub>E</sub>X 用の図形プリプロセッサ GNU pic、およびこれが出力する special コマンド (Gpic specials) のセットが Gpic です。Gpic では、一部の命令がオリジナルの pic や Tpic と異なる仕様になっているようです。

GNU pic は GNU roff のパッケージに含まれています。

X680x0 への移植は、吉野智典 (真里子) 氏によって行われました。移植時点で、X680x0 には画像のセーブプログラムである pic.x がすでに存在していたことから、UNIX オリジナルのファイル名に由来する pic.x ではなく、ropic.x にリネームされています。また、Gpic が含まれる GNU roff のパッケージには roff の出力を dvi 形式に変換するコンバータ grodvi.x も含まれますので、入手しておくに役に立つことがあるかもしれません。

所 在: nifty / fsharp3 / lib3 / 125 : groff.LZH<sup>2)</sup>

使 用 法: たとえば、GNU pic のソースファイルが sample である場合、以下のようにすると Gpic コードが標準出力に吐き出されますので、これをリダイレクトします。

```
A> ropic.x -t sample > sample.gpic
```

マニュアル: 『Tpic: T<sub>E</sub>X 用の pic』

Tim Morgan 著 / 中林歩 訳 1990

nifty / flabo / lib11 / 103 : jtpicman.lzh

『gpic 1.06 日本語マニュアル』

FSF 編 / 中林歩 訳 1992

nifty / flabo / lib11 / 166 : JGPICMAN.LZH

1) T<sub>E</sub>X の \special コマンドのことで、『何の処理もしない』という命令。このようにして出力された dvi ファイルのなかに埋め込まれた命令は、デバイスドライバによって解析・出力されます。したがって、このようなソースはデバイスドライバに依存することになります。

2) いささか古い (Ver.0.3。最新は筆者が知るかぎり Ver.1.06) ので、最新の Tpic 命令には対応していません。

---

xfig, transfig

---

概 略: xfig は、UNIX の X-Window 用の図形エディタです。

直線や円はもちろん、スプライン曲線・正多角形の描写、領域塗りつぶし・文字入力・コピー・回転など、描写および編集において必要となるおおよその機能は揃っています。そして、なによりエディット中の図形を T<sub>E</sub>X (P<sub>T</sub>CT<sub>E</sub>X, epic, eepic) や Tpic, PostScript などの言語体系に変換してファイル出力する機能を有することが特徴です。そしてこのとき、xfig で扱うことができる fig 形式のデータを、実際に他の言語体系に変換する作業を行うのが transfig パッケージに含まれる実行ファイル群です。

OTO 氏が移植された X680x0 版では、出力可能な形式として cut ファイル<sup>1)</sup> が加えられています。

オリジナルの xfig は、3 ボタンのマウスを前提にしているため、2 ボタンのマウスを使用する X680x0 版では両方のボタンを同時に押すことで 3 番目のボタンの機能を実現しています。

所 在: nifty / fsharp3 / lib9 / 115 : xfig2103.Lzh

nifty / fsharp3 / lib9 / 89 : transfig.Lzh

使 用 法: fontman.x を、パッケージに付属しているコンフィギュレーションファイル xfig.fm で常駐させ、その後、xfig を起動させます。

```
A> fontman.x xfig.fm
A> xfig.x -me -right
```

xfig 起動時に “-me” オプションを指定すれば、cm を単位として利用でき (初期設定では inch 単位)、“-right” オプションを指定すれば、機能を選択するメニューを描画ウィンドウの右側に持ってくることができます (初期設定では左側)。

解 説 書: 『An Introduction to X Window System 39』

(『UNIX MAGAZINE』1992.11)

中村真 著 アスキー

1)BEEPs 氏が『電腦倶楽部 Vol.13』(満開製作所)で発表したもので、モノクロ二次元のビットマップデータを圧縮したファイル形式を持っています。

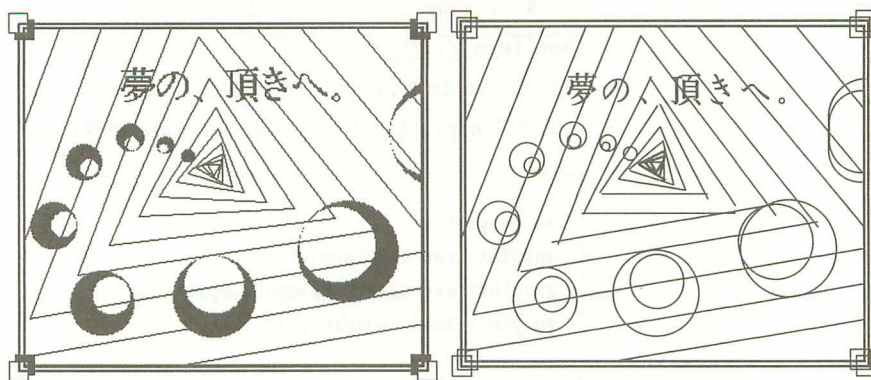
## Pictures in T<sub>E</sub>X

下のイラストは、彩人氏による「fairy T<sub>E</sub>X」です。データ形式は cut ファイル形式で、ここでは Ext 氏が作成した cutbox.sty によって出力しています。



次に示す図形データは、xfig を使用して作成しました。X680x0 版 xfig には、cut ファイルの出力をサポートしています。この機能を使用すれば、xfig で見ているままのイメージで出力することが可能です。下の左側がその例で、cutbox.sty で出力しています。

右側は、Transfig を利用した xfig の “Export” 機能によって、同じデータを L<sup>A</sup>T<sub>E</sub>X ソースの形式に変換し、得られたソースを少しだけ手直しをしたものです。



▲ xfig で作成したデータは、L<sup>A</sup>T<sub>E</sub>X や eepic.sty や Tpic、POSTSCRIPT などのデータ形式に変換することができます。ただし、変換したデータが必ずしも xfig で見たままの結果をもたらすとは限りません。なぜなら、変換する言語体系の機能的な限界などがあるためです。たとえば L<sup>A</sup>T<sub>E</sub>X の場合、eepic.sty の解説で示したような機能限界によって、以上のように領域の塗りつぶしなどが省略されていることがわかります。



---

GNU PLOT

---

1) 「GNU」とついてはいますが、いわゆる GNU プログラミングではありません。偶然の一致です。しかし、多くの人が「GNU」で検索をかけるためか、現在では GNU プログラムと同一の場所に置かれていることが多いようです。ただし、いっしょに置かれているだけで、あくまで GNU や FSF との関わりがないので、その扱いには注意が必要です。

2) 計算だけを行うコマンドとして、“print”が用意されています。

3) GNU PLOT が出力した eepic 形式のデータは、ときに TeX の capacity を越えてしまいます。その際には、山崎健 (融) 氏が作成したフィルタプログラム eepic2tpic によって、GNU PLOT が出力した eepic ソースを Tpic 形式に変換することで対処が可能です。

概 略: GNU PLOT<sup>1)</sup> は、関数のグラフを対話的にかいてくれるほか、複素数に対応した関数電卓<sup>2)</sup>としても利用することが可能なプログラムです。

Thomas Williams 氏・Colin Kelley 氏・David Kotz 氏らが作成しました。画面出力はもとより、LaTeX や eepic, POSTSCRIPT のソースも出力してくれます。

X680x0 版は、Ver.2.0 が ROM 男氏によって、Ver.3.0 が飛田衛氏によって、Ver.3.2 が仁泉大輔氏によって移植されました。仁泉氏の移植された Ver.3.2 X Rel.2 では、cut ファイル形式や fig 形式もサポートしています<sup>3)</sup>。

なお、この系譜とは別に、KEN 氏によって Ver.3.5 が移植されました。これは cut ファイル形式のサポートなどはないものの、オリジナルの GNU PLOT に対して、ソースレベルでかなり正確な移植がされています。

所 在: nifty / fsharp3 / lib9 / 104 : gnuplot32r1.Lzh  
nifty / fsharp3 / lib9 / 107 : gpl32r2.Lzh  
nifty / fsharp1 / mes18 / 408 : eepic2tpic.Lzh  
nifty / fsharp3 / lib9 / 292 : GPL35X1.LZH

使 用 法: たとえば、GNU PLOT のデータファイル sample.gpl に以下のような記述がある場合には、まず、この行をコメントアウト (行頭に “#” を付加) します (??? は任意の記述)。

```
set term ???
```

その後、eepic 形式のソースを得たいのであれば、次のようにすることで eepic 形式の出力 sample.eepic が得られます。

```
A> gnuplot.x
gnuplot> set term eepic
gnuplot> set output "sample.eepic"
gnuplot> load "sample.gpl"
```

このとき、set term eepic の eepic の部分を LaTeX にすれば LaTeX の、PostScript にすれば POSTSCRIPT の形式で出力することが可能です。

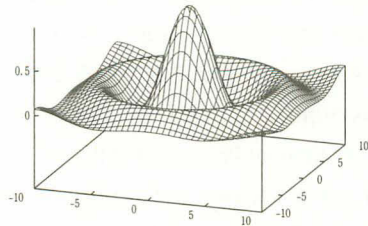
4) このマニュアルは GNU PLOT の Ver.3.2 用のものです。

マニュアル: 『GNU PLOT An Interactive Plotting Program Version 3.0』<sup>4)</sup>  
T. Williams・C. Kelley・D. Kotz・他著 / tamaru 訳  
nifty / fsharp3 / lib9 / 108 : gpl32doc.Lzh

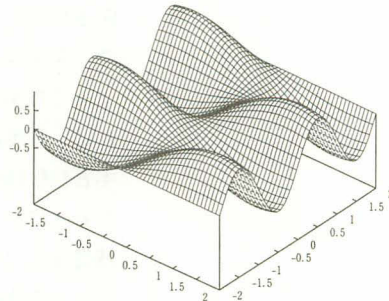
## GNUPLOT の出力サンプル

サンプル作成: 仁泉大輔 (NiftyServe:HGA02713)

cont.ps

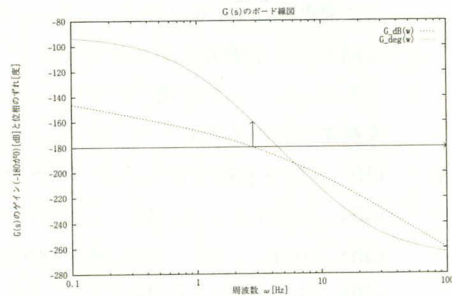


nami.ps



右は  $z = \sin r/r$  ( $r = x^2 + y^2$ )、左は  $z = \sin x \sin \pi y$  を表す図を、三次元プロットしてみました。

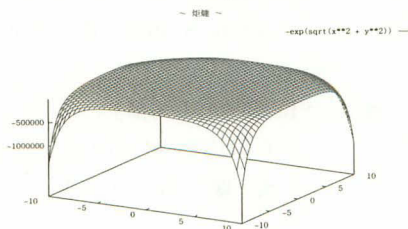
gs.ps



二次元平面の図で、 $G(j\omega) = 5/(j\omega(1 + 0.1j\omega)(1 + 0.5j\omega))$  のボード線図<sup>1</sup>を表す図です。これは、周波数を表す横軸を対数軸としていたり、媒介変数モードを使っていたりと実用的なサンプルといえるでしょう。

最後におまけです。この図は単に  $z = -\exp \sqrt{x^2 + y^2}$  のグラフを描画しただけです。:-)

kotatu.ps



## GHOSTSCRIPT, dvi2ps

概 略: GHOSTSCRIPT は、John Warnock 氏が設計したインタプリタ型のページ記述言語 PostScript と互換性のある GNU プロダクトです。Aladdin Enterprises の L. Peter Deutsch 氏によって作成されました。きわめて多種多様なプリンタに対して出力が可能で、非 PostScript プリンタにも対応しています。片山紀生氏によって日本語化パッケージが作成されているため、これを適用すれば、日本語 PostScript の出力も可能です。

本書初版刊行時点の X680x0 版は、日本語化された GHOSTSCRIPT をもとにして、井上康成 (K-ras) 氏が移植されています。しかし、残念なことに現在の公開版 X6.00 $\beta$  が対応しているプリンタはキヤノンの BJ-10v だけです。そのため、花井氏によって BJ-10v 用の出力データを吸い上げてエプソンの AP-850 用に変換するプログラムが開発されています。

dvi2ps は、TeX が出力した dvi 形式のファイルを PostScript 形式に変更するフィルタです。X680x0 版への移植は、GHOSTSCRIPT と同じく井上康成氏が行っています。本書がサポートしているようなビットマップ展開型のデバイスドライバではなく、命令体系変換型のデバイスドライバのひとつです。

所 在: nifty / fsharp3 / lib9 / 54 : GS23\_00A.LZH  
 nifty / fsharp1 / Mes16 / 773, 775-777 : gs00b\_bin.Lzh  
 nifty / funix / lib8 / 305, 306 : ghostscript-fonts-2.6.tar.Z  
 nifty / fsharp1 / Mes16 / 778 : gskfonts.Lzh  
 nifty / fsharp1 / Mes16 / 779 : gskdemos.Lzh  
 nifty / fsharp3 / lib9 / 65 : dvi2ps17.Lzh  
 nifty / fsharp1 / Mes16 / 848 : dvi2ps01.lzh  
 nifty / fsharp1 / Mes16 / 959 : bj2ap.Lzh

使 用 法: アーカイブ内の Readme.x68 に詳しく示されています。

マニュアル: 『PostScript リファレンスマニュアル 第二版』

Adobe Systems 著 / アドビシステムズ・ジャパン 監訳

アスキー 1991

解 説 書: 『Inside DVI→PS』

(『UNIX MAGAZINE』1993.12～)

高山健三 著 アスキー



概 略: lips3dvi は、T<sub>E</sub>X が出力した dvi 形式のファイルをキヤノンのレーザプリンタ制御言語である LIPS3 形式に変換するプログラムで、アスキーが著作権を有します。

このプリンタドライバを使うことで、LIPS3 を制御言語とするプリンタで T<sub>E</sub>X の出力が可能になります。また、エプソンの ESC/Page に対応したプログラム escpdvi も同アーカイブに含まれています。

本書がサポートしているようなビットマップ展開型のデバイスドライバではなく、命令体系変換型のデバイスドライバですから、和文フォントとしてプリンタ内蔵のスケラブルフォントと、JXL4 フォーマットの大日本フォントを切り替えて使用することが可能です。

X680x0 へのソース差分が、野分氏によって作成されています。

所 在: nifty / flabo / lib4 / 214: lips3dvi2.09a.tar.Z<sup>1)</sup>  
nifty / fsharp1 / mes20 / 249: lips3x68.lzh<sup>2)</sup>

使 用 法: 野分氏が作成している差分キットを適用したあと、GCC と LIBC でコンパイルして実行ファイルを作成します。  
実際の使用についてはマニュアルをご覧ください。

1) ソースファイルです。添付ディスク 8 に含まれています。

2) ドキュメントに加筆されているものが添付ディスク 8 に含まれています。

---

BiBTeX

---

概 略： Oren Patashnik 氏が作成した参考文献処理用の支援プログラム。  
構築した文献データベースとの相互参照を自動化してくれます。  
X680x0 には、アスキーが日本語化したものが OTO 氏によって  
移植されました。

所 在： nifty / fsharp3 / lib9 / 58 : jbibtex.Lzh  
nifty / fsharp3 / lib9 / 59 : bibtexbs.Lzh

使 用 法： 文献スタイルファイル \*.bst を環境変数 TEXINPUTS が示すディ  
レクトリに収め、文献データベースファイル \*.bib を環境変数  
BIBINPUTS が示すディレクトリに収めればインストールは完了  
です。

利用する際には、文献データベースファイル (たとえば  
sample.bib) を用意して、ソースの \end{document} の手前に次  
のような記述を加えます。

```
\bibliographystyle{jplain} \bibliography{sample}
```

以上の設定を行うことで文献スタイルファイル jplain.bst に定  
められた形式の文献リストを作成します。処理の手順が、

LaTeX 処理 ⇒ BiBTeX 処理 ⇒ LaTeX 処理 ⇒ LaTeX 処理  
のようになることに注意しましょう。

解 説 書： 『LaTeX 美文書作成入門』

奥村晴彦 著 技術評論社 1991

『化学者と生化学者のための LaTeX パソコンによる論文作成の手  
引』

藤田眞作 著 東京化学同人 1993

---

## Makeindex

---

概 略： L<sup>A</sup>T<sub>E</sub>X 用の索引作成支援プログラムです。

X680x0 版は森村健司 (Curios) 氏によって作成されました。アスキーの日本語 T<sub>E</sub>X に添付されていた日本語版 *Makeindex* と同様に使用できます。

なお、ファイル中の文字列を辞書配列順にソートするプログラム *jjsort.x* もアーカイブに含まれており、これと *Makeindex* の本体 *makeind.x* とを連携させるためのバッチファイルも添付されているので、索引の作成はほぼ自動で行うことが可能になっています。

所 在： nifty / fsharp3 / lib9 / 48 : *indexer.lzh*

nifty / fsharp1 / mes16 / 487 : *makeind.lzh*

nifty / fsharp1 / mes16 / 491 : *makeind.lzh*<sup>1)</sup>

使 用 法： アーカイブ内のドキュメント *readme.doc* に詳細が示されています。まず、L<sup>A</sup>T<sub>E</sub>X のソースファイルのスタイルオプションに *makeidx* を指定します。次に、プリアンプルに以下の記述を、

```
\makeindex
```

さらに *document* 環境内の索引を出力したい位置に以下の記述を加え、L<sup>A</sup>T<sub>E</sub>X で処理します。

```
\printindex
```

一通り L<sup>A</sup>T<sub>E</sub>X 処理を行ってページ変動が起こらなくなったら、

Makeindex 処理 ⇒ L<sup>A</sup>T<sub>E</sub>X 処理

を行います。

*Makeindex* の処理は、*Makeindex* のアーカイブに添付されているバッチファイル *makeind.bat* を使用する場合、以下のように実行してください。ここでは、仮に *sample.tex* の索引を作成するものとします。

```
A> makeind sample
```

1) これらの上位バージョンが *MakeIndex.Lzh* の名で添付ディスク 8 に含まれています。

解 説 書： 『L<sup>A</sup>T<sub>E</sub>X 美文書作成入門』

奥村晴彦 著 技術評論社 1991



---

plain2

---

概 略： 通常のテキスト (プレーンテキスト) ファイルを、 $\text{\LaTeX}$  ないし  $\text{\roff}$  のソースに自動生成して標準出力に吐き出すプログラムです。NEC の内田昭宏氏によって開発されたものが、湯浅夏樹氏によって X680x0 に移植されています。

plain2 のアーカイブに添付されているドキュメントファイル `readme.j` に、plain2 の変換性質が詳しく述べられています。ここで示されている一定の記述方式に則って書かれたプレーンテキストに対してであれば、相当程度、 $\text{\LaTeX}$  ソースの自動生成効率を発揮してくれます。うまく利用すればソース入力の際の手間を激減してくれるでしょう。

また、図表などを作成する際、罫線などを利用してプレーンテキストで記述し、plain2 で  $\text{\LaTeX}$  のソースファイルに変換して取り込むという使い方も考えられます。

1) 「機能的にはこちらのバージョンの方が上である」という移植者、湯浅夏樹氏の助言をいただき、バージョンは古いですが、こちらのアーカイブが添付ディスク 8 に含まれています。

所 在： nifty / fsharp3 / lib9 / 41 : PLAIN2\_X.LZH<sup>1)</sup>

nifty / fsharp3 / lib9 / 137 : PLAIN2X2.LZH

使 用 法： plain2 の吐き出す標準出力をリダイレクトして  $\text{\LaTeX}$  ソースを得ます。

```
A> plain2.x -tex sample.doc > sample.tex
```

## plain2 による L<sup>A</sup>T<sub>E</sub>X ソースの生成

plain2 は、特定の記述の仕方を守ってさえいれば、ある程度の L<sup>A</sup>T<sub>E</sub>X ソースを作成してくれます。ここであげるサンプルの場合、左側のプレーンテキストを plain2 で処理した結果が右側のソースになりました。

itemize, enumerate, description 環境を使い分けていることがわかっていただけたと思います。

表組を作る際にも、plain2 を使用すれば、ある程度視覚的に作業がすすめられるかもしれませんね。

### 1. 「コンピュータ犯罪」

「コンピュータ犯罪」とは、コンピュータが犯罪の目的あるいは道具のいずれかになっている犯罪行為をいう。

#### 1.1. 窃盗型

- ・ 金銭を盗む
- ・ 商品を盗む
- ・ 情報を盗む

#### 1.1.1. テクニック

- (1) サラミ (The Salami)
- (2) トラップ・ドア (trap door)

#### 1.2 改変・破壊型

ハッカー行為 : ネットワークの不正利用  
破壊行為 : データの破壊・書き換え・消去。

#### 1.2.1 テクニック

| 不正プログラム |                  |
|---------|------------------|
| 寄生虫     | Worms            |
| ウイルス    | Virus            |
| 時限爆弾    | Time Bomb        |
| ロジック爆弾  | Logic Bomb       |
| トロイの木馬  | The Trojan Horse |

```
\documentstyle[b5j]{jarticle}
\setcounter{secnumdepth}{6}
\setcounter{tocdepth}{6}
\topsep=0.1cm \parsep=0.1cm \itemsep=0.0cm
\newenvironment{nquote}[1]{%
{\list{}{\leftmargin=#1}\item[]}%
}{\endlist}
\begin{document}

\section{「コンピュータ犯罪」}
\par
「コンピュータ犯罪」とは、コンピュータが犯罪の目的ある
いは道具のいずれかになっている犯罪行為をいう。
\medskip

\subsection{窃盗型}
\begin{itemize}
\item 金銭を盗む
\item 商品を盗む
\item 情報を盗む
\medskip
\end{itemize}

\subsubsection{テクニック}
\begin{enumerate}
\item サラミ (The Salami)
\item トラップ・ドア (trap door)
\medskip
\end{enumerate}

\subsection{改変・破壊型}
\begin{description}
\item[ハッカー行為 :] ネットワークの不正利用
\item[破壊行為 :] データの破壊・書き換え・消去。
\medskip
\end{description}

\subsubsection{テクニック}
~\
\begin{tabular}{|l|l|}
\hline
\multicolumn{2}{|c|}{不正プログラム} ~\
\hline
寄生虫 & %
\multicolumn{1}{|c|}{Worms} ~\
\hline
ウイルス & %
\multicolumn{1}{|c|}{Virus} ~\
\hline
時限爆弾 & %
\multicolumn{1}{|c|}{Time Bomb} ~\
\hline
ロジック爆弾 & %
\multicolumn{1}{|c|}{Logic Bomb} ~\
\hline
トロイの木馬 & %
\multicolumn{1}{|c|}{The Trojan Horse}~\
\hline
\end{tabular}~\
\end{document}
```

## dvi2tty

**概 略：** グラフィックを出力できない端末用に、アルファベットや記号を使って TeX の出力を擬似的に表現するプレビューアです。グラフィック表示ではないので正確な表現ではありませんが、出力のおおよその雰囲気はつかめます。また、TeX 処理系を持たない相手に対してソースを渡す際にも使用されているようです。

Svante Lindahl 氏が Pascal で開発、Marcel J. E. Mol 氏によって C に移植され、これをもとにして X680x0 版が移植されています。

**所 在：** nifty / fsharp3 / lib9 / 19 : Dvi2Tty.Lzh

**使 用 法：** プレビューしたい dvi ファイルが sample.dvi で、その出力を sample.doc とする場合、以下のようにします。

```
A> dvi2tty.x -osample.doc -w90 sample.dvi
```

オプション “-w” を設定することで、1 行の文字数を調整できます。



# プレビューワとしての dvi2tty

## 1 preview.x (print.x) の出力

- 文章の場合  
preview.x (print.x) と dvi2tty.x の出力には、どれだけの違いがあるでしょうか。
- 表の場合

| 《 TeX における単位標記の一例 》 |        |                          |
|---------------------|--------|--------------------------|
| pt                  | ポイント   | 1pt = 0.3515mm = 65536sp |
| mm                  | ミリメートル | 1mm = 0.1cm = 2.834pt    |
| in                  | インチ    | 1in = 25.4mm = 72.27pt   |

- 数式の場合

$$A = \int_a^b f(x) dx = \lim_{n \rightarrow \infty} \sum_{i=1}^n f(x_i) \Delta x_i$$

## 2 dvi2tty.x の出力

### プレビューワとしての dvi2tty

#### 1 preview.x (print.x) の出力

ffl 文章の場合

preview.x (print.x) と dvi2tty.x の出力には、どれだけの違いがあるでしょうか。

ffl 表の場合

| 《 TEX_ における単位標記の一例 》 |        |                          |
|----------------------|--------|--------------------------|
| pt                   | ポイント   | 1pt = 0.3515mm = 65536sp |
| mm                   | ミリメートル | 1mm = 0.1cm = 2.834pt    |
| in                   | インチ    | 1in = 25.4mm = 72.27pt   |

ffl 数式の場合

$$A = \int_a^b f(x) dx = \lim_{n \rightarrow \infty} \sum_{i=1}^n f(x_i) \Delta x_i$$

## Nemacs or Mule, and Emacs Lisp

1)Nemacs は「日本語 Emacs」を意味します。

2)Mule は「多国語対応 Emacs」(Multilingual enhancement to GNU Emacs) のことで、日本語の全角および半角文字のほか、中国語やハングル語・フランス語・ドイツ語・エスペラント語・ロシア語・ヘブライ語・タイ語など、20ヶ国語以上の文字表示に対応しています。

概 略: Emacs (Nemacs<sup>1)</sup>, Mule<sup>2)</sup>) には tex-mode (仮に emacs-tex-mode とします) がついており、コマンド補完 (completion) など、ソース作成上のさまざまな支援をしてくれます。ただ、emacs-tex-mode は plain TeX 用のものといつてよく、LaTeX についての機能は十分ではありません。そこで、LaTeX 用の Emacs Lisp コードが作成されています。

cmutex は、後述する  $\mu$ Emacs の LaTeX-mode の原型となったパッケージです。カーネギーメロン大学 (CMU) で開発されました。原型を Olin Shivers 氏が作成し、その後、さまざまな人の手によって改良されていったようです。ショートカットキーによるコマンド打ち込みをはじめ、ソースファイルの整形や添削などの機能も持っています。emacs-tex-mode と比べて LaTeX ソースの作成について機能強化されたものです。

AUC-TeX は、デンマークの Aslborg 大学で開発されました。plain TeX に加え、LaTeX,  $\mathcal{A}\mathcal{M}\mathcal{S}$ -TeX,  $\mathcal{A}\mathcal{M}\mathcal{S}$ -LaTeX, BibTeX など、広範囲にわたる “TeXmode” を備えています。おそらく、現在最も高機能な TeX 用 Emacs Lisp パッケージなのではないでしょうか。

yatex は、Yet Another tex-mode 『野鳥』として、広瀬雄二氏によって作成されています。機能自体は AUC-TeX と比べるべくもないものですが、その分シンプルで、日本語入力を前提としているだけに使い勝手もいいようです。

所 在: pcvan / sscience / osl1 / 490 : CMUTEX.LZH  
nifty / flabo / lib11 / 252 : auctex73.taz  
nifty / flabo / lib11 / 272 : yatex143.lzh

使 用 法: 上記 Emacs Lisp パッケージを X680x0 版 Nemacs や Mule にインストールし、当該パッケージを使用してソースを作成します。

マニュアル: 『野鳥: YaTeX A Guide to the Yet Another tex-mode for emacs Version 1.40 野鳥の飼い方』<sup>3)</sup>

3)texinfo 形式です。

広瀬雄二 著 1993

nifty / flabo / lib11 / 192 : YDOC140.LZH

解 説 書: 『Nemacs 入門 6』(『UNIX MAGAZINE』1991.7)

荒井美千子 著 アスキー

『Nemacs 入門 15, 16 cmutex』(『UNIX MAGAZINE』1993.1, 3)

荒井美千子 著 アスキー

『使ってみよう AUC-TeX』(『UNIX MAGAZINE』1993.9~12)

小林伸治 著 アスキー

概 略： いかむ氏・homy氏らにより開発された j1.31 を、lika氏・SALT氏・PEACE氏・SHUNA氏・rima氏らが機能拡張とバグフィックスを重ねた結果、現在に至る X680x0 版の  $\mu$ Emacs は、他機種の名義エディタとは比べようもないほどの高性能・高機能を実現しています。そのうちのひとつである LaTeX-mode は、Emacs 上で動作する Lisp パッケージ、cmutex をもとにして rima 氏が開発したもので、コマンド補完をはじめ、ショートカットキーによるコマンド打ち込みなど、ソースファイル作成上のさまざまな支援してくれます。 $\mu$ Emacs の FEP コントロール機能<sup>1)</sup>と併用することで、L<sup>A</sup>T<sub>E</sub>X ソースファイルの記述がたいへん楽になることは間違いありません。「Nemacs は大きすぎて」という方は、 $\mu$ Emacs を使用しましょう<sup>2)</sup>。

所 在： nifty / fsharp3 / lib5 / 588 : em\_r5b.Lzh

使 用 法： インストールして、 $\mu$ Emacs の LaTeX-mode でソースの入力を行います。編集するファイルの拡張子が .tex であれば、自動的に同モードになりますし、拡張子が .tex ではない場合でも、標準のキーバインドにおいて  $\sim$ Xm に割り付けられている「モード追加」で LaTeX を指示すれば LaTeX-mode になります。

なお、「所在」の箇所ですしたバージョンに付属している dump キットには、開発者によって不具合が報告されています。makedump.bat の exe -98304 ... を exe -98432 に、makeex.c の DIFF\_SIZE を 0x18000 から 0x18080 に変更して makeex.x を再作成することで、暫定的に対応できるそうです。

解 説 書： 『 $\mu$ Emacs 3.10 j1.43(rel.2) EXTEND SET』

lika・SALT・PEACE・SHUNA・rima 著 1991

nifty / fsharp1 / mes16 / 457-460 : em\_man\_r2.Lzh<sup>3)</sup>

『MicroEmacs フル・スクリーン・テキスト・エディタ リファレンス・マニュアル Version 3.9』

Brian Straight・Daniel M. Lawrence 著 /

新島智之・加藤章 訳 1987

nifty / fsharp3 / lib10 / 12 : JMEDOC.LZH<sup>4)</sup>

1)  $\mu$ Emacs のコマンド使用時に、FEP の状態 (開閉) を自動的に制御してくれる機能。

2) 本章の筆者も、 $\mu$ Emacs を愛用しており、本書のソース作成も  $\mu$ Emacs で行いました。

3) Mtank氏によって L<sup>A</sup>T<sub>E</sub>X ソース形式にまとめられています。ただし、これは rel.2 のドキュメントをもとにしているものです。rel.5b のドキュメントは実行アーカイブに添付されています。

4) 「がっちゃん」版  $\mu$ Emacs v3.9e J1.3 の英文マニュアルを邦訳したものです。原稿はテキスト・フォーマット ntf の形式で書かれています。アーカイブには通常のテキスト形式に出力したものも含まれています。



## あとかき

すごく時間がかかってしまいましたが、やっと TeX の本をまとめることができました。当初は私がこの本にもっと深くかかわるはずだったのですが、なかなか他の仕事が忙しくて、それを許してくれませんでした。結果として他の執筆者や編集の方に多大な迷惑をおかけしてしまいました。本当はもっと早くこの本が出るはずだったのですが、上記のような事情で遅れてしまいました（全部が私の責任ではないと思うけど……。甘いかな。）

TeX も GCC も UNIX から私が移植したのですが、移植に際してのトラブルがほとんどなかったので、メンテナンスの必要がまったくありませんでした。そういった理由もあって、ソフトウェア作成の負担では一番軽い立場にあったのですが、なかなか筆が進まず……。何度も「いつ出そう？」とたずねてくださった某販売店の方にも、これで「出ました」とやっと報告することができます。本書によって、TeX の世界を存分に楽しんでいただけたら幸いです。

吉野智興

本書のデバイスドライバ関係を担当した Ext です。デバイスドライバの開発者本人でもあります。私が NIFTY-Serve (以下、NIFTY と略す) に加入したのが 1989 年 7 月 28 日でした。NIFTY の事務局に問い合わせたから間違いはありません。

シャープフォーラムのともみちゃん、元気してます？

その日すぐさま NIFTY のシャープフォーラムに加入しました。そして、シャープフォーラムでの私のデビュー作が本書に添付されているプレビューアの原型だったのです。あれから本書が発刊されるまでちょうど 5 年の歳月が流れました。私は数学科出身ですから、計算に間違いはありません。

松村先生、四方先生、吉野先生、内藤先生、お世話になりました。

その間、プリンタドライバも生まれ、プレビューアとともに NIFTY シャープフォーラムのユーザに育てられ成長し続けました。いろいろな要望が次から次へと出てくるなか、ひとつひとつ着実に機能アップしてゆき、時がたつにつれだんだんと太ってきたことに気がついたのです。私のことではありません。プレビューアとプリンタドライバのプログラムサイズのことです。体重を測ったわけではありませんが、自分を信じていますから間違いはありません。

シャープフォーラムのユーザの皆さん、ありがとう。

そこで、この 2 つのプログラムのフォント生成部分を統合してフォントマネージャが生まれました。

これはとても衝撃的な事件でした。

なぜなら、プレビューアとプリンタドライバを私の子供たちにたとえるなら、フォントマネージャは私の孫ということになってしまうからです。まだ結婚すらしていないのに、初孫というのはいただけません。

いえいえ、これはまったくのでたらめです。そうではなくて、フォントマネージャの機能に優れた特徴が備わっていたからでした。どんなに優れた特徴があるかの記述は本文にもありますのでここでは割愛しますが、たとえば、本書にも添付してある X680x0 の多書体プロジェクト (日本語の各種変形フォントを扱う TeX システム) は、パソコン、ワークステーションクラスでは今でも世界中を見渡しても相当するものは見あたらないと思います。フォントマネージャなしではとうてい実現できなかったでしょう。

多書体プロジェクトの作者がいていたのだから間違いはありません。

りまちゃん、DIG そして Seirios ご苦労様。

このような優れたシステムを、大規模なメディアを通じて、より多くの皆さんに使っていただきたいと日頃から思っていたところ、ソフトバンク社から X680x0 版の TeX 本を書かないかとのお願いを受け、二つ返事で承諾しました。

本原稿で TeX を使うというのは最近では珍しくありませんが、著者・編集者間での連絡にパソコン通信を使ったのには少なからず興奮しました。思ったようには意思が通じないこともありましたが、パソコン通信も立派なメディアであるのだと、あらためて感じたものです。なにより議論内容がそのままログとしてあとまで残るのがよいですね。そのログによりますと、連絡ボードは去年の 6 月にスタートしています。なんと出版するまで 1 年以上もかかっていたのです。なにもそんなに威張らなくてもいいのにお思いでしょうが。

その間、ソフトバンクの 040turbo 本の割り込みがかかって、編集作業が一時中断したこともありました。

2 人目の赤ちゃんおめでとう。今度見にくぞお> BEEPs

話は変わります (ぶっしゅとうすたああつく<sup>1)</sup>) が、この 040turbo 上でも本書の TeX システムが動きます。040turbo で動かすと、X680x0 用 プレビューアは、486DX2 66MHz マシンで動作する MS-Windows のプレビューアとほぼ互角の速度が得られるそうです。

見切りをつけて互換機に流れていったみんな、帰っておいでえ。

(ぼっふふろむすたああつく<sup>2)</sup>)

誰だ? “まだスタックポインタ初期化してない” って言うやつは? <sup>3)</sup>

しかし、そのおかげで多少余裕ができ、最近の話題を詰め込むことができました。その分、内容も濃く仕上がったと確信しております。

ソフトバンク社の関係者には本当にお世話になりました。昔は「流通業界を独占している」だの、「フリーソフトに対する理解がまったくない」だのいろいろと陰口を叩いておりましたが、これを機会に改心するつもりです。やっぱ、いいよ。ソフトバンクは。

1) 編集部注: 「push to stack」と思われる。

2) 編集部注: 「pop from stack」と思われる。

3) 編集部注: 先の推測はどうやら正しいらしい。

特にあささんには感謝しても感謝しきれるものではありません。

ということで、本書はとてもすばらしい本であります。

つくった本人がいうのだから間違いはありま(いてっ) …ってこれが一番信用できませんね。(^^;

川本琢二

単なる X680x0 版  $\text{\TeX}$  の一ユーザにすぎなかった私のところに、 $\text{\TeX}$  本の原稿を書きませんか？ と話がきたのは今年のまだ暑いころでした。私なんかやっていたいの？ というのが正直な気持ちでしたが、よい経験だと思って引き受けてみました。それから、ない知識を絞り出して執筆を進めたのですが、はじめての経験なので勝手がわからず、満足のいく出来にはなりませんでした。担当の方や共著の方々の期待に応えることができず、心苦しんでいます。

また、今回、この本のために 2 つほどプログラムを組んでみましたが、実力不足がもろに出てしまい、あまり使い勝手はよくないと思います。なにしろプログラムとしては未熟なので笑って許してやってください。

さて、やはり避けては通れない X680x0 の現状の話題ですが、正直なところ、ますます厳しくなりつつあると思います。PowerPC が普及しつつある昨今、もはや 68030 でさえ、時代遅れと思われてしまいますし、新機種の話も聞こえてきません。このまま廃れていくのかと思うと悲しい気分になります。でも、X680x0 にもまだまだ魅力はありますよね。長年慣れ親しんだこの愛機を、私はこれからも使い続けていきたいと思っています。

山崎岳志

4) オリジナルの  $\text{\TeX}$  の最新版が Ver.3.1415 であるのに対して、アスキーの日本語  $\text{\TeX}$  は Ver.2.99 がベースです。 $\text{\LaTeX}$  にしても、公式に対応しているのは Ver.2.09 (<24 may 1989>) でしかありません。 $\text{\TeX}$  の Ver.3 はマルチバイトコードの文字に対応しているのですから、アスキーさんとインプレスさんには、ぜひ、この日本語化に取り組んでいただきたいものです。

もともと一介の「 $\text{\LaTeX}$  使い」にすぎなかったはずの私が、こうして X680x0 版  $\text{\TeX}$  本の執筆に参加し、「こんなのは  $\text{\LaTeX}$  の使い方じゃない」と不満をいながらも本書のスタイルファイルやマクロをいじりまわし、何とか本の形にして(まだなっていないが)、今、あとかきを書いているのだから、奇妙なものである。

他にも何やらいろいろと、書きたいこと、描きたいこと、言いたいこと、伝えたいことがあったような気がする<sup>4)</sup>が、いまは、これまで。

過去私を支えてくれて、現在私を支えてくれている、そして、未来に私を支えてくれるだろうすべての人に、私のなけなしの感謝のすべてを込めて。

実森仁志



# INDEX

## 記号

- CRLF ..... 201
- dpi ..... 188
- dump オプション ..... 211
- extraCRLF ..... 201
- FF ..... 202
- graphic ..... 202
- height ..... 188
- highReso ..... 63
- init ..... 199
- mf= ..... 222
- MSBisLeft ..... 195
- MSBisUpper ..... 195
- pinBytes ..... 198
- pinHeights ..... 198
- prBufSize ..... 198
- relative ..... 208
- remark ..... 188
- repeat ..... 210
- start ..... 206
- width ..... 188
- xOffset ..... 193
- y ..... 225
- yOffset ..... 193
- 8 進数 ..... 146
- 11pt ..... 105
- 12pt ..... 105
- 16 進数 ..... 146
- \$ ..... 99
- \$ \$ ..... 130
- \$ \$ \$ ..... 131
- \$\cal \$ ..... 144
- \$\mit \$ ..... 144
- % ..... 99
- & ..... 12, 99
- ~ ..... 99
- \ ..... 97, 99, 110
- \[ \] ..... 131
- \( \) ..... 130
- \, ..... 154
- \\ ..... 150
- \\_ ..... 99, 153
- \Alph ..... 122
- \alph ..... 122
- \arabic ..... 122
- \author ..... 109
- \begin{displaymath} ..... 131
- \begin{document} ..... 109, 111
- \begin{eqnarray} ..... 131
- \begin{equation} ..... 131
- \begin{itemize} ..... 119, 121, 123
- \begin{math} ..... 130
- \bf ..... 135, 144
- \boldmath\$\cal \$\unboldmath ..... 144
- \boldmath\$\mit \$\unboldmath ..... 144
- \chapter ..... 113
- \cleardoublepage ..... 152
- \clearpage ..... 152
- \date ..... 109
- \documentstyle ..... 104
- \dump ..... 11, 218
- \em ..... 124
- \end ..... 227
- \end{displaymath} ..... 131
- \end{document} ..... 110, 111
- \end{eqnarray} ..... 131
- \end{equation} ..... 131
- \end{itemize} ..... 119, 121, 123
- \end{math} ..... 130
- \fnsymbol ..... 122
- \footnote ..... 125
- \footnotemark ..... 125
- \footnotesize ..... 139
- \footnotetext ..... 125
- \gt ..... 144
- \hfil ..... 153
- \hfill ..... 153
- \hspace ..... 153
- \hspace\* ..... 153
- \Huge ..... 139
- \huge ..... 139
- \it ..... 135, 144
- \labelitemi ..... 120
- \LARGE ..... 139

- \Large ..... 139
  - \large ..... 139
  - \linebreak ..... 150
  - \magstep ..... 139
  - \maketitle ..... 105, 109
  - \mc ..... 144
  - \mit ..... 144
  - \newfont ..... 145
  - \newline ..... 150
  - \newpage ..... 152
  - \noindent ..... 102
  - \nolinebreak ..... 150
  - \nopagebreak ..... 151
  - \normalsize ..... 105, 139
  - \pagebreak ..... 151
  - \par ..... 102
  - \paragraph ..... 115
  - \part ..... 115
  - \qqquad ..... 154
  - \quad ..... 153
  - \renewcommand ..... 122
  - \rm ..... 144
  - \Roman ..... 122
  - \roman ..... 122
  - \samepage ..... 152
  - \sc ..... 144
  - \scriptsize ..... 139
  - \section ..... 113
  - \setcounter ..... 115
  - \sf ..... 144
  - \sl ..... 144
  - \small ..... 139
  - \subparagraph ..... 115
  - \subsection ..... 113
  - \subsubsection ..... 115
  - \table ..... 227
  - \tiny ..... 139
  - \title ..... 109
  - \tt ..... 144
  - \undump ..... 15
  - \unboldmath ..... 144
  - \verb ..... 127
  - \verb\* ..... 128
  - \vfil ..... 156
  - \vfill ..... 156
  - \vspace ..... 156
  - \vspace\* ..... 156
  - # ..... 99
  - ..... 99
  - { ..... 99
  - } ..... 99
  - ~ ..... 99
  - ¥ ..... 97
- A**
- a4j ..... 105
  - a5j ..... 105
  - $\mathcal{A}\mathcal{M}\mathcal{S}$ - $\mathrm{L}\mathrm{A}\mathrm{T}\mathrm{E}\mathrm{X}$  ..... 240
  - amsppt.sty ..... 238
  - $\mathcal{A}\mathcal{M}\mathcal{S}$ - $\mathrm{T}\mathrm{E}\mathrm{X}$  ..... 238
  - ascjletter ..... 108
  - ascletter ..... 108
  - ascmac ..... 105
  - AUC- $\mathrm{T}\mathrm{E}\mathrm{X}$  ..... 268
  - aux ファイル ..... 84
- B**
- b4j ..... 105
  - b5j ..... 105
  - .base 拡張子のファイル ..... 19, 218
  - B $\mathrm{B}\mathrm{T}\mathrm{E}\mathrm{X}$  ..... 233
  - Big  $\mathrm{T}\mathrm{E}\mathrm{X}$  ..... 29, 32
  - big endian ..... 203
  - BJ-10 ..... 186
- C**
- Chem- $\mathrm{T}\mathrm{E}\mathrm{X}$  ..... 242
  - chemstruct.sty ..... 242
  - CM Font Family ..... 142
  - cmutex ..... 268
  - COMMAND.X ..... 30
  - cut ファイル ..... 256, 258
- D**
- dchem.sty ..... 242
  - depth ..... 55
  - description 環境 ..... 122
  - document 環境 ..... 110
  - dpi サイズ ..... 224
  - dump ファイル ..... 71
  - dump2tty.x ..... 173
  - dvi2ps.x ..... 49, 260
  - dvi2tty ..... 266
  - dvi ファイル ..... 7, 48, 84, 91
- E**
- ecleepic.sty ..... 253
  - eclepsf.sty ..... 254

eepic.sty ..... 253  
 em ..... 126, 138  
 Emacs ..... 268  
 Emacs Lisp ..... 268  
 enumerate 環境 ..... 120  
 enumi ..... 122  
 epic ..... 105  
 epic.sty ..... 253  
 EPSF ..... 254  
 epsf.sty ..... 254  
 eqnarray 環境 ..... 131  
 equation 環境 ..... 131  
 escpdvi ..... 261  
 ex ..... 138, 140

**F**

FAX2PBM.EXE ..... 68  
 fax2ttl.lzh ..... 68, 73  
 FAXION 大魔王 ..... 68  
 FILES ..... 18, 30  
 FMset ..... 40, 92  
 .fmt 拡張子のファイル ..... 19  
 fmt ファイル ..... 11, 12, 14, 16  
 fragile なコントロール・  
     シーケンス ..... 159

**G**

GCC ..... 4  
 gftopk ..... 217  
 GHOSTSCRIPT ..... 50, 260  
 GNU C Compiler ..... 4  
 GNUPLOT ..... 258  
 Gpic ..... 255

**I**

idx ファイル ..... 84  
 inifm.x ..... 216, 219  
 initex.x ..... 10, 15  
 INTEL オーダー ..... 203  
 install.x ..... 22  
 itemize 環境 ..... 119, 121, 123

**J**

jarticle ..... 105, 108, 114  
 jbook ..... 108  
 jfm ファイル ..... 183  
 jreport ..... 108, 114  
 jtwocolumn ..... 105

**K**

Knuth 博士 ..... 2, 232  
 kodvi.win ..... 49

**L**

Lamport 氏 ..... 79, 116, 136, 233  
 L<sup>A</sup>T<sub>E</sub>X ..... 240  
 landscape ..... 105  
 L<sup>A</sup>T<sub>E</sub>X ..... 10, 79, 233  
 latex.x ..... 12, 13, 14  
 L<sup>A</sup>T<sub>E</sub>X の出力性能 ..... 94  
 L<sup>A</sup>T<sub>E</sub>X の本体部分 ..... 104  
 LESQUA ..... 68  
 lips3dvi ..... 261  
 little endian ..... 203  
 lndrv.x ..... 13, 14  
 ln.x ..... 14  
 log ファイル ..... 84  
 LR モード ..... 113

**M**

magstep ..... 223  
 makefont.x ..... 221  
 makeidx ..... 105  
 Makeindex ..... 263  
 MARU24.FNT ..... 180  
 METAFONT ..... 214  
 MFBASES ..... 219, 220  
 mfged ..... 50  
 MFHOME ..... 221  
 MFINPUTS ..... 220  
 MFPOOL ..... 220  
 .mf 拡張子のファイル ..... 19  
 μEmacs ..... 89, 269  
 minipage 環境 ..... 125  
 MOTOROLA オーダー ..... 203  
 MSB ..... 197  
 Mule ..... 268  
 multicol ..... 105  
 multicol.sty ..... 248  
 MusicExp ..... 246  
 MusicT<sub>E</sub>X ..... 246  
 M<sub>U</sub>T<sub>E</sub>X ..... 244  
 myfonts.fm ..... 167, 176, 178

**N**

Nemacs ..... 268  
 New Font Selection Scheme  
     ..... 140, 141



NFSS ..... 140, 141  
 NTT- $\TeX$  ..... 4  
 .numgf 拡張子のファイル ..... 19

**O**

OMP ..... 244

**P**

PBM2FAX.EXE ..... 68  
 piclatex ..... 105  
 P $\TeX$  ..... 252  
 .pk 拡張子のファイル ..... 20  
 plain  $\TeX$  ..... 10, 232  
 plain2 ..... 264  
 plain.base ..... 218  
 plain.fmt ..... 12  
 .pool 拡張子のファイル ..... 19  
 preamble ..... 109  
 preview.cfg ..... 59  
 preview.p3m ..... 59  
 preview.x ..... 48, 56  
 print.cfg ..... 64, 186  
 print.p3m ..... 64  
 print.x ..... 49, 64, 93  
 p $\TeX$  ..... 78  
 pt ..... 138  
 ptex.pool ..... 17

**Q**

quotation 環境 ..... 117  
 quote 環境 ..... 117

**R**

ROM フォント ..... 24, 31

**S**

secnumdepth ..... 115  
 seminar.sty ..... 236  
 Sl $\TeX$  ..... 236  
 Starfax 形式 ..... 69, 71  
 strage.sys ..... 168  
 stylefile ..... 108  
 style\_option ..... 105  
 .sty 拡張子のファイル ..... 19  
 SXpreview.x ..... 48

**T**

Tarticle.sty ..... 250  
 tate.sty ..... 250

tatex.sty ..... 250  
 tcard.sty ..... 250  
 tex2ttl.lzh ..... 69  
 $\TeX$  ..... 2  
 $\TeX$  ブック ..... 147, 232  
 $\TeX$  の環境変数 ..... 16  
 $\TeX$  のデフォルトサイズ ..... 189  
 $\TeX$  の HOME ディレクトリ ..... 39, 65  
 TEXEDIT ..... 17, 89  
 TeXenv.bat ..... 39  
 TEXFMT ..... 17  
 TEXTFONTS ..... 17  
 TEXTFORMATS ..... 17  
 texinfo 形式 ..... 20  
 TEXINPUTS ..... 17  
 TEXPK ..... 221  
 TEXPOOL ..... 17  
 .tex 拡張子のファイル ..... 19  
 tfm ファイル ..... 7, 183  
 .tfm 拡張子のファイル ..... 19  
 titlepage ..... 105, 106  
 title.sys ..... 69  
 toc ファイル ..... 84  
 Tpic ..... 255  
 transfig ..... 256  
 ttl2fax.lzh ..... 68, 71  
 TwentyOne.x .... 15, 18, 25, 30, 215  
 TwOne30x.x ..... 30  
 TwOne202.x ..... 30  
 TwOne203.x ..... 30  
 twoside ..... 105

**U**

UTILS. $\TeX$  ..... 242

**V**

verbatim 環境 ..... 128  
 verbatim\*環境 ..... 129  
 V $\TeX$  ..... 234  
 virmf.x ..... 216, 220  
 virtex.x ..... 10, 16

**W**

Warning ..... 84  
 -width ..... 188  
 WYSIWYG ..... 2

**X**

X680x0 版  $\TeX$  ..... 5, 6

xfig ..... 256  
 Xe<sub>Λ</sub>MT<sub>E</sub>X ..... 242

## Y

yatex ..... 268

## Z

Z's STAFF ..... 31, 33, 167, 177  
 Z's WORD JG Ver.3  
     ..... 24, 31, 34, 167, 178  
 Z's WORD JG ..... 31, 34, 167, 177

## あ

アウトラインフォント ..... 166  
 アスキー版日本語 T<sub>E</sub>X ..... 5  
 イタリック補正 ..... 144  
 イメージデータ ..... 195  
 インストールディレクトリ ..... 39  
 インデント ..... 111  
 引用 ..... 116  
 インライン数式 ..... 130  
 ウィズ ..... 141  
 ウェイト ..... 141  
 動く引数 ..... 159  
 エスケープ文字 ..... 110  
 エラー ..... 18, 86, 226  
 エラー・インジゲータ ..... 87  
 エラー対策 ..... 63, 66  
 エラーメッセージ ..... 157  
 エラー・ロケータ ..... 87

## か

改行 ..... 90, 101, 150  
 改行幅 ..... 139  
 改ページ ..... 151  
 カウンタ ..... 115  
 カットファイル ..... 258  
 拡大率 ..... 224  
 簡条書き ..... 118  
 環境 ..... 110  
 環境エリアサイズ ..... 39  
 脚注 ..... 125  
 キャッシュ ..... 165  
 キャッシュバッファ ..... 170  
 キャッシュフィルタ ..... 167, 168  
 強調 ..... 124  
 空行 ..... 102, 111  
 空白 ..... 99, 111  
 グルーピング ..... 124, 132

警告 ..... 84  
 原稿サイズ ..... 190  
 コマンド ..... 7, 96  
 近藤版 GCC ..... 4  
 コントロール・シーケンス 7, 96, 110  
 コントロール・シーケンスの埋め込み  
     ..... 136  
 コントロール・シンボル ..... 97, 99  
 コントロール・ワード 97, 98, 99, 110  
 コンフィギュレーションファイル 163

## さ

最上位ビット ..... 197  
 サイズ ..... 141  
 作成フォント名 ..... 224  
 左右モード ..... 113  
 シェイプ ..... 141  
 視覚デザイン ..... 116  
 字下げ ..... 102  
 斜体フィルタ ..... 54  
 章節見出しの出力制御 ..... 113  
 書体倶楽部 ..... 31, 34, 167, 178  
 シリアルプリンタ ..... 195  
 シリーズ ..... 141  
 シングルクォーテーションマーク 154  
 シンボリックリンク ..... 12, 13  
 数式モード ..... 113  
 スタイルオプション ..... 105  
 スタイルファイル ..... 104, 108  
 スタイルファイルオプション ..... 105  
 スプールバッファ ..... 198  
 スラント処理 ..... 54  
 全角文字 ..... 106  
 ソースファイル ..... 7  
 相互参照 ..... 84

## た

ダイエット ..... 53, 177  
 ダイエットフィルタ ..... 177, 178  
 対応プリンタ ..... 23, 36  
 多書体化 (Various Font) ..... 234  
 縦方向の空白 ..... 155  
 タブ ..... 111  
 ダブルクォーテーションマーク ... 154  
 単位記号 ..... 138  
 単語間空白 ..... 99, 111  
 段落モード ..... 112  
 ツァイトフォント ..... 165, 177, 234  
 ディスプレイ数式 ..... 131

デバイスドライバ ..... 48  
 デフォルトサイズ ..... 189  
 デリミタ ..... 99, 106  
 ドキュメントスタイル ..... 108  
 特殊改行コード ..... 201  
 特殊文字 ..... 97  
 ドット ..... 172  
 ドライバ制御系のオプション ..... 187

## な

日本語 T<sub>E</sub>X ..... 4  
 日本語フォント ..... 31  
 ノーマルレゾリューション ..... 63

## は

パーソナル日本語 T<sub>E</sub>X  
     フォントライブラリ .. 31, 35  
 ハイレゾリューション ..... 63  
 パスの区切り ..... 16  
 バッファサイズ ..... 190  
 パラグラフモード ..... 112  
 半角カナ ..... 103, 111  
 半角スペース ..... 97  
 ビクチャーモード ..... 113  
 微小な空白 ..... 154  
 ビットイメージデータ ..... 195  
 ビットマップフォント 31, 32, 165, 177  
 描画モード ..... 113  
 ファックスファイル変換 ..... 68  
 ファックスモデム ..... 69  
 ファミリー ..... 141  
 フォントジェネレータ ..... 51, 165  
 フォントテーブル ..... 146  
 フォントドライバ ..... 51, 52  
 フォントの大きさ ..... 139  
 フォントの変形 ..... 50  
 フォントの変更 ..... 140  
 フォントフィルタ ..... 52  
 フォントマネージャ ..... 50, 163  
 フォントミキサ ..... 52, 54  
 復帰改行 ..... 201  
 プリアンブル ..... 109  
 プリミティブ ..... 104  
 プリンタ制御系のオプション ..... 187  
 プレビュー ..... 7  
 プレビューア ..... 56  
 フロート ..... 152  
 文書スタイル ..... 108  
 ページバッファ ..... 60

ページプリンタ ..... 196  
 ポイント ..... 126, 172  
 補正 ..... 194

## ま

前田薫 ..... 4  
 マークアップ方式 ..... 56  
 マクロパッケージ ..... 229  
 マクロファイル ..... 10  
 丸文字フォント ..... 180  
 文字の大きさ ..... 137

## や

野鳥 ..... 268  
 横方向の空白 ..... 153

## ら

ラストプリンタ ..... 188  
 レイアウトパラメータ ..... 109  
 論理デザイン ..... 116

## わ

割り算ファクタ ..... 207



本書に添付されたディスク内の、プログラム、データおよびそれに準じるもの（以下、これらをまとめて「プログラム」と総称します）は、正しく動作することを望んで製作されていますが、その動作はいっさい保証されていません。したがって、各プログラムを使用したために生じたいかなる損害についても、以下に名前をつらねる人および法人がその補償をすることはありません。各「プログラム」は、ユーザの責任において使用してください。

また、各「プログラム」は、刊行時点の最新版を添付するように努めていますが、さまざまな理由から、必ずしも最新版ではないことがあります。この場合でも、以下で名前をつらねる者に、これに対応する義務はないものとします。

本書の内容に関するお問い合わせは、返信用の封筒に切手を貼ったものを同封のうえ、必ず封書で「ソフトバンク株式会社 ハードウェア活用書編集部 『X680x0 TeX』係」までお願い致します。なお、本書の内容以上に関するお問い合わせにはお答えしかねますので、御了承ください。

### X68k Programming Series #3

## X680x0 TeX

1994年7月26日 初版 第1刷 発行

|    |                   |                   |
|----|-------------------|-------------------|
| 著者 | よしの ちくみ<br>吉野 智興  | かわもと たくじ<br>川本 琢二 |
|    | やまざき たかし<br>山崎 岳志 | じつもり ひとし<br>実森 仁志 |

発行者 橋本 五郎

発行所 ソフトバンク株式会社 出版事業部

〒103 東京都中央区日本橋浜町 3-42-3

TEL 販売 03 (5642) 8101

編集 03 (5642) 8140

印刷 東京書籍印刷株式会社

© Printed in Japan

ISBN 4-89052-542-4

落丁本、乱丁本はお取り替えいたします。

定価は表紙に記載されています。

Cover Design = Tetsuya Yonetani

Style Design = Tateaki Hori



郵便はがき

料金受取人払

日本橋局  
承認

1277

差出有効期間  
平成 8 年 4 月  
30 日まで

1 0 3 - 0 0

1 6 1

東京都中央区  
日本橋浜町3-42-3

ソフトバンク(株)出版事業部

ハードウェア活用書編集部行

住所

|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  |  |
|--|--|--|--|--|

☎

氏

名

年

齢

性

別

男

女

職業・勤務先  
学校(学部・学年)

所有機種



# X680x0 TeX

弊社ソフトバンクの本をお買い上げいただきありがとうございます。  
今後の編集の資料にさせていただきますので、下記のアンケートにお答え下さい。ご協力をお願いいたします。

## ■この本を何でお知りになりましたか？

- |                  |              |             |
|------------------|--------------|-------------|
| 1. 雑誌広告（雑誌名／     | ）            |             |
| 2. 雑誌の紹介記事で（雑誌名／ | ）            |             |
| 3. 書店で見て         | 4. 書店ですすめられて | 5. 人にすすめられて |
| 6. その他（          | ）            |             |

## ■この本をお買上げの書店名は？

都・道  
府・県

区・市

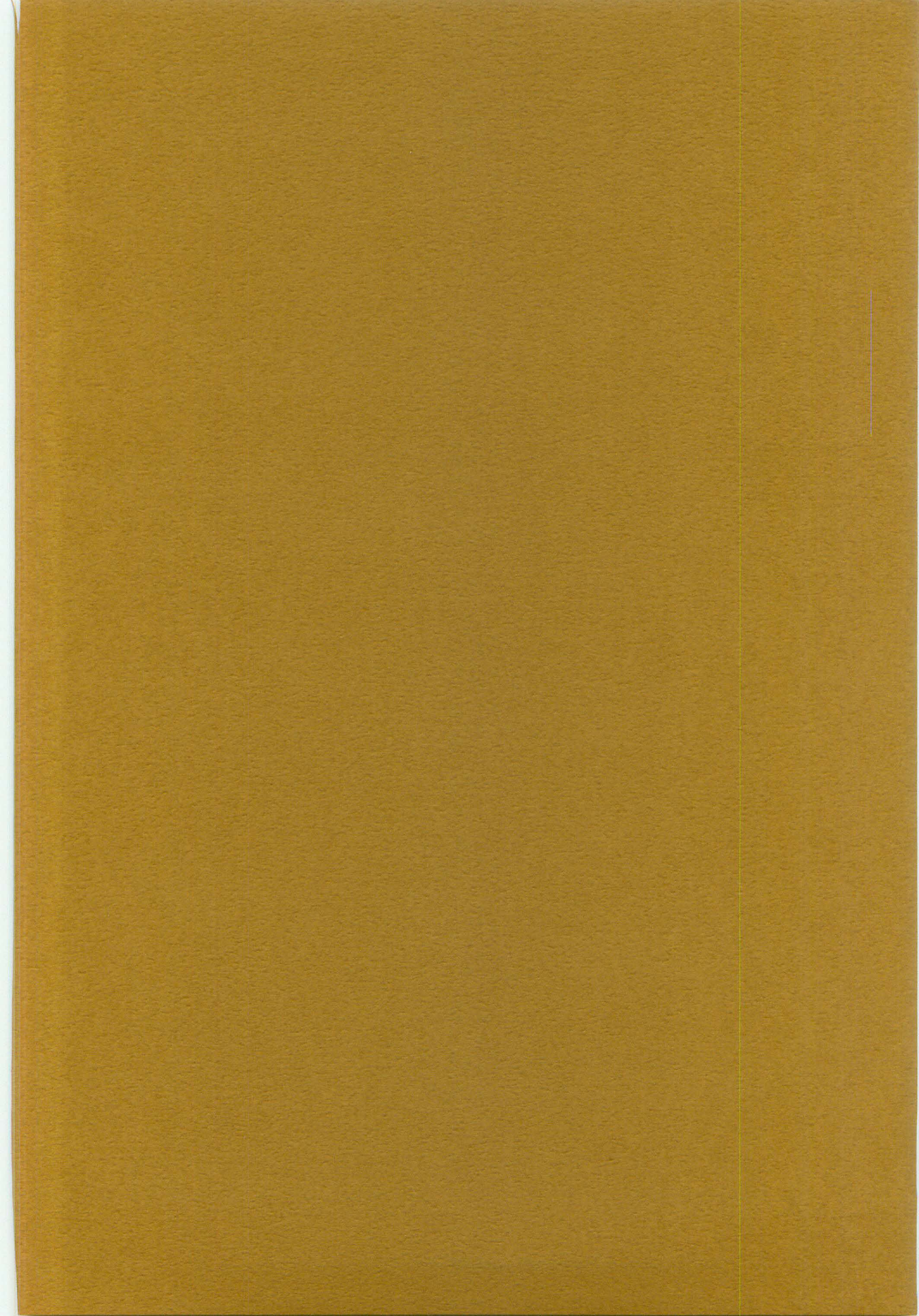
書店

## ■以下の質問にお答え下さい

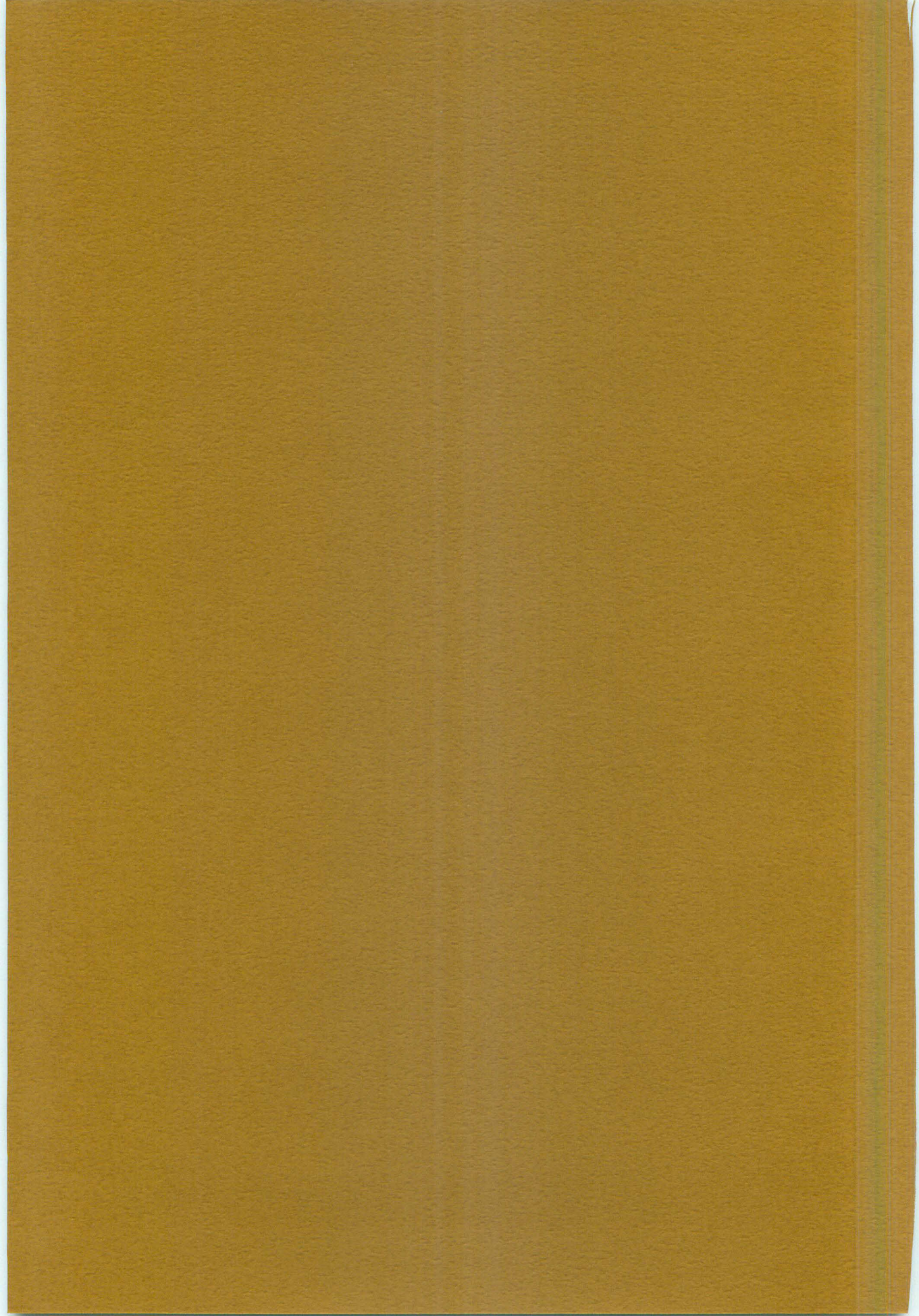
- |        |             |           |           |
|--------|-------------|-----------|-----------|
| 内 容    | 1. わかりやすい   | 2. ふつう    | 3. わかりにくい |
| 装 丁    | 1. よい       | 2. ふつう    | 3. わるい    |
| 価 格    | 1. 高い       | 2. ふつう    | 3. 安い     |
| TeX    | 1. すでに使っている | 2. 使っていない |           |
| パソコン通信 | 1. している     | 2. していない  |           |

## ■お読みになった感想をお聞かせ下さい

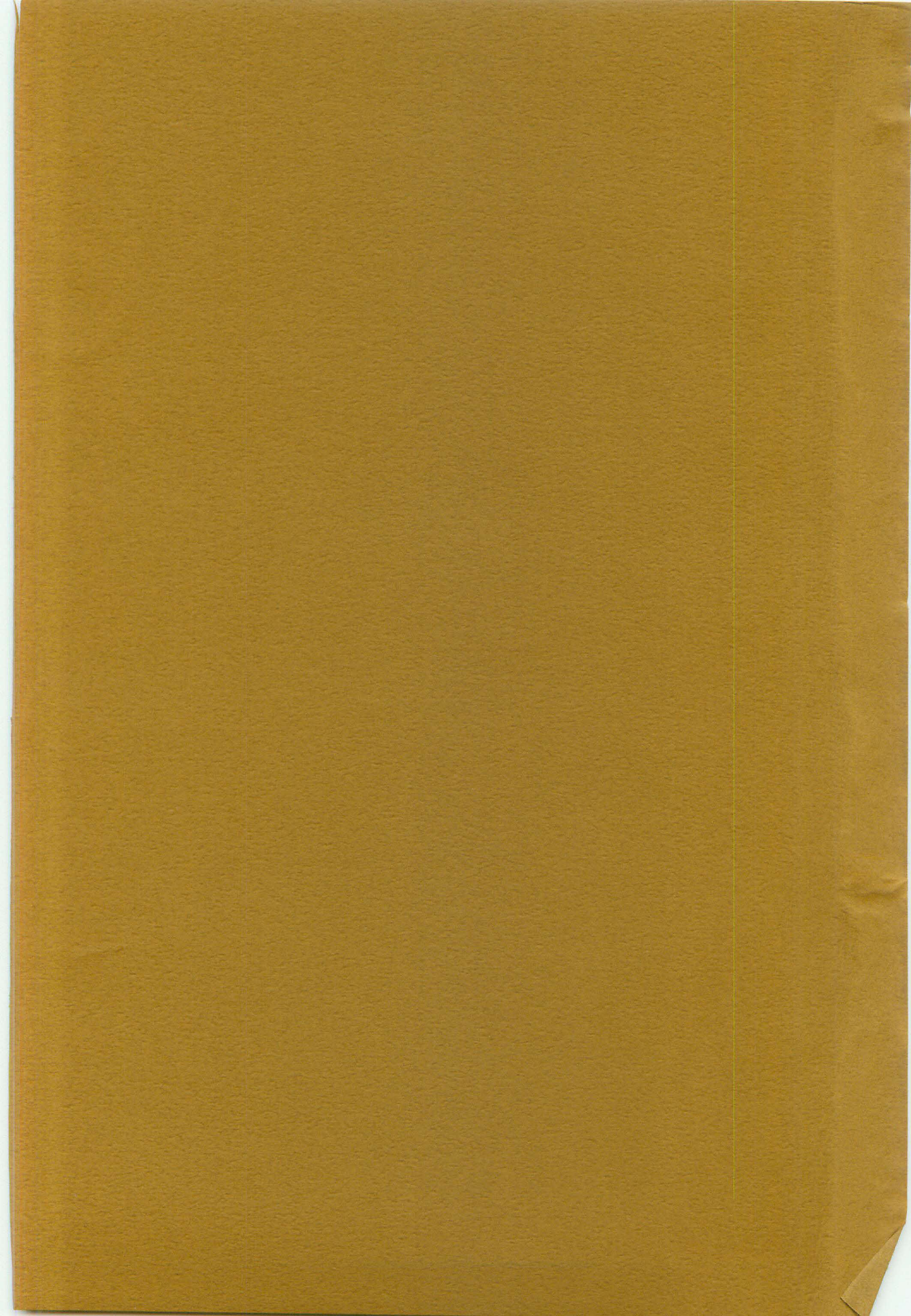
## ■今後どのような企画をお望みですか？













**X 6 8 k**

Programming Series

(#3)

**X680x0  
TEX**