LaTeX 2ε への道

94 分 *LaTeX 2ε* 入門

by Tobias Oetiker

Hubert Partl, Irene Hyna and Elisabeth Schlegl

NOMURA Masataka 訳

Version 1.00, 28 June, 2000

Copyright ©1999 Tobias Oetiker and all the Contributers to LShort. All rights reserved.

Copyright ©2000 NOMURA Masataka とこの冊子に関係した全ての人々．

この冊子はフリーです．Free Software Foundation の GNU General Public License（バージョン 2，もしくは希望すればそれ以降のバージョン）に基づいて，再配布することも修正することも出来ます．

この冊子は，役に立つと思い配布していますが，保証は一切ありません．商売性や特定の目的のための適合性に関する暗黙の保証もありません．詳しくは，GNU General Public License を御覧下さい．

この冊子とともに，GNU General Public License も入手して下さい．できなければ，Free Software Foundation, Inc.（住所：675 Mass Ave, Cambridge, MA 02139, USA）に連絡して下さい．

Copyright ©1999 Tobias Oetiker and all the Contributers to LShort. All rights reserved.

This document is free; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This document is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this document; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

ありがと！

この冊子[訳注 1]に書かれている内容の多くは，オーストリアで以下の人たちによってドイツ語で書かれたLaTeX 2.09の入門書を元にしています．

Hubert Partl `<partl@mail.boku.ac.at>`
Zentraler Informatikdienst der Universität für Bodenkultur Wien
Irene Hyna `<Irene.Hyna@bmwf.ac.at>`
Bundesministerium für Wissenschaft und Forschung Wien
Elisabeth Schlegl `<no email>`
in Graz

ドイツ語で書かれた冊子に興味があれば，Jörg KnappenによってLaTeX 2_εに書き直されたものが`CTAN:/tex-archive/info/lkurz`から入手できます．

この冊子を作成する際，`comp.text.tex`で見直しをお願いし，多くの返答を頂きました．この冊子を改善するために，以下の人たちから訂正，提案，内容に関して連絡を頂きました．彼らは，この冊子を現在の形にするために大いに助けてくれました．彼らすべてに感謝いたします．この冊子の中の誤りはすべて私に責任があります．正しい意味の言葉があったとしたら，それは以下の人たちの誰かが私に連絡してくれたからに違いありません．

Rosemary Bailey, Friedemann Brauer David Carlisle, Christopher Chin, Chris McCormack, Wim van Dam, Michael John Downes, David Dureisseix, Elliot, David Frey, Robin Fairbairns, Erik Frisk, Frank, Alexandre Guimond, Cyril Goutte, Greg Gamble, Neil Hammond, Rasmus Borup Hansen, Martien Hulsen, Werner Icking, Jakob, Eric Jacoboni, Alan Jeffrey, Byron Jones, David Jones, Johannes-Maria Kaltenbach, Andrzej Kawalec, Alain Kessi, Christian Kern, Jörg Knappen, Kjetil Kjernsmo, Maik Lehradt, Alexander Mai, Martin Maechler, Claus Malten, Kevin Van Maren, Lenimar Nunes de AndradeHubert Partl, John Refling, Mike Ressler, Brian Ripley, Young U. Ryu, Bernd Rosenlecher, Chris Rowley, Hanspeter Schmid, Craig Schlenter, Christopher Sawtell, Josef Tkadlec, Didier Verna, Fabian Wernli, Carl-Gustav Werner, Chris York, Fritz Zaucker, Rick Zaccone, and Mikhail Zotov, #TeX.

また，日本語訳に関しては，以下の方々のお世話になりました．

NIDE Naoyuki, #pTeX

[訳注 1] この冊子は，Tobias Oetikerによる「*The not so short introduction to LaTeX 2_ε*」を日本語に訳したものです．

まえがき

LaTeX[1] は，組版の品質が高く，科学・数学分野の文書を作成するのに非常に適した組版システムです．またこれ以外にも LaTeX には，手紙から一冊の本までというあらゆる文書を作成するのにも適しています．LaTeX は，組版エンジン（組版機構）として，TeX[2] を使用しています．

この冊子では LaTeX 2$_{\varepsilon}$について説明し，たいていの文書を LaTeX を使用して作成することができるようになっています．LaTeX についての詳細な記述については，参考文献 [1, 3] を参照して下さい．

LaTeX は，PC，Macintosh から UNIX，VMS システムといった規模までのほとんどのコンピュータ上で使うことができます．多くの大学のコンピュータネットワーク上にも LaTeX がインストール可能であり，すぐに利用することができます．使用しているコンピュータ上での LaTeX の使い方に関しては，ローカルガイド [4] を参照して下さい．LaTeX を始めるにあたり問題が生じたら，この冊子を紹介してくれた人に尋ねてみて下さい．この冊子には，LaTeX のインストールやセットアップの方法については書かれていません．LaTeX を使用して，文書を作成する方法が書かれているのです．

この冊子は，5 つの章から構成されています．

第 1 章 では，LaTeX 2$_{\varepsilon}$で作成する文書の基本的構造について説明しています．また，LaTeX の生い立ちについても簡単に触れています．この章を読めば，LaTeX の概略がわかるでしょう．ここで述べることは LaTeX の概略でしかありませんが，他の章の内容を理解したり，LaTeX の全体を知るのに役立つでしょう．

第 2 章 では，文書を組版するための基本的な LaTeX のコマンドと環境について説明しています．この章を読めば，LaTeX を使った初めての文書を書くことができるようになるでしょう．

第 3 章 では，LaTeX の強力な武器の一つである数式を組版する方法について，ここでも多くの例を示して説明しています．この章の最後に，LaTeX において使用できる数式記号のすべてを表にしています．

第 4 章 では，索引や参考文献の作成方法，EPS（Encapsulated Postscript）ファイルによる図の取り込み方，その他役に立つ事柄について説明しています．

第5章 では，LaTeX による標準的な文書レイアウトを変更するちょっと危なっかしい内容について説明しています．ここを読めば，美しいとされる LaTeX による出力をあきれかえるほどひどいレイアウトに変更する方法がわかります．

この冊子はそんなに膨大なものではないので，最初の章から順に読んで下さい．LaTeX で文書を作成するための多くの情報が，この冊子の中の様々な例題に含まれていますので，例題は注意深く読んでください．

LaTeX に関して必要なものがあれば，Comprehensive TeX Archive Network（CTAN）ftp サイトを覗いて下さい．アメリカでは `ftp://ctan.tug.org/`，ドイツでは `ftp://ftp.dante.de/`，イギリスでは `ftp://ftp.tex.ac.uk/` というサイトとなっています．これらの国以外に住んでいるのであれば，ネットワーク的に近いところの CTAN ミラーサイトを選んで利用して下さい(訳注 2)．

自分が使用しているコンピュータで新たに LaTeX を使用したければ，必要なものを `CTAN:/tex-archive/systems` で調べてインストールして下さい．

この冊子に対して追加，削除，変更すべき事項などがあれば，お知らせ下さい．特にこの冊子のわかりやすい箇所，もっとうまく説明すべき箇所などについて LaTeX 初心者からの連絡をお願いいたします．

Tobias Oetiker　`<oetiker@ee.ethz.ch>`

Department of Electrical Engineering,
Swiss Federal Institute of Technology
スイス連邦工科大学 電気工学科

NOMURA Masataka　`<nomura@cc.kshosen.ac.jp>`

Department of Marine Engineering,
Kobe University of Mercantile Marine
神戸商船大学 商船学部

この冊子の最新バージョンは，`CTAN:/tex-archive/info/lshort/` から入手できます(訳注 3)．

(訳注 2) 日本国内では，

- `ftp://ftp.riken.go.jp/pub/`
- `ftp://ftp.u-aizu.ac.jp/pub/tex/`
- `ftp://ftp.center.osaka-u.ac.jp/`
- `ftp://ftp.iij.ad.jp/pub/TeX/`

などがあります．

(訳注 3) 日本語の最新版は，`http://www-s.eng.kshosen.ac.jp/%7Enomura/hobby/jlshort/` から入手できます．

目 次

図 目 次

表 目 次

第1章　はじめの一歩

この章の最初の節では，LaTeX 2ε とその生い立ちについて簡単に説明します．次の節では，LaTeX 文書の基本的構造を見ていきます．この章を読めば，LaTeX とはどのようなものかという概略を知ることができ，さらに読み進むにつれて出てくるすべての新しい情報から LaTeX を知るのに役立つでしょう．

1.1　はじめに

1.1.1　TeX

TeX は，Donald E. Knuth 教授 [2] が作成したコンピュータプログラムであり，テキストと数式の組版を目指したものです．Knuth 教授は，特に哀れで痛ましい自著の書籍や論文における印刷品質の低下・悪化傾向を改め，その当時，出版界に浸透し始めたコンピュータによる印刷技術の可能性を調べるため，1977 年に組版エンジンである TeX のプログラムを書き始めました．今日使用されているような TeX は 1982 年に公開されたものであり，その後，8 ビット文字や多種多様の言語を扱うために強化・改良されたものが 1989 年に公開されています．TeX は非常に安定しており，様々なコンピュータ上で動作し，実際にほとんどバグがないことで有名なソフトウェアです．TeX のバージョン番号は最終的に π となるのですが，現在は 3.14159 です．

TeX を言葉にするときには，ドイツ語の “Ach” やスコットランド方言の “Loch” における “ch” のように “Tech” と発音します．また ASCII 環境では，TeX のことは `TeX` と表記します．

1.1.2　LaTeX

LaTeX は，あらかじめ決められた本格的なレイアウトに従って最高の品質で文書を組版，出力するためのマクロパッケージであり，TeX を組版エンジンとし，Leslie Lamport 氏 [1] によって作成されました．

1994 年，LaTeX パッケージは Frank Mittelbach 率いる LaTeX3 チームによって更新されました．これは，強く要望のあった様々な改良と LaTeX 2.09 が数年前にリリースされて以来生じていたすべての修正を再統合するためのものでした．旧バージョンと区別するために，新しい LaTeX は LaTeX 2ε と呼ばれます．この冊子は，LaTeX 2ε について説明を行っています．

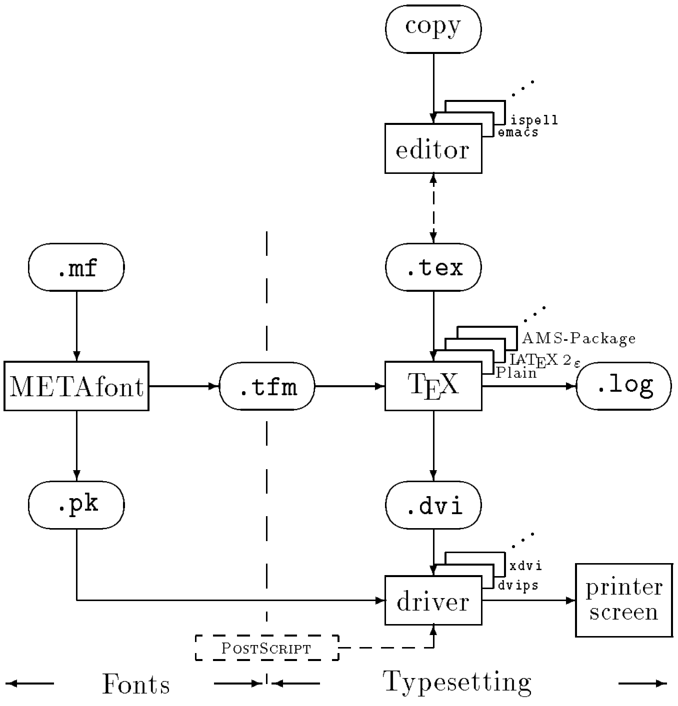


図 1.1: TeX の内堀・外堀

LaTeX は，“Lay-tech” とか “Lah-tech” と発音されます．`ASCII` 環境のように LaTeX と書けないときには，`LaTeX` と書きます．LaTeX 2_{ε}は，“Lay-tech two e” と読み，文字の上げ下げや字詰めを行うことのできない環境では`LaTeX2e` と書きます．

上の図 1.1 は，TeX や LaTeX 2_{ε}がどのように動作しているかを示しています．この図は，Kees van der Laan による `wots.tex` からのものです．

1.2　基本的事項

1.2.1　執筆者，本のデザイナ，組版者

本などを出版する際，執筆者はタイプした印刷用の原稿を出版社に渡します．そして，出版社のデザイナが本のレイアウト（文章幅，フォント，見出しの上下のスペース量などなど）を決定します．デザイナはレイアウトの指示を原稿に書き込み，印刷工に渡します．この指示に従って本が組版されます．

本のデザイナは，原稿執筆中に執筆者が意図していたことを理解し，プロの知識に基づいて原稿に書かれている内容から章見出し，引用文献，例題，数式などのレイアウトを決定します．

LaTeX を使用して組版を行う場合，LaTeX が本のデザイナ，TeX が組版工としての役割を分担します．しかし，LaTeX は単なるプログラムでしかないため，LaTeX が解釈できるようにするために指示を与えなければなりません．つまり執筆者は，作成している文書中に論理的な構造を明示した情報もあわせて与えなければならないのです．この情報というのが，“LaTeX コマンド”として文書中に書き込まれるものなのです．

この LaTeX の方法は，*MS Word* とか *Corel WordPerfect* といった最近のワープロの手法である WYSIWYG[1]とは全く異なっています．これらのアプリケーションソフトでは，執筆者はコンピュータに文書を入力しながら直接文書のレイアウトを決定していきます．このようにして，最終的に印刷されたものがどのようになるかをコンピュータ画面上で確かめることができます．

LaTeX を使うと，文書を入力している際には通常，最終的な出力結果を確認することはとても難しくなります．しかし，LaTeX でファイルを処理（組版）した後であれば，最終的な出力結果をコンピュータの画面上に表示することで確かめることができます．つまり，実際にプリンタから文書を出力する前に修正を行うことができるのです．

1.2.2　レイアウトデザイン

レイアウトのデザインは，技能です．組版知識の未熟な執筆者は，よく重大な体裁間違いを起こします．それは「文書が芸術的にすばらしく見えるのは，うまくデザインされているからだ」というように，本のデザインはたいてい美的感覚の問題だと思っているからです．しかし文書は読まれるべきものであり，美術館などに飾られるものではありません．読みやすさと理解しやすさは，その見栄えよりもずっとずっと重要なのです．例えば，

- 見出しにおける文字の大きさや番号付けは，読者にとって章や節の構造が分かりやすいように選ばれていなければなりません．
- 一行の長さは，紙面が美しく見えるくらいには長く，また読む人の目が疲れないくらいには短くなければなりません．

WYSIWYG システムでは，見た目が良いだけで構造のほとんどない，もしくはあっても矛盾した構造をもった文書を作りがちです．しかし LaTeX では，文書の論理的な構造を執筆者が明示する必要があるため，そういった間違った体裁になるのを防ぎ，最も適したレイアウトを行うことができるのです．

1.2.3　よい利点と悪い点

WYSIWYG システムに慣れた人が LaTeX を使用している人に「いわゆるワープロと比べて LaTeX の良い点，もしくは良くない点」を話題にすること

---

[1] What You See Is What You Get（画面で見ているとおりに印刷される）の略

がよくあります．そんな議論はたいてい手に余るので，始まってしまったら低姿勢を保つのがベストでしょう．しかし，時にはそんな議論から逃れられなくなることも...

そこで LaTeX に有利な情報をここに示しておきましょう．LaTeX がワープロよりも優れている主な点は，以下の通りです．

- まるで本当に“印刷された”ように，本職並みに巧みにレイアウトされた文書が作成できる．
- 簡便な方法で，数式を組版することができる．
- 使用者は，文書の論理的な構造を定めるほんの少しのわかりやすいコマンドを覚えるだけでよく，実際のレイアウトをいじくり回す必要はほとんどありません．
- 脚注，相互参照，目次や参考文献といった複雑な構造でさえ簡単に作成することができます．
- LaTeX の基本配布システムが全く対応していないような組版作業の多くに対して，フリーの追加拡張パッケージが存在しており，このパッケージを使うことによって，PostScript ファイルによる図を文書中に挿入したり，厳格な規格に従う参考文献を組版したりすることができます．これら追加拡張パッケージの多くは，*The LaTeX* コンパニオン [3] に説明されています．
- 執筆者が適切に構造化されたテキストを書くことができるように，LaTeX は手助けします．つまり構造を明記することによって，それに従って LaTeX が動作するのです．
- LaTeX 2_ε の組版エンジンである TeX は，高い可般性があり，また無償で入手できます．したがって，現在稼働しているほとんどのコンピュータ上で動かすことができます．

LaTeX には次のような欠点もあります．私にはそんな にたくさんあるとは思えないのですが，山と挙げる人もいるでしょうね ^_^;．

- LaTeX は，魂を売り渡した人にはたいした効果をもたらさないでしょう...
- あらかじめ定義されている文書レイアウト用にパラメータが調整されていますが，新たに全体のレイアウトをデザインすることは困難で時間がかかります[2]．

---

[2] これは，やがて登場する LaTeX3 システムで採り入れられる基本要素の一つとなるようです．

- 構造化されていない文書や系統立てて整えられていない書類を書くことは難しくなります．
- 最初の一歩を促してはくれますが，ハムスターは決して論理的マークアップの概念を完全に理解しているわけではありません．

1.3 LaTeX の入力ファイル

LaTeX の入力ファイルは，ASCII 形式のプレインテキストファイルです．ですから，入力ファイルの作成にはどんなテキストエディタを使用してもかまいません．入力ファイルには，文書の内容に加えてさらにテキストをどのように組版するかを LaTeX に伝えるコマンドも含んでいます．

1.3.1 スペース

LaTeX では，スペースやタブといった“ホワイトスペース”は，どちらも同じように“スペース”として扱われます．ホワイトスペースは，二つ以上が連続していても一つの“スペース”として扱われます．入力ファイルにおける行頭のホワイトスペースは通常無視されます．また，行末の改行は“スペース”と同様に処理されます．

入力ファイル中の空行は，段落の区切りを表します．空行は，いくつ連続していても一つの空行と見なして処理が行われます．以下に示す例題を見て下さい．左側に示されているのが入力ファイルのテキスト，右側が LaTeX によって組版された出力結果です．

```
It does not matter whether you
enter one or several     spaces
after a word.

An empty line starts a new
paragraph.
```

It does not matter whether you enter one or several spaces after a word.

An empty line starts a new paragraph.

1.3.2 特殊文字

LaTeX では，以下に示す記号は特別な意味を持ち，あらゆるフォントで直接打ち出すことのできない特殊文字となっています．

`$ & % # _ { } ~ ^ \`(訳注 1)

次の例を見てわかるように，これらの記号を文書中に出力するには記号の前にバックスラッシュ（`\`）を付けなければなりません．

(訳注 1)使用しているシステムによっては，このバックスラッシュが¥の表示になっています．

```
\$ \& \% \# \_ \{ \}
```

$ & % # _ { }

また数学記号やアクセント記号など，その他の記号も数多く出力することができます．バックスラッシュ\記号自身は，バックスラッシュを記号の前に付けても出力されません．この記号（`\\`）は，改行を行うコマンドとして使用されているからです[3]．

1.3.3 LaTeX コマンド

LaTeX コマンドは，大文字小文字の区別があり，次に示す二つの形式を採ります．

- コマンドはバックスラッシュ`\`で始まり，アルファベットだけから成る文字列です(訳注 2)．各コマンド名は，スペース，数字もしくはアルファベット以外の文字によって区切られます．
- コマンドは，バックスラッシュとたった一つの特殊文字から作られます．

LaTeX は，コマンドのあとのホワイトスペースは無視するので，このスペースを出力したい場合には，コマンド名の後に`{}`とスペース，もしくはスペースコマンド（`\␣`）を置く必要があります．`{}`は，LaTeX がコマンド名のあとの全てのスペースを無視するのを防ぎます．

```
I read that Knuth divides the
people working with \TeX{} into
\TeX{}nicians and \TeX perts.\\
Today is \today.
```

I read that Knuth divides the people working with TeX into TeXnicians and TeXperts.
Today is 平成 13 年 5 月 25 日.

コマンド名の後に，ブレース（中括弧：`{ }`）に括られた引数が必要なコマンドもあります．コマンド名に続いてブラケット（角括弧：`[ ]`）内に書かれるオプション引数をもつコマンドもあります．

```
You can \textsl{lean} on me!
```

You can *lean* on me!

```
Please, start a new line
right here!\newline
Thank you!
```

Please, start a new line right here!
Thank you!

---

[3] では，どうするか．代わりに`$\backslash$`コマンドを試してみて下さい．このコマンドで \ が出力されます．

(訳注 2)日本語での定義も可能かと思います．

1.3.4 コメント

入力ファイル中に%があると，LaTeX はそこから行末までの文字，改行文字そして次の行頭のすべてのホワイトスペースを無視します．

これは，組版された最終出力には現れないコメントを入力ファイル中に書くために使われます．

```
This is an % stupid
% Better: instructive <----
example: Supercal%
              ifragilist%
    icexpialidocious
```

This is an example: Supercalifragilisticexpialidocious

%は，ホワイトスペースや改行が許されない長い入力行を分割するためにも使われます．

長いコメントを書きたいときには，verbatim パッケージを読み込むことによって `comment` 環境を使用することができます．

```
This is another
\begin{comment}
rather stupid,
but helpful
\end{comment}
example for embedding comments in your document.
```

This is another example for embedding comments in your document.

1.4 入力ファイルの構造

LaTeX が入力ファイルを処理する際には，そのファイルがある構造に従っていることを要求しています．つまり入力ファイルは，以下のコマンドで始まっていなければなりません．

```
\documentclass{...}
```

これは，どんな種類の文書を書こうとしているかを LaTeX に知らせるためのものです．このコマンドの後に，文書全体のスタイルを変えるコマンドを書いたり，LaTeX システムに新しい機能を加えるパッケージを読み込んだりするのです．そのようなパッケージを読み込むためには，

```
\usepackage{...}
```

コマンドを使用します．

これらがすんだ後に[4]，文書本体を

```
\begin{document}
```

---

[4] `\documentclass` と `\begin{document}` の間の領域をプリアンブルと呼びます

コマンドで書き始めるのです．

さて，LaTeX コマンドとともに文章を入力し終えたら，最後に

```
\end{document}
```

コマンドを書き込み，LaTeX に入力ファイルの終わりであることを伝えます．このコマンドより後ろは，何が書かれていても LaTeX は無視します．

図1.2は，LaTeX 2_{ε}で処理を行うための最低限の入力ファイルの内容を示しています．もう少し複雑な入力ファイルを，図1.3に示します．

1.5　文書のレイアウト

1.5.1　文書クラス

入力ファイルを処理するとき，LaTeX は執筆者が作成しようとしている文書の形式をまず最初に知らなければなりません．これは，`\documentclass` コ

```
\documentclass{article}
\begin{document}
Small is beautiful.
\end{document}
```

図1.2: LaTeX 入力ファイルの最小構造

```
\documentclass[a4paper,11pt]{article}
\usepackage{latexsym}
\author{H.~Partl}
\title{Minimalism}
\frenchspacing
\begin{document}
\maketitle
\tableofcontents
\section{Start}
Well, and here begins my lovely article.
\section{End}
\ldots{} and here it ends.
\end{document}
```

図1.3: 実際的な雑誌記事の例

マンドで記述されます．

`\documentclass[options]{class}`

ここで *class* は，作成する文書の形式(クラス)を指定する引数です．表 1.1 は，ここで説明した文書クラスを示しています．LaTeX 2ε は，手紙やスライドなどの文書形式を作成するための別のクラスも用意しています．*option* は，文書クラスで設定されている標準値を変更するためのオプション引数であり，コンマで区切って並べます．標準的な文書クラスにおける一般的なオプション引数を表 1.2 に示します．

例：入力ファイルが次のように書かれているとします．

```
\documentclass[11pt,twoside,a4paper]{article}
```

これは，基本の文字サイズ *11* ポイント，*A4* 用紙に両面印刷用に適したレイアウトを行った *aritcle* クラスの文書を組版することを LaTeX に指示しています．

1.5.2 パッケージ

文書を書いていくと，あらかじめ定義されている LaTeX の機能だけではうまく組版できないことがいろいろと出てきます．文書中に図や色付き文字，別のファイルを取り込みたい場合などには，LaTeX の機能を拡張する必要があります．そのような機能強化をするために読み込む定義ファイルをパッケー

表 1.1: 文書クラス

`article`	：科学雑誌，プレゼンテーション，短い報告書，プログラムの説明書，案内状などの短い文書のためのクラス
`report`	：数章から成る比較的長い報告書，ちょっとした冊子，学位論文などのためのクラス
`book`	：本格的な本のためのクラス
`slides`	：スライドのためのクラス．このクラスは，san serif の大きな文字で出力を行います．代わりに，FoilTeX[a] の使用を考える人もいるかもしれません．

---

[a] CTAN:/tex-archive/macros/latex/packages/supported/foiltex

表 1.2: クラスオプション

`10pt`, `11pt`, `12pt` ：文書の基本となる文字の大きさを指定します．オプションが指定されていなければ，`10pt` が指定されたものとみなされます．

`a4paper`, `letterpaper`, ... ：用紙の大きさを指定します．何も指定しなければ，`letterpaper` となります(訳注 3)．その他に，`a5paper`，`b5paper`，`executivepaper`，`legalpaper` が指定できます．

`fleqn` ：別行立ての数式を中央ではなく，左端で揃えて組版します．

`leqno` ：数式番号を数式の右側ではなく，左側に付けます．

`titlepage`, `notitlepage` ：表題を独立したページにするかしないかを指定します．オプションを指定しなければ，表題は `aritcle` クラスでは独立したページにはなりませんが，`report`，`book` クラスでは独立したページになります．

`twocolumn` ：文書を二段組で組むことを LaTeX に指示します．

`twoside, oneside` ：両面印刷用の組版を行うかどうかを指定します．オプションを指定しなければ，`article` と `report` クラスでは片面印刷用，`book` クラスでは両面印刷用の組版が行われます．このオプションは，文書の組版形式のみに関連しており，`twoside` オプションを指定したからと言って，プリンタに両面印刷を実行させるわけではないことに注意して下さい．

`openright, openany` ：章の始まりを奇数ページのみからにするかどうかを指定します．これは，章に関するコマンドをもたない `article` クラスでは意味がありません．オプションを指定しなければ，`report` クラスでは次のページから，`book` クラスでは奇数ページから章が始まります．

(訳注 3) ASCII が日本語化した LaTeX では，`a4paper` になるようです．

ジと呼びます．パッケージは，以下のコマンドを用いて読み込みます．

```
\usepackage[options]{package}
```

ここで，*package* はパッケージの名前，*options* はパッケージ特有の機能を引き出すためのオプション引数です．LaTeX 2_εの基本システムに含まれているパッケージもあります（表 1.3 参照）が，たいていのものは，LaTeX の基本システムとは別に配布されています．使用しているコンピュータ上で利用できるパッケージについての詳しい情報は，ローカルガイド [4] をご覧下さい．LaTeX のパッケージに関しては，*The LaTeX* コンパニオン [3] が主要な参考文献です．この本には，数百のパッケージについての説明が LaTeX 2_εをどのように拡張するかの情報とともになされています．

1.6 LaTeX で扱うファイル

LaTeX を使うときには，様々な拡張子を持つけれどもさっぱり意味のわからないファイルに混乱することでしょう．以下のリストは，TeX を使用する際に目にする様々なファイル形式について説明したものです．この表は，すべての拡張子について書かれてはいません．もし必要だと思うものがあれば，ご連絡下さい．

.tex LaTeX や TeX の入力ファイル．`latex` プログラムでタイプセットを行います．

.sty LaTeX のマクロパッケージ．LaTeX 文書中で`\UsePackage` コマンドによって読み込まれるファイル．

.dtx ドキュメント化された TeX ファイル．LaTeX スタイルファイルの主な配布形式．.dtx ファイルをタイプセットすることにより，.dtx ファイルに書かれている LaTeX パッケージのマクロコードの説明書が得られます．

.ins 対応した.dtx ファイルに含まれるファイルを作成するためのファイル．ネットワークで LaTeX パッケージを入手する際には，.dtx ファイルと一緒に.ins ファイルも入手して下さい．.dtx ファイルを処理するために.ins ファイルを LaTeX で処理します．

.cls 文書の形式を定義するクラスファイル．`\documentclass` コマンドで読み込まれます．

以下のファイルは，入力ファイルを LaTeX で処理する際に作成されます．

.dvi デバイスに依存しないファイル．LaTeX でタイプセットを行った際に出力されるファイル．プレビューアで内容を確認したり，`dvips` などのプログラムで印刷を行います．

表 1.3: LaTeX の基本システムに含まれるパッケージ

---

doc：LaTeX の定義ファイルを文書化します．`doc.dtx`[a]と *The LaTeX* コンパニオン [3] に詳細が書かれています．

exscale：数式表示において，拡大縮小を行う拡張フォントが使用できるようになります．詳細については，`ltexscale.dtx` 参照．

fontenc：LaTeX が使用するフォントのエンコーディングを指定します．詳細については，`ltoutenc.dtx` 参照．

ifthen：‘if . . . then . . . else . . . ’ 形式のコマンドが使用できるようになります．詳細については，`ifthen.dtx` もしくは *The LaTeX* コンパニオン [3] 参照．

latexsym：古い LaTeX にあった記号を出力するために，`latexsym` パッケージを読み込みます．詳細については，`latexsym.dtx` もしくは *The LaTeX* コンパニオン [3] 参照．

makeidx：索引を作成するためのコマンドを使用できるようにします．詳細については，本冊子の 4.3 節もしくは *The LaTeX* コンパニオン [3] 参照．

syntonly：組版することなく，入力ファイルを処理します．

inputenc：アスキーコード，ISO Latin-1, ISO Latin-2, 437/850 IBM コードページ，Apple Macintosh，Next，ANSI-Windows やユーザが定義したコードなどの入力ファイルのエンコーディングを指定します．詳細については，`inputenc.dtx` 参照．

---

[a] このファイルは使用しているシステムに存在するはずであり，書き込み許可があるディレクトリで `latex doc.dtx` を実行することによって，説明書となる `dvi` ファイルが作成されます．この表に記述されている他のパッケージファイルについても同様です．

.log 直前のタイプセット時に出力された詳細情報が書かれたログファイル．

.toc 章，節などの見出しを書き出したファイル．次にタイプセットを行う際に読み込まれ，目次を作成するために使われます．

.lof .toc と同様ですが，図の一覧を作成するためのファイル．

.lot .toc と同様ですが，表の一覧を作成するためのファイル．

.aux 一度タイプセットを行った際の情報を次のタイプセットで利用するために作成されるファイル．また，.aux ファイルには相互参照に関する情報も書かれています．

.idx 索引が必要な文書では，LaTeX がこのファイルに索引語を書き出します．`makeindex` プログラムで処理されるファイル．索引に関する詳細については，60 ページの 4.3 節を参照して下さい．

.ind .idx ファイルを `makeindex` プログラムで処理したファイル．次のタイプセットで文書中に読み込まれ索引を出力します．

.ilg `makeindex` プログラム実行時のログファイル．

1.6.1 ページスタイル

LaTeX では，あらかじめ定義されているヘッダ・フッタの組み合わせで三種類のページスタイルが設定できます．以下のコマンドの *style* パラメータで，どのスタイルを使用するか指定します．

```
\pagestyle{style}
```

使用できるページスタイルを表 1.4 に示します．

表 1.4: LaTeX で使用できるページスタイル

`plain` ：ページの下部中央ににページ番号のみを付けて組版します．これがデフォルトです．

`headings` ：各ページ上部に章見出しとページ番号を付けて組版します．ページ下部には何も出力されません（この冊子に使われているスタイルです．）

`empty` ：ページ上部，下部ともに何も出力しません．

次のコマンドで，このコマンドが現れたページのスタイルを変更することができます．

```
\thispagestyle{style}
```

新たにヘッダ・フッタを定義したい場合には，*The LaTeX* コンパニオン [3] もしくは 61 ページの 4.4 節を参照して下さい．

1.7　大規模な文書の作成

大規模な文書を作成する場合，入力ファイルをいくつかに分割することになるでしょう．このようなときには，以下の二つのコマンドが役に立ちます．

filename.tex という名前をもつ別ファイルの内容を読み込みたい箇所に，次のコマンドを使用します．

```
\include{filename}
```

この時 LaTeX は，改ページを行った後に *filename.tex* で指定されたファイルの内容を読み込み組版します．

二つ目のコマンドはプリアンブルで使用するもので，`\include` コマンドに書かれたファイルのうち，実際に読み込むファイルを LaTeX に指示するコマンドです．

```
\includeonly{filename,filename,...}
```

上のコマンドが文書のプリアンブルで実行されると，`\includeonly` コマンドの引数に並べられたファイルに対してのみ`\include` コマンドが実行され，実際に読み込まれます．ファイル名とコンマの間にスペースは入れないようにして下さい．

`\include` コマンドは，改ページを行った後に読み込んだ内容に従って組版を行います．こうすることによって，このファイルを読み込まなくても改ページの位置が変わったりせず常にそのファイルの先頭から新しいページが始まるので，`\includeonly` コマンドを使用するときには助かります．

```
\input{filename}
```

上のコマンドを使用すると，引数に指定されたファイルをその箇所に読み込んで組版します．

LaTeX で文書のエラーチェックを簡単に行うには，syntonly パッケージが利用できます．これは，DVI ファイルなどの出力は行わず，コマンドの使い方や構文が適切かどうかをざっと調べるだけなので，タイプセットが早く終わり，貴重な時間を節約することができます．使い方はきわめて簡単です．

```
\usepackage{syntonly}
\syntaxonly
```

出力を行いたいときには，二行目をコメントアウトする（行頭にパーセント記号を書き加える）だけです．

第2章　テキストの組版

第1章を読むことで，LaTeX 2_εが処理する文章が基本的にどんな構造をしていなければならないか理解できたことでしょう．この章では，実際の文書を作成するために必要な構造について説明していきます．

2.1　テキストと言語構造

文書（現代のDAAC[1]文学を除く）を作成するために重要なことは，考え，情報，知識といったものを読者に伝えることです．執筆者が伝えようとしている考えがうまく構造化されていれば，読者は書かれた内容を理解するのは容易でしょうし，出力レイアウトが内容の論理性と意味的な構造を反映していれば，この構造をより捉えやすいでしょう．

LaTeXは，文書が論理的で意味的に構造化されていなければならないと言う点において，他の組版システムと異なっています．つまり，文書クラスや様々なスタイルファイルで与えられる“規則”に従った出力結果が得られるのです．

LaTeX（そして組版）において最も重要な文章の構成単位は段落です．段落は，論理的で筋道の通った思考や見解を反映すべきレイアウト形態なので，“文章の構成単位”と言うのです．以下の節で，\\コマンドによる改行の仕方，入力ファイル中の空行による段落の作り方を説明しますので，新たなる考えが始まるのであれば新しい段落を始めるべきですし，そうでなければ改行のみを使うべきです．段落にすべきかどうか迷ったら，書いている文章がどのように思考や見解を伝えているか考えて下さい．一貫した考えを述べているのに新たに段落を開始するべきではありません．全く新しい考えが現れたら，段落を改めるべきです．

ほとんどの人は，正しく段落分けされる重要性について全く理解していません．多くの人は段落の意味さえ知りません．特にLaTeXでは，段落の意味を知らなくても段落を区切ることができるので，文書中に数式が書かれる時に特に間違いが生じやすくなります．以下の例を見て，数式の前後に空行（段落区切り）が使われていたり，いなかったりする理由を理解して下さい（もしこれらの例を理解するのに，必要なコマンドについてまだよくわかっていなければ，この章と次の章を読んでから再びここに戻って確かめて下さい．)

---

[1] Different At All Cost：スイスドイツ語のUVA（Um's Verrecken Anders）の訳

```
% Example 1
\ldots when Einstein introduced his formula
\begin{equation}
  e = m \cdot c^2 \; ,
\end{equation}
which is at the same time the most widely known
and the least well understood physical formula.


% Example 2
\ldots from which follows Kirchoff's current law:
\begin{equation}
  \sum_{k=1}^{n} I_k = 0 \; .
\end{equation}

Kirchhoff's voltage law can be derived \ldots


% Example 3
\ldots which has several advantages.

\begin{equation}
  I_D = I_F - I_R
\end{equation}
is the core of a very different transistor model. \ldots
```

段落の次に小さな文章の構成単位は文です．英文では，略語のピリオドの後よりも文の終わりを表すピリオドの後のスペースの方が大きくなっています．LaTeX は，執筆者がどちらのスペースを必要としているのか理解しようとします．LaTeX が間違う時には，どちらが必要なのかを伝えなければなりません．これについては，後ほど説明します．

文書の構造は，文の構造にまで拡張できます．ほとんどの言語はとても複雑な句読点の規則をもっていますが（ドイツ語や英語を含めて）多くの言語で，句点が言葉の流れを一時的に止めるということを覚えていれば，たいていの場合正しい位置にコンマを打つことができます．どこにコンマを打てばよいか定かでない時には，文章を声に出して読み，すべてのコンマで一呼吸おいてみて下さい．この時ぎこちない部分があれば，その場所にあるコンマは削除します．また息継ぎ（や一呼吸おいたり）した方がよいような場所には，コンマを打ちます．

最終的に段落は，章，節，小節などを組み立て，より高いレベルで論理的に構造化されていかなければなりません．しかし，例えば `\section{文章の構造と言語}` と書いた時のレイアウトの効果は明らかなので，これらの高いレベルの構造がどのように使われるのかはほとんど自明なことでしょう．

2.2　改行と改ページ

2.2.1　各行の長さを揃える

本というものは，通常一行の長さがすべて同じ長さになるように組版されています．LaTeX では，段落全体の文章を最適に配置することによって，単語間に必要な改行やスペースを挿入します．必要であれば，うまく一行に収まらない英単語に対してハイフネーションを行います．どのように段落が組版されるのかは，文書クラスによります．通常，段落の最初の行はインデントされ，段落と段落との間に余分なスペースは加えられません．詳しくは，5.3.2 節を参照して下さい．

特別な場合には，LaTeX に対して改行を明示する必要があります．

`\\`，`\newline`

このコマンドは改行を行いますが，新しい段落のレイアウト(訳注 1)は行いません(訳注 2)．

`\\*`

このコマンドは，改行位置での改ページを禁止した改行を行います．

`\newpage`

このコマンドは，新しいページを開始します．

`\linebreak[`n`]`, `\nolinebreak[`n`]`, `\pagebreak[`n`]` and `\nopagebreak[`n`]`

これらのコマンドは，コマンド名が示すとおりの動作を行い，オプション引数 n に 0 から 4 までの整数値を取ることによって動作を変えることができます．出力結果の見栄えがかなり悪いときには，オプション引数 n に 4 より小さな値を指定することによって，これらのコマンド本来の働きを抑えるオプションを LaTeX に知らせることができます．これらの “break” コマンドと “new” コマンドとを混同しないようにして下さい．“break” コマンドを指定すると，次の節で説明するように LaTeX はページの右端や下端を揃えようとして余分なスペースを挿入します．“新しい行（new line）” を始める必要があれば，それに対応したコマンドを使って下さい．コマンド名はわかりますよね!．

LaTeX は，最適な改行位置を見つけだそうとしますが，標準的な方法でうまくいかない場合には，その行を段落の右端に突き出して組版を行います．そんな時 LaTeX は，“overfull hbox” と警告してきます．これは，LaTeX が単語

(訳注 1)段落最初の行のインデントなどのことでしょう．

(訳注 2)ただし，\\を二つ続けては使えません．

の適切なハイフンの位置を探せないときによく生じます[2]．`\sloppy` コマンドを使用すると，改行の基準を多少緩めることができます．`\sloppy` コマンドは，最終的な出力が最適で見栄えのよいものではなくなっても，単語間のスペースを増やすことによって行末が長く右マージンにはみ出るのを防ぎます．この場合には，“underfull hbox”という警告が出ますが，たいていい満足のいかない出力結果しか得られません．`\fussy` コマンドを使用することで，LaTeX 本来の挙動に戻すことができます．

2.2.2　ハイフン

必要があれば，LaTeX はハイフン処理を行います．しかしハイフンの場所が正しくなければ，以下のコマンドを使用して LaTeX に正しいハイフンの位置を伝えます．

`\hyphenation{word list}`

このコマンドは，引数として並べられた単語のうち“-”が付けられた位置のみでハイフンネーションを行うことを指定します．したがって引数は，通常の文字からなる単語，もしくは現在使用中の言語で通常の文字とみなされる記号しか含んではいけません．ハイフネーションの指示は，`\hyphenation` コマンドが現れたときに使用している言語に対して使用されます．これは，文書のプリアンブルでこのコマンドを使用した場合，英語のハイフンネーションに影響すると言うことです．これに対して，`\begin{document}`コマンドの後や babel のような多国語言語をサポートしたパッケージを使用している場合，ハイフンネーションの指示は，babel パッケージで指定されている言語に対して有効になります．

以下の例では，`\hyphenation` コマンドの引数に書かれた“Hyphenation”と同じように，小文字のみからなる“hyphenation”もハイフネーションされます．また，“FORTRAN”，“Fortran”，“fortran”はハイフネーションされません．特殊文字，特殊記号は，引数に使用できません．

例：

```
\hyphenation{FORTRAN Hy-phen-a-tion}
```

`\-`コマンドを使用すると，単語の任意の位置にハイフンを付けることができますが，このコマンドに指定した位置のみでしかハイフネーションできなくなります．LaTeX は，アクセント記号などの特殊文字を含んだ単語のハイフネーションを自動で行うことはできませんので，このコマンドは特殊文字を使用している単語の場合に特に利用されます．

---

[2] LaTeX は，“Overfull hbox”を出力して注意を促しますが，そのような行を見つけるのはいつも簡単とは限りません．`\documentclass` コマンドに `draft` オプションを指定すれば，行の右マージンに注意を促すための太線が出力されます

```
I think this is: su\-per\-cal\-%
i\-frag\-i\-lis\-tic\-ex\-pi\-%
al\-i\-do\-cious
```

I think this is: supercalifragilisticexpialido-
cious

単語の途中でハイフネーションを行って改行しないようにするには，次のコマンドを使用します．

`\mbox{`*text*`}`

このコマンドは，どんな状況でも単語の途中で改行しません．

```
My phone number will change soon.
It will be \mbox{0116 291 2319}.

The parameter
\mbox{\emph{filename}} should
contain the name of the file.
```

My phone number will change soon. It will be 0116 291 2319.

The parameter *filename* should contain the name of the file.

2.3 定義済みの文字列

これまでに説明してきた例の中で，以下のような特別な文字列を出力する簡単な LaTeX コマンドを見てきました．

コマンド	例	説明
`\today`	平成 13 年 5 月 25 日	タイプセット日時を使用言語で出力(訳注 3)
`\TeX`	TeX	あなたの好きな組版システムの名前
`\LaTeX`	LaTeX	おもちゃの名前
`\LaTeXe`	LaTeX 2$_\varepsilon$	LaTeX の現在の姿

2.4 特殊文字と特殊記号

2.4.1 引用符

引用符に，タイプライターにあるような " 記号は使用しないで下さい．通常の出版物では，引用の開始と終了を表す特別な記号があります．LaTeX では，開き引用符には二つの ‘ を，閉じ引用符には二つの ’ を使用します．

```
‘‘Please press the ‘x’ key.’’
```

“Please press the ‘x’ key.”

(訳注 3) ASCII が日本語化した LaTeX では，\西暦コマンドを指定すると「2001 年 5 月 25 日」のように出力されます．

2.4.2 ダッシュとハイフン

LaTeX は，四種類のダッシュを使い分けることができます．連続するダッシュの数によって，三種類のダッシュを出力することができます．四つ目の記号は実際にはダッシュではなく，数式におけるマイナス記号です．

```
daughter-in-law, X-rated\\
pages 13--67\\
yes---or no? \\
$0$, $1$ and $-1$
```

daughter-in-law, X-rated
pages 13–67
yes—or no?
0, 1 and -1

これらのダッシュの名前は，‘-’ ハイフン，‘–’ エンダッシュ，‘—’ エムダッシュ，‘−’ マイナス記号です．

2.4.3 チルダ（ $\sim$ ）

ウェブのアドレスでよく使用される文字にチルダがあります．LaTeX でチルダを出力するために `\~` コマンドを使用すると，実はこのようになり（ ˜ ）求める出力とは異なっているのがわかります．代わりに以下のようにするとよいでしょう．

```
http://www.rich.edu/\~{}bush \\
http://www.clever.edu/$\sim$demo
```

http://www.rich.edu/˜bush
http://www.clever.edu/$\sim$demo

2.4.4 省略記号（ . . . ）

タイプライターでは，コンマもピリオドも他の文字と同じ幅を持っていますが，印刷物においては他の文字よりも幅が狭く，また直前の文字との間隔も狭くなっています．つまり，省略記号のつもりでピリオドを三つ並べてもその間隔が正しくないため，正しい‘省略記号’を出力することはできません．そのめに，LaTeX では以下に示すような省略記号のドットを出力する特別なコマンドがあります．

`\ldots`

```
Not like this ... but like this:\\
New York, Tokyo, Budapest, \ldots
```

Not like this ... but like this:
New York, Tokyo, Budapest, . . .

2.4.5 合字

次に示すように文字の並びによっては，文字を順番に並べて出力するのではなく，文字と文字を組み合わせた特別な記号を実際に使って組版されることがあります．

ff fi fl ffi . . . ではなく ﬀ ﬁ ﬂ ﬃ. . .

いわゆるこれらの合字は，該当する二つの文字の間に`\mbox{}`を挿入することによって避けることができます．これは，二つの単語から成っている単語の場合に必要となることがあります．

```
Not shelfful\\
but shelf\mbox{}ful
```

Not shelfful
but shelfful

2.4.6 アクセント記号と特殊文字

LaTeX は，多くの言語におけるアクセント記号と特殊文字をサポートしています．表 2.1 は，全てのアクセント記号を文字 o に付けたものを示しています．他の文字でも同様にアクセント記号が付けられます．

ただし i や j の上にアクセント記号を付ける場合には，頭の部分にある点を取り除く必要があります．点のない i や j を生成するには，それぞれ `\i`，`\j` と入力します．

```
H\^otel, na\"\i ve, \’el\‘eve,\\
sm\o rrebr\o d, !‘Se\~norita!,\\
Sch\"onbrunner Schlo\ss{}
Stra\ss e
```

Hôtel, naïve, élève,
smørrebrød, ¡Señorita!,
Schönbrunner Schloß Straße

表 2.1: アクセント記号と特殊文字

ò	`\‘o`	ó	`\’o`	ô	`\^o`	õ	`\~o`
ō	`\=o`	ȯ	`\.o`	ö	`\"o`	ç	`\c c`
ŏ	`\u o`	ǒ	`\v o`	ő	`\H o`	o̧	`\c o`
ọ	`\d o`	o̲	`\b o`	o͡o	`\t oo`		
œ	`\oe`	Œ	`\OE`	æ	`\ae`	Æ	`\AE`
å	`\aa`	Å	`\AA`				
ø	`\o`	Ø	`\O`	ł	`\l`	Ł	`\L`
ı	`\i`	ȷ	`\j`	¡	`!‘`	¿	`?‘`

2.5 英語以外の言語のサポート

英語以外の言語で文書を書く必要がある場合，LaTeX を適切に設定しなければならない箇所が二つあります．

1. 全てが自動で作られる文字列[3]を新しい言語に対応させなければなりません．多くの言語では，Johannes Braams が作成した babel パッケージを使用することでこれらの変更に対応することができます．
2. LaTeX は，新しい言語に対するハイフネーションの規則を知る必要があります．LaTeX にハイフネーション規則を教えることは，多少面倒です．使用する言語のハイフネーションパターンを用いて，新たにフォーマットファイルを作りなおさなければならないのです．このことに関しては，より詳しい情報がローカルガイド [4] にあります．

使用しているコンピュータシステムですでに適切な設定がしてあれば，次のコマンドを `\documentclass` コマンドの後で使用することによって，babel パッケージを使用することができます．

```
\usepackage[language]{babel}
```

システムで使用できる *language* は，システムのローカルガイドにも書かれているはずです．babel パッケージは，指定した言語に対する適切なハイフネーション規則を自動的に適用します．使用している LaTeX が，指定した言語のハイフネーション規則に対応していなくても，babel パッケージは組版された文書の見栄えが極端に悪くなるようなハイフネーションは避けてくれるでしょう．

言語によっては，簡単に特殊文字の入力を行うことができるように *babel* パッケージが新しいコマンドを定義しています．例えば，ドイツ語は多くのウムラウト（äöü）使用するので，*babel* パッケージでは `\"o` の代わりに `"o` と入力することで ö を出力することができるようになっています．

直接キーボードから特殊文字を入力することができるコンピュータシステムもあります．LaTeX は，そのように入力された文字も扱うことができます．1994 年 12 月の LaTeX 2ε の公開から，いくつかの入力エンコーディングに対するサポートが LaTeX 2ε の基本配布に含まれています．inputenc パッケージを調べてみて下さい．このパッケージを使用するときには，異なるエンコーディングを使用するため，入力ファイルが他の人のコンピュータ上には表示できない可能性があることを考えておくべきです．例えば，PC 上ではドイツ語のウムラウト ä のコードは 132 ですが，ISO-LATIN 1 を使用している Unix システムでは 228 となってます．そのため，これらの機能は注意して使わなければなりません．

---

[3] 目次，図目次など

フォントのエンコーディングはまた別の問題になります．フォントのエンコーディングは，TeXのフォントにおけるそれぞれの文字の位置を定義したものです．元々のComputer Modern TeXフォントには，古い7ビットのアスキー文字セットである128文字しか含まれていません．そこで，アクセント文字が必要になると，TeXはアクセントなしの通常の文字とアクセント記号とを組み合わせることによってアクセント文字を作り出します．見た目には問題ないのですが，この手法では，アクセント文字を含んだ単語のハイフネーションが自動で行えなくなっています．

ただ，最新のTeXはECフォントを含んでいます．これらのフォントはComputer Modernフォントに似ていますが，ヨーロッパの言語で使われるほとんどのアクセント文字を表す特殊文字を含んでいます．これらのフォントを使えば，英語以外の言語で作成される文書のハイフネーションを行うことができます．ECフォントは，以下のように文書のプリアンブルにfontencパッケージを読み込むことによって使用することができます．

```
\usepackage[T1]{fontenc}
```

2.6　単語間のスペース

組版を行った文書の行の右端を揃えるために，LaTeXは単語間に挿入するスペースを様々な大きさに調節します(訳注 4)．また，文章をより読みやすくするために，文の終わりは多少広めのスペースにします．LaTeXは，ピリオド，疑問符，感嘆符で文の終わりを判断します．大文字の直後のピリオドは通常省略を表すことが多いため，文の終わりとはみなしません．

これとは逆に，大文字の後のピリオドで文を終わらさせたり，小文字で終わる省略記号などを正しく処理させるには，文書の執筆者がそれらを指定しなければなりません．スペースの前にバックスラッシュを置くと，大きさを変化させないスペースを出力します．チルダ‘~’は，大きさが変化せず，加えてその位置での改行を禁止するスペースを作り出します．ピリオドの前の`\@`コマンドは，大文字に続く場合でもこのピリオドで文が終わることを示します．

```
Mr.~Smith was happy to see her\\
cf.~Fig.~5\\
I like BASIC\@. What about you?
```

Mr. Smith was happy to see her
cf. Fig. 5
I like BASIC. What about you?

次のコマンドを使用することで，ピリオドの後のスペースも単語間のスペー

(訳注 4)ここで述べられているのは欧文の場合であり，和文の場合は基本的に，句読点，記号などの前後のスペースなどを調節します．

スと同じにすることができます．

```
\frenchspacing
```

このコマンドは，LaTeX に通常の単語間のスペースよりも大きなスペースをピリオドの後に挿入しないように指示します．これは参考文献では普通に行われていることであり，また英語以外の言語では通常の文章でもそうなっています．`\frenchspacing` コマンドを使用すると，`\@`コマンドは必要なくなります．

2.7　表題，章見出し，節見出し

作成した文書を読者に理解してもらうためにも，章，節，小節に分割した方がよいでしょう．このために LaTeX は，引数に見出しを取る特別なコマンドを用意しています．これらのコマンドを正しい順序で使うことは，文書執筆者の義務です．

以下の見出し用のコマンドは，`article` クラスで使用できるものです．

```
\section{...}           \paragraph{...}
\subsection{...}        \subparagraph{...}
\subsubsection{...}     \appendix
```

`report` と `book` クラスでは，次の二つのコマンドも使用することができます(訳注 5)．

```
\part{...}              \chapter{...}
```

`article` クラスは章（`chapter`）に関する定義がないので，それらの文書を章として加えることで本にまとめることが非常に簡単になっています．見出しの上下のスペース，番号付け，見出しの文字の大きさなどは，LaTeX が自動で設定，出力します．

これらのコマンドのうち，次の二つは多少特殊なコマンドとなっています．

- `\part` コマンドは，章の通し番号に影響しません．
- `\appendix` コマンドは引数を取りません．このコマンドは，章の番号を数字から文字に変更するだけです[4]．

LaTeX は，最後に行った文書のタイプセットの結果から，見出しとそのページ番号を取りだし，目次を作成します．

(訳注 5)私のシステムでは，`article` クラスでも`\part` は使用できます．

[4] article クラスでは，節の番号を変更します．

目次は，この

```
\tableofcontents
```

コマンドが書かれた場所に作成，出力されます．新しく作成された文書の場合，正しい目次を得るために二度タイプセット（“LaTeX 処理”）を行わなければなりません．文書を三度処理しなければならないこともありますが，処理が必要なときには LaTeX が教えてくれます．

上で示されたすべての見出しコマンドには，“アスタリスク付き” コマンドが存在します．“アスタリスク付き” コマンドは，コマンド名の後にアスタリスク $*$ を付けるだけで，目次にも現れず，番号付けもせずに見出しのみを出力します．例えば，`\section{Help}`コマンドは `\section*{Help}`コマンドとした方が適切でしょう．

通常，各見出しは文書中でコマンドの引数に入力された通りに目次に出力されます．しかし，時には見出しがあまりにも長くて，目次にうまく収まらないなど対処に困ることもあります．そこで，実際に出力する見出しの引数の前に，目次にのみ出力される文字列をオプション引数として指定することができるようになっています．

```
\chapter[Read it! It's Exciting]{This is a very long
                  and especially boring title}
```

文書全体の表題は，次のコマンドを使用して出力することができます．

```
\maketitle
```

表題の中身は，`\maketitle` コマンドを使用する前に，以下のコマンドを用いて定義します．

```
\title{...}，\author{...} ，\date{...}（任意）
```

`\author` コマンドの引数には，`\and` コマンドで区切って複数の著者名を並べることができます．

上に示したコマンドの例を，8 ページの図 1.3 に示します．

LaTeX 2_{ε}では上述の見出しコマンドとは別に，`book` クラスにおいてさらに次に示す三つのコマンドが使用できます．

```
\frontmatter，\mainmatter，\backmatter
```

これらは，本の内容を分けるのに利用できます．これらのコマンドは，実際の本で見られるような章見出し，ページ番号となるように出力形式を変更します(訳注 6)．

(訳注 6)百聞は一見にしかず．実際に試してみて下さい．

2.8　相互参照

本，レポート，論文では，図，表，文章の一部を相互参照することがよくあります．LaTeXには，相互参照をするために次のコマンドが用意されています．

`\label{`*marker*`}`，`\ref{`*marker*`}`，`\pageref{`*marker*`}`

ここで，*marker* は文書作成者が指定する識別子です．LaTeXは，`\ref`コマンドを対応する`\label`コマンドで指定された節，小節，図，表，定理の番号に置き換えます．`\pageref`コマンドは，対応する`\label`コマンドが指定されたページ番号を出力します[5]．各見出しの目次作成の場合と同じように，一度行ったタイプセットの結果を用いて番号付けが行われます．

```
A reference to this subsection
\label{sec:this} looks like:
``see section~\ref{sec:this} on
page~\pageref{sec:this}.''
```

A reference to this subsection looks like: “see section 2.8 on page 28.”

2.9　脚注

次のコマンドを使用すると，そのページの下部に脚注を出力することができます．

`\footnote{`*footnote text*`}`

通常`\footnote`コマンドは，参照される単語や文の後に置き[6]ます[7]．

```
Footnotes\footnote{This is
 a footnote.} are often used
by people using \LaTeX.
```

Footnotes[a] are often used by people using LaTeX.

[a] This is a footnote.

2.10　文字の強調

タイプライターを使って文書を作成する際，重要な単語は下線を引いて強調します．しかし出版されているような本では，イタリック体で単語を強調

---

[5] これらのコマンドは，参照している内容がわかっているわけではありません．`\label`コマンドは，最後のタイプセットで自動で生成される番号を保存しているだけなのです．

[6] 「置き」は，よく使われる単語の一つ「置く」の連用形です．

[7] そのため文や文の一部を参照する場合，`\footnote`コマンドはコンマやピリオドの後に置かれます．

してあります(訳注 7)．LaTeX には，文書を強調するために以下のコマンドが用意されています．

```
\emph{text}
```

このコマンドが引数に対してどのような出力を行うかは，前後の文字列の書体によります．

```
\emph{If you use
  emphasizing inside a piece
  of emphasized text, then
  \LaTeX{} uses the
  \emph{normal} font for
  emphasizing.}
```

If you use emphasizing inside a piece of emphasized text, then LaTeX uses the normal *font for emphasizing.*

文字列の強調とフォントの変更との違いに注意して下さい．

```
\textit{You can also
  \emph{emphasize} text if
  it is set in italics,}
\textsf{in a
  \emph{sans-serif} font,}
\texttt{or in
  \emph{typewriter} style.}
```

You can also emphasize *text if it is set in italics,* in a *sans-serif* font, or in *typewriter* style.

2.11 環境

```
\begin{name}  text  \end{name}
```

ここで，引数 *name* は環境名です．環境は，使用する順番が守られている限り入れ子にすることができます．

```
\begin{aaa}...\begin{bbb}...\end{bbb}...\end{aaa}
```

以下の節で，主要な環境の説明をしていきます．

2.11.1 様々な箇条書き

`itemize` 環境は，単純な箇条書きに適しています．`enumerate` 環境は番号付けされた箇条書きに，`description` 環境は見出し項目を付けた箇条書きを出力するのに使用されます．

(訳注 7)これは欧文の場合で，和文ではゴシック体になるでしょうか．

```
\flushleft
\begin{enumerate}
 \item You can mix the list
       environments to your taste:
 \begin{itemize}
  \item But it might start to
        look silly.
  \item[-] With a dash.
 \end{itemize}
 \item Therefore remember:
 \begin{description}
  \item[Stupid] things will not
        become smart because they
        are in a list.
  \item[Smart] things, though, can
        be presented beautifully in
        a list.
 \end{description}
\end{enumerate}
```

1. You can mix the list environments to your taste:
    - But it might start to look silly.
    - \- With a dash.
2. Therefore remember:

    Stupid things will not become smart because they are in a list.

    Smart things, though, can be presented beautifully in a list.

2.11.2　右寄せ，左寄せ，センタリング

`flushleft` 環境と `flushright` 環境は，段落内の文章をそれぞれ左寄せ，右寄せに組みます．`center` 環境は，文章をセンタリングします．改行のために `\\` コマンドを使用しなければ，LaTeX が自動で改行を行います．

```
\begin{flushleft}
 This text is\\ left-aligned.
 \LaTeX{} is not trying to make
 each line the same length.
\end{flushleft}
```

This text is
left-aligned. LaTeX is not trying to make
each line the same length.

```
\begin{flushright}
 This text is right-\\aligned.
 \LaTeX{} is not trying to make
 each line the same length.
\end{flushright}
```

This text is right-
aligned. LaTeX is not trying to make each
line the same length.

```
\begin{center}
 At the centre\\of the earth
\end{center}
```

At the centre
of the earth

2.11.3　様々な引用

`quote` 環境は，引用文，重要句，実例などを示すために使われます．

```
A typographical rule of thumb
for the line length is:
\begin{quote}
 No line should contain more than
 66~characters.

 This is why \LaTeX{} pages have
 such large borders by default.
\end{quote}
That's why multicolumn print is
often used in newspapers.
```

A typographical rule of thumb for the line length is:

> No line should contain more than 66 characters.
>
> This is why LaTeX pages have such large borders by default.

That's why multicolumn print is often used in newspapers.

さらに引用には，`quotation` 環境と `verse` 環境というよく似た二つの環境があります．`quotation` 環境は，段落最初の行のインデントを行うので複数の段落にわたる長い引用に使用します．`verse` 環境は，改行が大切な詩の引用で使用します．行末に`\\`を置くか，詩の各行の後に空行を置くことで改行することができます．

```
I know only one English poem by
heart. It is about Humpty Dumpty.
\begin{flushleft}
 \begin{verse}
  Humpty Dumpty sat on a wall:\\
  Humpty Dumpty had a great fall.\\
  All the King's horses and all
  the King's men\\
  Couldn't put Humpty together
  again.
 \end{verse}
\end{flushleft}
```

I know only one English poem by heart. It is about Humpty Dumpty.

> Humpty Dumpty sat on a wall:
> Humpty Dumpty had a great
> fall.
> All the King's horses and all
> the King's men
> Couldn't put Humpty together
> again.

2.11.4　入力通りの出力

`\begin{verbatim}`コマンドと`\end{verbatim}`コマンドとで囲まれた文章は，いかなる LaTeX コマンドも意味をなさず，すべての改行とスペースも含めて，まるでタイプライターで打ち出されたように直接出力されます．

段落内の文章中では，次のコマンドで同じことができます．

\verb+*text*+

+ 記号は区切り記号の単なる例で，* やスペース以外ならどんな文字でも使用できます．この冊子に書かれている多くの例は，このコマンドを使用して出力されています．

```
The \verb|\ldots| command \ldots

\begin{verbatim}
 10 PRINT "HELLO WORLD ";
 20 GOTO 10
\end{verbatim}
```

The `\ldots` command …

```
 10 PRINT "HELLO WORLD ";
 20 GOTO 10
```

```
\begin{verbatim*}
 the starred version of
 the      verbatim
 environment emphasizes
 the spaces   in the text
\end{verbatim*}
```

```
␣the␣starred␣version␣of
␣the␣␣␣␣␣␣verbatim
␣environment␣emphasizes
␣the␣spaces␣␣␣in␣the␣text
```

`\verb` コマンドには，アスタリスクがついたものもあり，同じようにして使うことができます．

```
\verb*|like   this :-) |
```

```
like␣␣␣this␣:-)␣
```

`verbatim` 環境と `\verb` コマンドは，他のコマンドの引数の中では使用できません．

2.11.5　表組み

`tabular` 環境は，横罫や縦罫を使用した素晴らしい表を組版するために使用されます．LaTeX では，表の各項目の欄の幅を自動的に決めてくれます．

`\begin{tabular}{`*table spec*`}`

`tabular` 環境の引数 *table spec* で，表の形式を決めます．各項目を左寄せするには `l` を，右寄せするには `r` を，センタリングを行うには `c` を指定します．また，改行を行い，行末を揃えた段落を含む項目の欄には `p{`*width*`}`を用い，縦罫には`|`を指定します．

`tabular` 環境内では，`&`で次の項目欄に移動し，`\\`で次の行に移ります．この改行の際，`\hline` を使用すると横罫が描かれます．

```
\begin{tabular}{|r|l|}
 \hline
 7C0 & hexadecimal \\
 3700 & octal \\
 11111000000 & binary \\
 \hline \hline
 1984 & decimal \\
 \hline
\end{tabular}
```

7C0	hexadecimal
3700	octal
11111000000	binary
1984	decimal

```
\begin{tabular}{|p{4.7cm}|}
 \hline
 Welcome to Boxy's paragraph.
 We sincerely hope you'll
 all enjoy the show.\\
 \hline
\end{tabular}
```

Welcome to Boxy's paragraph. We sincerely hope you'll all enjoy the show.

各項目の間は，`@{...}`で指定することができます．このコマンドは，各項目間のスペースを取り除き，ブレース（ { } ）内に指定されたコマンドに置き換えます．以下の例で見られるように，小数点をそろえる時などに使用します．また，`@{}`を使って，表中の各項目前後にある余分なスペースを取り除く時などにも使用できます．

```
\begin{tabular}{@{} l @{}}
 \hline
 no leading space\\
 \hline
\end{tabular}
```

no leading space

```
\begin{tabular}{l}
 \hline
 leading space left and right\\
 \hline
\end{tabular}
```

leading space left and right

元々，LaTeX には小数点で数値を揃える[8]方法が用意されていないので，右寄せの整数部と左寄せの小数部の二つの項目をうまく使って実現します．`\begin{tabular}`コマンドの引数内に，`@{.}`コマンドを用い，通常の項目間のスペースを小数点 “.” で置き換えます．これで，小数点の位置を揃えることができます．ただし，表中では小数点の位置に`&`を置くのを忘れないようにして下さい．この際，小数点を含んだ数値の項目欄の上部に付けるラベルには，`\multicolumn` コマンドを使用して，複数の項目を一つの項目にまとめて扱います．

```
\begin{tabular}{c r @{.} l}
 Pi expression      &
  \multicolumn{2}{c}{Value}\\
 \hline
 $\pi$              & 3&1416 \\
 $\pi^{\pi}$        & 36&46  \\
 $(\pi^{\pi})^{\pi}$& 80662&7\\
\end{tabular}
```

Pi expression	Value
π	3.1416
π^{π}	36.46
$(\pi^{\pi})^{\pi}$	80662.7

---

[8] ‘tools’ がシステムにインストールされていれば，dcolumn パッケージを参照下さい．

```
\begin{tabular}{|c|c|}
 \hline
 \multicolumn{2}{|c|}{\textbf{Ene}}\\
 \hline
 Mene& Muh!\\
 \hline
\end{tabular}
```

Ene	
Mene	Muh!

2.12　浮動体

現在，ほとんどの出版物にはたくさんの図や表が掲載されています．これらはページをまたがって出力されないように，特別な扱いが必要になります．一つの方法として，図や表がそのページの余白に収まらないときには常に新しいページを始めてしまうという方法があります．しかし，この方法では見栄えの非常に悪い空白の目立つページを作ってしまうことになります．

この問題を解決するために，図や表がそのページに収まらない場合，その残りのページは文章で埋め，図や表を‘浮動体’として次ページ以降に配置するという方法を採ります．LaTeX には，浮動体として表（table）と図（figure）の二つの環境があります．これら二つの環境を十分に活用するには，LaTeX が内部でどのように浮動体を扱っているのかを大まかにでも理解しておくことが重要になってきます．さもなければ，LaTeX が浮動体を望んだ場所に全く配置してくれないために，欲求不満にもなるでしょう．

では，まず浮動体を扱う LaTeX のコマンドを見てみましょう．

`figure` 環境や `table` 環境内の内容は，浮動体として扱われます．どちらの環境も配置場所を指定するオプションパラメータ *placement specifier* を使用することができます．

\begin{figure}[*placement specifier*]，\begin{table}[*placement specifier*]

この引数（*placement specifier*）は，浮動体を配置する場所を LaTeX に伝えるために使用されるオプションパラメータで，浮動体の配置場所を示す文字の組合せを指定します（表 2.2 参照）．

表は，例えば次のようなコマンドで始められます．

```
\begin{table}[!hbp]
```

配置場所を示した `[!hbp]` は，この表を文書中でこのコマンドが書かれた場所（`h`），ページ下部（`b`），浮動体だけが独立したページ（`p`），もしくは見栄えが多少悪くなっても指定された条件を満たす場所になんとか配置する（`!`）ことを LaTeX に伝えます．もし，*placement specifier* が指定されなければ，標準のクラスファイルでは `[tbp]` が指定されたものとして扱われます．

LaTeX は，浮動体をその配置場所の指定に従って配置しようとします．浮動体がそのページに配置できない場合には，図，表，それぞれの待ち行列に保留されます[9]．新しいページが始まると，まず最初に LaTeX は待ち行列に存在している浮動体で「浮動体だけが独立した」特別なページを作成できるかどうかを調べます．不可能だとわかると，待ち行列中の最初の浮動体はまさに今，文章中に現れたかのように扱われます．つまり，LaTeX はその浮動体の配置場所に指定された文字（もはや配置不可能である ‘h’ の指定は除く）に従って浮動体の再配置を試みます．文章中に現れる新しい浮動体はどれも，適切な待ち行列に加えられていきます．LaTeX は浮動体の種類ごとに，本文中に現れたその順番を厳密に守ります．そのため，配置不能の図が出てくると，それ以降の全ての図を文書の最後に配置してしまうことになるのです．つまり，

> もし浮動体が望んだ場所に出力されないのであれば，二つある浮動体の待ち行列のうちの一つで配置困難になっているたった一つの浮動体があるのです．

少し難しい説明をしてきたので，`table` 環境と `figure` 環境に関する別のコマンドについて説明しておきます．

`\caption{`*caption text*`}`

このコマンドは，浮動体に説明文をつけるものです．通し番号と “図”，“表” といった文字は自動で付けられます．

以下の二つのコマンドは，それぞれ図の目次，表の目次を出力するもので，

表 2.2: 浮動体の配置場所

記号	浮動体を配置する場所...
`h`	この環境が書かれた文書中のまさに *here*（この場所）に配置します．主に小さな浮動体で使用します．
`t`	ページ *top*（上部）に配置します．
`b`	ページ *bottom*（下部）に配置します．
`p`	浮動体だけが独立した特別な *page*（ページ）に配置します．
`!`	浮動体の配置を制約する内部パラメータ[a]の多くを無視して配置を行います．

[a] 1 ページに出力可能な浮動体の最大値など．

---

[9] この行列は，最初に並んだものから取り出される「先入れ先出し」行列です．

`\tablefocontents` コマンドと同じように動作します．

`\listoffigures`，`\listoftables`

これらの目次には，それぞれ図，表の全ての説明文が出力されます．説明文に長い文章を使うような場合には，目次には短い文を出力した方がよいでしょう．これは，`\caption` コマンドのオプション引数に短い説明文を指定するだけで行うことができます．

```
\caption[Short]{LLLLLoooooonnnnnggggg}
```

`\label` コマンドや`\ref` コマンドを使用すれば，文書中で浮動体の番号を参照することもできます．

次の例は，文書中に四角の枠を描き，文書に余白を空けたものです．最終的に図などを貼り付けるための余白を文書中に作り出したければ，以下のコードを試して下さい．

```
Figure~\ref{white} is an example of Pop-Art.
\begin{figure}[!hbp]
 \makebox[\textwidth]{\framebox[5cm]{\rule{0pt}{5cm}}}
 \caption{Five by Five in Centimetres.} \label{white}
\end{figure}
```

上の例では，LaTeX は非常に熱心に（！）図をこの場所（h）に置こうとします[10]．それが無理な場合，ページの下部（b）に配置しようとします．そのページにこの図を置くことができないとなると，この図と，表の待ち行列があればそこから取り出した表を含んだ浮動体だけのページを作ることができるかどうかを決定します．浮動体だけのページを作るには図表が足りない場合，LaTeX は新しいページを起こし，初めて図表が現れたとみなしてもう一度図の配置を行います．

ときには，次のコマンドを使う必要もあるでしょう．

`\clearpage` もしくは `\cleardoublepage`

これらは，待ち行列に残っている全ての浮動体を即座に配置し，新しいページを開始することを LaTeX に命令するコマンドです．`\cleardoublepage` コマンドは，新しいページが奇数ページから始まるようにするコマンドです．

この冊子の後半で，LaTeX 2_ε 文書内に PostScript で描かれた図を挿入する方法を説明します．

---

[10] 図の待ち行列が空の場合．

第3章　数式の組版

準備は整った！この章では，TeX の大きな特徴である数式の組版を扱います．しかし前もって言っておくと，この章では数式の組版に関してほんの一部を扱うにすぎません．多くの人にとっては，ここで述べられていることで十分でしょうが，もしここで説明されていることで必要な数式が書けなくても気を落とさないで下さい．そんな時には，AMS-LaTeX[1]やその他のパッケージで扱われているでしょうから，そちらをご覧下さい．

3.1　概要

LaTeX は，数式を組版するための特別なモードを持っています．段落内の文中における数式は，`\(`と`\)`，`$`と`$`や，`\begin{math}`と`\end{math}`で囲んで入力します．

```
Add $a$ squared and $b$ squared
to get $c$ squared. Or, using
a more mathematical approach:
$c^{2}=a^{2}+b^{2}$
```

Add a squared and b squared to get c squared. Or, using a more mathematical approach: $c^{2} = a^{2} + b^{2}$

```
\TeX{} is pronounced as
$\tau\epsilon\chi$.\\[6pt]
100~m$^{3}$ of water\\[6pt]
This comes from my $\heartsuit$
```

TeX is pronounced as $\tau\epsilon\chi$.

100 m^{3} of water

This comes from my $\heartsuit$

長い数式を出力するには，別行立て数式として組版することが好まれます．そのためには，数式を`\[`と`\]`や，`\begin{displaymath}`と`\end{displaymath}`で囲みます．この形式では，数式に番号が付きません．もし LaTeX を使って数式に番号を付けたいのであれば，`equation` 環境を使用して下さい．

```
Add $a$ squared and $b$ squared
to get $c$ squared. Or, using
a more mathematical approach:
\begin{displaymath}
 c^{2}=a^{2}+b^{2}
\end{displaymath}
And just one more line.
```

Add a squared and b squared to get c squared. Or, using a more mathematical approach:

$$c^{2} = a^{2} + b^{2}$$

And just one more line.

---

[1] `CTAN:/tex-archive/macros/latex/packages/amslatex`

`\label` と`\ref` を使えば，文中で数式番号を参照することができます．

```
\begin{equation}\label{eq:eps}
 \epsilon > 0
\end{equation}
From (\ref{eq:eps}), we gather
\ldots
```

$$\epsilon > 0 \tag{3.1}$$

From (3.1), we gather …

別行立て数式とした場合，文中における数式とは異なったスタイルで組版されることに注意して下さい．

```
$\lim_{n \to \infty}
 \sum_{k=1}^n \frac{1}{k^2}
 = \frac{\pi^2}{6}$
```

$\lim_{n \to \infty} \sum_{k=1}^n \frac{1}{k^2} = \frac{\pi^2}{6}$

```
\begin{displaymath}
\lim_{n \to \infty}
 \sum_{k=1}^n \frac{1}{k^2}
 = \frac{\pi^2}{6}
\end{displaymath}
```

$$\lim_{n \to \infty} \sum_{k=1}^n \frac{1}{k^2} = \frac{\pi^2}{6}$$

数式モード とテキスト モード には，違いがあります．index すうしきモード@数式モード 例えば，数式モード では，

1. 入力中のスペースと改行には特別な意味はありません．スペースの大きさは，数式の形から論理的に導かれるか，`\,`，`\quad`，`\qquad` といった特別なコマンドを使って制御することになります．
2. 空行は許されません．一つの数式は一つだけの段落からなります．
3. 各文字は変数名とみなされ，そのように組版されます．もし数式内で通常の文章を（通常の立体のフォント（Computer Modern Roman）とスペースで）組版したければ，`textrm{...}`コマンドを使用し，出力させたい文字列を引数に指定します．

```
\begin{equation}
 \forall x \in \mathbf{R}:
 \qquad x^{2} \geq 0
\end{equation}
```

$$\forall x \in \mathbf{R}: \qquad x^{2} \geq 0 \tag{3.2}$$

```
\begin{equation}
 x^{2} \geq 0\qquad
 \textrm{for all }x\in\mathbf{R}
\end{equation}
```

$$x^{2} \geq 0 \qquad \textrm{for all } x \in \mathbf{R} \tag{3.3}$$

数学者は，記号の使い方についてとてもうるさいようです．‘ブラックボードボールド体’ を扱うのが，ここでの一般的な例でしょう．これは，amsfonts や amssymb パッケージを読み込んで，`\mathbb` コマンドを使用することで出力することができます．上の例は，次のようになります．

```
\begin{displaymath}
 x^{2} \geq 0\qquad
 \textrm{for all }x\in\mathbb{R}
\end{displaymath}
```

$$x^{2} \geq 0 \qquad \text{for all } x \in \mathbb{R}$$

3.2　数式モードにおけるグループ化

数式モードにおけるコマンド多くは，コマンドに続く一文字に対してしか影響を及ぼさないので，もし数文字に対して作用させたければ，それらの文字をブレース（`{ }`：中括弧）で囲んでグループ化しなければなりません．

```
\begin{equation}
 a^x+y \neq a^{x+y}
\end{equation}
```

$$a^x + y \neq a^{x+y} \tag{3.4}$$

3.3　数式の書き方

この節では，数式を出力する上で使用するコマンドについて説明していきます．使用できる数学記号については，49 ページの第 3.10 節を参照して下さい．

ギリシャ文字の小文字は，`\alpha`，`\beta`，`\gamma`，...のように，また大文字は，`\Gamma`，`\Delta`，...のように入力します[2]．

```
$\lambda,\xi,\pi,\mu,\Phi,\Omega$
```

$\lambda, \xi, \pi, \mu, \Phi, \Omega$

上付きと下付きの添字は，それぞれ `^` と `_` を用います．

```
$a_{1}$ \qquad $x^{2}$ \qquad
$e^{-\alpha t}$ \qquad
$a^{3}_{ij}$\\
$e^{x^2} \neq {e^x}^2$
```

$a_{1} \qquad x^{2} \qquad e^{-\alpha t} \qquad a^{3}_{ij}$

$e^{x^{2}} \neq {e^{x}}^{2}$

平方根を出力するには，`\sqrt` コマンドを使用します．n 乗根を出力するには，オプション引数を用いて `\sqrt[`n`]` のようにします．平方根の記号の大きさは，LaTeX が自動的に決定してくれます．もし根号記号のみが必要になれば，`\surd` コマンドを使用して下さい．

```
$\sqrt{x}$ \qquad
$\sqrt{ x^{2}+\sqrt{y} }$
\qquad $\sqrt[3]{2}$\\[3pt]
$\surd[x^2 + y^2]$
```

$\sqrt{x} \qquad \sqrt{x^{2}+\sqrt{y}} \qquad \sqrt[3]{2}$

$\surd[x^2 + y^2]$

---

[2] 大文字の Alpha は，通常のローマン体の A と同じ字体なので，LaTeX 2ε では定義されていません．新しい数式用のコードが定義されれば，変わるかもしれません．

`\overline` コマンドと `\underline` コマンドは，それぞれ数式の上と下に水平な直線を引くコマンドです．

```
$\overline{m+n}$
```

$\overline{m+n}$

`\overbrace` コマンドと `\underbrace` コマンドは，数式を囲むような水平なブレース（中括弧）をそれぞれ数式の上と下に書くコマンドです．

```
$\underbrace{ a+b+\cdots+z }_{26}$
```

$\underbrace{a+b+\cdots+z}_{26}$

変数の上に小さな矢印やチルダのような数式アクセント記号を付けるには，49 ページの表 3.1 に示すようなコマンドを使用します．数文字を覆うような幅広の山形記号やチルダを出力するには，`\widetilde` コマンドや `\widehat` コマンドを使用します．’記号は，プライム記号を出力します．

```
\begin{displaymath}
 y=x^{2}\qquad y'=2x\qquad y''=2
\end{displaymath}
```

$$y=x^{2}\qquad y'=2x\qquad y''=2$$

ベクトルは，変数の上部に小さな矢印記号を添えることで表します．このためのコマンドが `\vec` コマンドです．点 A から点 B までのベクトルを出力するために，`\overrightarrow` と `\overleftarrow` という二つのコマンドがあります．

```
\begin{displaymath}
 \vec a\quad\overrightarrow{AB}
\end{displaymath}
```

$$\vec a\quad\overrightarrow{AB}$$

$\log$ などの関数名は，変数を表すイタリック体ではなく立体で表示されるのが普通です．そこで LaTeX では，主要なほとんどの関数名を出力するために以下のコマンドが用意されています．

```
\arccos  \cos   \csc  \exp  \ker     \limsup  \min  \sinh
\arcsin  \cosh  \deg  \gcd  \lg      \ln      \Pr   \sup
\arctan  \cot   \det  \hom  \lim     \log     \sec  \tan
\arg     \coth  \dim  \inf  \liminf  \max     \sin  \tanh
```

```
\[\lim_{x \rightarrow 0}
\frac{\sin x}{x}=1\]
```

$$\lim_{x\rightarrow 0}\frac{\sin x}{x}=1$$

剰余関数には，二項演算子“$a \bmod b$”を表す `\bmod` と“$x \equiv a \pmod{b}$”の形式で出力する `\pmod` の二つのコマンドがあります．

分数を出力するには，`\frac{...}{...}`コマンドを使用します．$1/2$ の出力形式も，‘分母’，‘分子’ の桁数が少ない場合に見栄えがよいのでよく使用されます．

```
$1\frac{1}{2}$~hours
\begin{displaymath}
 \frac{ x^{2} }{ k+1 }\qquad
 x^{ \frac{2}{k+1} }\qquad
 x^{ 1/2 }
\end{displaymath}
```

$1\frac{1}{2}$ hours

$$\frac{x^{2}}{k+1}\qquad x^{\frac{2}{k+1}}\qquad x^{1/2}$$

二項係数や同じように横線のない分数を出力するには，`{... \choose ...}` コマンド，`{... \atop ...}` コマンドを使用します．後者のコマンドは，括弧が出力されないだけで前者のコマンドと同じような出力になります（amsmath パッケージでは，これらの古い形式のコマンドを使用することをはっきりと禁止しており，これらのコマンドは `\binom` コマンドと `\genfrac` コマンドに置き換えられています．後者は様々な分数構造を定義できるコマンドになっています．例えば，`\newcommand{\newatop}[2]{\genfrac{}{}{0pt}{1}{#1}{#2}}`の定義で，`\atop` コマンドと同じ構造を出力することができます．）

```
\begin{displaymath}
 {n \choose k}\qquad {x \atop y+2}
\end{displaymath}
```

$${n \choose k}\qquad {x \atop y+2}$$

関係項演算子などでは，演算子の上に記号などを重ねて出力することがよくあります．`\stackrel` コマンドは，二番目の引数を通常の位置に出力し，その上部に上付き添字の文字サイズで最初の引数で与えられた記号を出力します．

```
\begin{displaymath}
 \int f_N(x) \stackrel{!}{=} 1
\end{displaymath}
```

$$\int f_N(x) \stackrel{!}{=} 1$$

積分記号は `\int` コマンド，総和記号は `\sum` コマンド，また積記号は `\prod` コマンドで出力することができます．この場合，下限，上限は下付き添字，上付き添字を出力するためのコマンド `^` と `_` を用います[3]．

```
\begin{displaymath}
 \sum_{i=1}^{n} \qquad
 \int_{0}^{\frac{\pi}{2}} \qquad
 \prod_\epsilon
\end{displaymath}
```

$$\sum_{i=1}^{n} \qquad \int_{0}^{\frac{\pi}{2}} \qquad \prod_\epsilon$$

TEX には，括弧やその他の区切り記号を表すために，様々なコマンドが用意されています（例えば，$[\ \langle\ \|\ \updownarrow$ など）．パーレン（( )：括弧）やブラケッ

---

[3] さらに AMS-LATEX では，複数行にわたる上限，下限を指定することができるようになっています．

ト（[ ]：角括弧）は対応するキーを直接入力することで，ブレース（{ }：中括弧）は`\{ \}`で，またその他の区切り記号はそれぞれのコマンド（例えば，`\updownarrow`など）を入力することで出力できます．出力できる区切り記号については，51ページの表3.8を参照して下さい．

```
\begin{displaymath}
 {a,b,c}\neq\{a,b,c\}
\end{displaymath}
```

$$a, b, c \neq \{a, b, c\}$$

開き区切り記号の前に`\left`コマンドを，また閉じ区切り記号の前に`\right`コマンドを置けば，TeXは数式の高さに合わせた大きさの区切り記号を自動で打ち出します．ただし，`\left`コマンドはすべて`\right`と対になっていなければなりません．また区切り記号のサイズが正しく決められるのは，同じ行で対の区切り記号が出力される場合のみです．もし右側の区切り記号が必要ない場合には，目に見えない区切り記号である‘`\right.`’を使うことになります(訳注1)．

```
\begin{displaymath}
 1 + \left( \frac{1}{ 1-x^{2} }
     \right) ^3
\end{displaymath}
```

$$1 + \left(\frac{1}{1-x^{2}} \right)^3$$

場合によっては，区切り記号の正しい大きさをLaTeXに指示する必要があります．そのためのコマンドに，`\big`，`\Big`，`\bigg`，`\Bigg`があります．これらのコマンドは区切り記号を表すコマンドの前に書きます[4]．

```
$\Big( (x+1) (x-1) \Big) ^{2}$\\
$\big(\Big(\bigg(\Bigg($\quad
$\big\}\Big\}\bigg\}\Bigg\}$\quad
$\big\|\Big\|\bigg\|\Bigg\|$
```

$\Big((x+1) (x-1) \Big)^{2}$

$\big(\Big(\bigg(\Bigg($ $\quad$ $\big\}\Big\}\bigg\}\Bigg\}$ $\quad$ $\big\|\Big\|\bigg\|\Bigg\|$

数式中で三点リーダを出力するには，いくつかのコマンドが使用できます．`\ldots`コマンドはベースライン上に，また`\cdots`コマンドは中央の高さにそれぞれ水平方向に点を三つ描きます．さらに垂直方向には`\vdots`コマンド，斜め方向のドットを出力するには`\ddots`コマンドがあります．3.5節に別の例があります．

```
\begin{displaymath}
 x_{1},\ldots,x_{n} \qquad
 x_{1}+\cdots+x_{n}
\end{displaymath}
```

$$x_{1},\ldots,x_{n} \qquad x_{1}+\cdots+x_{n}$$

(訳注1)左側の区切り記号が不用であれば，‘`\left.`’を使用します．

[4] これらのコマンドは，文字の大きさを変更したり，11pt，12ptといったオプションを指定している場合には正しい大きさにならないことがあります．この場合に，正しい大きさを自動で出力させるためには，exscaleやamsmathといったパッケージを使用して下さい．

3.4　数式モードにおけるスペースの制御

TeX が数式内で決定したスペースの大きさに満足できないときには，以下に示すスペースを制御するコマンドを使用して調整して下さい．`\,` コマンドは $\frac{3}{18}$ quad（␣），`\:` コマンドは $\frac{4}{18}$ quad（␣），`\;` コマンドは $\frac{5}{18}$ quad（␣）の小さなスペースをそれぞれ生成します．`\␣` コマンドは中くらいの大きさのスペースを生成します．また `\quad` コマンド（␣）や `\qquad` コマンド（␣）は大きなスペースを出力するのに使用します．`\quad` コマンドが出力するスペースの大きさは，その時使用しているフォントの ‘M’ の文字幅に一致します．`\!` コマンドは，$-\frac{3}{18}$ quad（␣）の負の大きさのスペースを生成します．

```
\newcommand{\ud}{\mathrm{d}}
\begin{displaymath}
 \int\!\!\!\int_{D} g(x,y)
 \, \ud x\, \ud y
\end{displaymath}
instead of
\begin{displaymath}
 \int\int_{D} g(x,y)\ud x \ud y
\end{displaymath}
```

$$\int\!\!\!\int_{D} g(x,y)\,\mathrm{d}x\,\mathrm{d}y$$

instead of

$$\int\int_{D} g(x,y)\mathrm{d}x\mathrm{d}y$$

微分を表す ‘d’ は，通常ローマン体で出力することに注意して下さい．

AMS-LaTeX には，多重積分記号のそれぞれの積分記号の間のスペースを見栄えよく調節した `\iint`，`\iiint`，`\iiiint`，`\idotsint` といったコマンドが用意されています．amsmath パッケージを読み込むと，上に示した例は以下のように書くことができます．

```
\newcommand{\ud}{\mathrm{d}}
\begin{displaymath}
 \iint_{D} \, \ud x \, \ud y
\end{displaymath}
```

$$\iint_{D}\,\mathrm{d}x\,\mathrm{d}y$$

詳細は（AMS-LaTeX と一緒に配布される）testmath.tex ファイル，もしくは “*The LaTeX* コンパニオン [3]” の第 8 章を参照して下さい．

3.5　数式を垂直方向に揃える

行列を出力するには，`array` 環境を使用します．この環境は，`tabular` 環境と多少似ています．行をあらためるために，`\\` コマンドが使われます．

```
\begin{displaymath}
 \mathbf{X} =
 \left( \begin{array}{ccc}
  x_{11} & x_{12} & \ldots \\
  x_{21} & x_{22} & \ldots \\
  \vdots & \vdots & \ddots
 \end{array} \right)
\end{displaymath}
```

$$\mathbf{X} = \left(\begin{array}{ccc} x_{11} & x_{12} & \ldots \\ x_{21} & x_{22} & \ldots \\ \vdots & \vdots & \ddots \end{array} \right)$$

array 環境を用いると，見えない \right 区切り記号 “.” を使用することによって，以下のような場合分けを表す数式を出力することができます．

```
\begin{displaymath}
 y = \left\{ \begin{array}{ll}
     a  & \textrm{if $d>c$}\\
     b+x& \textrm{in the morning}\\
     l  & \textrm{all day long}
     \end{array} \right.
\end{displaymath}
```

$$y = \left\{ \begin{array}{ll} a & \textrm{if $d>c$}\\ b+x & \textrm{in the morning}\\ l & \textrm{all day long} \end{array} \right.$$

tabular 環境内と同様に，array 環境内でも行列の要素を区切るための線を引くことができます．

```
\begin{displaymath}
 \left(\begin{array}{c|c}
  1 & 2 \\
  \hline
  3 & 4
 \end{array}\right)
\end{displaymath}
```

$$\left(\begin{array}{c|c} 1 & 2 \\ \hline 3 & 4 \end{array}\right)$$

一行のみの数式を出力する equation 環境の代わりに，複数行にわたる数式や複数の数式を並べて出力するための環境として，eqnarray 環境と eqnarray* 環境があります．eqnarray 環境ではそれぞれの行に数式番号が付けられますが，eqnarray* 環境では全く番号が打ち出されません．

eqnarray 環境と eqnarray*環境では，{rcl}という形式を持つ三つの項目欄からなる表と同じように入力します．ここで二番目の項目欄には等号や不等号，もしくはそれ以外の関係演算子などの記号を入力することになります．\\コマンドで改行を行います．

```
\begin{eqnarray}
 f(x) & = & \cos x      \\
 f'(x) & = & -\sin x    \\
 \int_{0}^{x} f(y)dy &
  = & \sin x
\end{eqnarray}
```

$$\begin{array}{rclr} f(x) & = & \cos x & (3.5) \\ f'(x) & = & -\sin x & (3.6) \\ \int_{0}^{x} f(y)dy & = & \sin x & (3.7) \end{array}$$

ここで，等号の左右のスペースがかなり大きくなっているのに注意して下さ

い．このスペースは，次の例に示すように`\setlength\arraycolsep{2pt}`を指定することによって狭くすることができます．

長い数式は，適切な箇所で自動的には分割が行われないので，改行位置とインデントの量を執筆者が指定しなければなりません．以下に示すような二つの方法が，長い数式を出力するためによく利用されます．

```
{\setlength\arraycolsep{2pt}
 \begin{eqnarray}
  \sin x & = & x -\frac{x^{3}}{3!}
           +\frac{x^{5}}{5!}-{}
                       \nonumber\\
  & & {}-\frac{x^{7}}{7!}+{}\cdots
 \end{eqnarray}}
```

$$\begin{aligned} \sin x \;=\; & x - \frac{x^{3}}{3!} + \frac{x^{5}}{5!} - \\ & - \frac{x^{7}}{7!} + \cdots \end{aligned} \qquad (3.8)$$

```
\begin{eqnarray}
 \lefteqn{ \cos x = 1
     -\frac{x^{2}}{2!} +{} }
                   \nonumber\\
 & & {}+\frac{x^{4}}{4!}
     -\frac{x^{6}}{6!}+{}\cdots
\end{eqnarray}
```

$$\begin{aligned} & \cos x = 1 - \frac{x^{2}}{2!} + \\ & \qquad + \frac{x^{4}}{4!} - \frac{x^{6}}{6!} + \cdots \end{aligned} \qquad (3.9)$$

`\nonumber` コマンドは，`eqnarray` 環境中で数式に番号を付けないように LaTeX に指示するコマンドです．

これらの方法で複数行にもわたる数式を見栄えよく揃えることは難しいため，amsmath パッケージではもっと強力な別のコマンドが用意されています（`split` 環境，`align` 環境参照）．

3.6　幽霊

幽霊を見ることはできませんが，依然として多くの人の心の一部を占めています．LaTeX の場合も同じで，スペースに関してちょっとしたイタズラをするために使用されます．

添字を出力するために，`^` や `_` を使用する際，垂直方向に文字を揃えたいことがありますが，少し LaTeX を助けるために`\phantom`コマンドを使うと，引数に指定された文字列が目には見えないけれどもあたかもそこに組まれたように，組版される文書にその文字列分だけのスペースを作り出すことができます．以下の例を見ると使い方がわかるでしょう．

```
\begin{displaymath}
 {}^{12}_{\phantom{1}6}\textrm{C}
 \qquad\textrm{as opposed to}\qquad
 {}^{12}_{6}\textrm{C}
\end{displaymath}
```

$${}^{12}_{6}\mathrm{C} \qquad \textrm{as opposed to} \qquad {}^{12}_{6}\mathrm{C}$$

```
\begin{displaymath}
 \Gamma_{ij}^{\phantom{ij}k}
 \qquad\textrm{as opposed to}\qquad
 \Gamma_{ij}^{k}
\end{displaymath}
```

$$\Gamma_{ij}{}^{k} \qquad \text{as opposed to} \qquad \Gamma_{ij}^{k}$$

3.7　数式の文字サイズ

数式モードでは，LaTeX は文字の配置位置に応じて文字サイズを選択します．例えば，添字は小さな文字で出力されます．数式の一部をローマン体で出力したい場合でも，`\textrm` コマンドは使用しないで下さい．`\textrm` コマンドを使用すると一時的にテキストモードに移行するため，正しい文字サイズを選択することができなくなるためです．文字サイズを正しく変更できるようにするには，`\mathrm` を使用して下さい．しかし，`\mathrm` コマンドは短い文字列にしか対応していないことにも注意しておいて下さい．また，依然としてスペースは無視され，アクセント文字も使用できません[5]．

```
\begin{equation}
 2^{\textrm{nd}} \quad
 2^{\mathrm{nd}}
\end{equation}
```

$$2^{\text{nd}} \quad 2^{\mathrm{nd}} \tag{3.10}$$

LaTeX は自動で文字サイズを決定して出力しますが，それでも正しい文字サイズを伝えなければならないこともあるでしょう．数式モードでは，以下の四つのコマンドで文字サイズを指定することができます．

`\displaystyle` $\displaystyle (123)$，`\textstyle` $\textstyle (123)$，`\scriptstyle` $\scriptstyle (123)$，
`\scriptscriptstyle` $\scriptscriptstyle (123)$

これらのコマンドは，別行立て数式における積分や総和記号の添字の出力形式にも影響を与えます．

```
\begin{displaymath}
 \mathop{\mathrm{corr}}(X,Y)=
  \frac{\displaystyle
   \sum_{i=1}^n(x_i-\overline x)
   (y_i-\overline y)}
  {\displaystyle\biggl[
   \sum_{i=1}^n(x_i-\overline x)^2
   \sum_{i=1}^n(y_i-\overline y)^2
  \biggr]^{1/2}}
\end{displaymath}
```

$$\mathop{\mathrm{corr}}(X,Y)=\frac{\displaystyle\sum_{i=1}^n(x_i-\overline x)(y_i-\overline y)}{\displaystyle\biggl[\sum_{i=1}^n(x_i-\overline x)^2\sum_{i=1}^n(y_i-\overline y)^2\biggr]^{1/2}}$$

上の例は，通常の`\left[`　`\right]` が出力する括弧よりも大きなサイズの括弧を必要とする場合の指定方法の一例でもあります．

---

[5] $\mathcal{A}\kern-.1667em\lower.5ex\hbox{$\mathcal{M}$}\kern-.125em\mathcal{S}$-LaTeX パッケージでは，正しい文字サイズを選択することのできる`\textrm` コマンドが用意されています．

3.8　定理，法則，. . .

数学の文書を書く際には，“補助定理”，“定義”，“公理” などを打ち出すことが多分必要となるでしょう．LaTeX はこのような構造を，以下の形式のコマンドを定義することによって使用することができます．

```
\newtheorem{name}[counter]{text}[section]
```

引数 *name* は，この “定理” 環境を識別するために用いるキーワードです．引数 *text* には，文書に出力する “定理” の実際の名前を指定します．

ブラケット（角括弧）内の引数はオプションで，“定理” で出力される番号を制御するために使用されます．引数 *counter* には，既に定義されている “定理” のキーワードである *name* を指定することで，新しく定義する “定理” 番号を一連の連続した番号として付けることができます．引数 *section* には，節などの番号を付けた “定理” 番号を出力する場合に，そのコマンド名を指定します．

name のキーワードを持つ `\newtheorem` コマンドを文書のプリアンブルで定義すれば，その文書内では以下のようにしてその環境を使用することができます．

```
\begin{name}[text]
This is my interesting theorem
\end{name}
```

この環境は，かなり論理的です．以下に示す例で，最終的に残っている疑問を取り除き，`\newtheorem` 環境は複雑でわかりにくい方法であるということを明らかにしてくれるでしょう．

```
% 文書プリアンブルでの定義
\newtheorem{law}{Law}
\newtheorem{jury}[law]{Jury}
% 文書本体
\begin{law}\label{law:box}
 Don't hide in the witness box
\end{law}
\begin{jury}[The Twelve]
 It could be you! So beware and
 see law~\ref{law:box}\end{jury}
\begin{law}No, No, No\end{law}
```

Law 1 *Don't hide in the witness box*

Jury 2 (The Twelve) *It could be you! So beware and see law 1*

Law 3 *No, No, No*

“Jury” 定理環境は，“Law” 定理環境と同じカウンタを使用しています．つまり “Jury” 定理環境と “Law” 定理環境には，一連の通し番号が付いてます．文書本体で使用する環境のブラケット（角括弧）内のオプション引数は，定理のタイトルなどを付けるのに使用します．

```
\flushleft
\newtheorem{mur}{Murphy}[section]
\begin{mur}
 If there are two or more
 ways to do something, and
 one of those ways can result
 in a catastrophe, then
 someone will do it.\end{mur}
```

Murphy 3.8.1 *If there are two or more ways to do something, and one of those ways can result in a catastrophe, then someone will do it.*

上の “マーフィー” の法則は，現在の節（`section`）番号の付いた番号となっているのがわかります(訳注 2)．同様に，章や小節などの他の見出しレベルと関連づけることもできます．

3.9　数式モードにおけるボールド体

LaTeX を使って，数式内でボールド体の記号を出力するのは簡単ではありません．これは，初心者が乱用するのを防ぐために採られた措置だろうと思われます．数式中でフォントをボールド体に変更する `\mathbf` コマンドは，ボールド体の文字を出力しますがローマン体（立体）であり，数学記号で通常使用するイタリック体のボールド体ではありません．そのためのコマンドとして `\boldmath` コマンドがありますが，これは数式モード内では使用できません．ただし，このコマンドはギリシャ文字などの記号にも適用できます．

```
\begin{displaymath}
 \mu, M \qquad \mathbf{M} \qquad
 \mbox{\boldmath $\mu, M$}
\end{displaymath}
```

$$\mu, M \qquad \mathbf{M} \qquad \boldsymbol{\mu, M}$$

上の例では，望んではいないと思いますがコンマも太文字になっていることに注意して下さい．

より簡単に使用できるように（amsmath パッケージで読み込まれる）amsbsy パッケージでは，`\boldsymbol` コマンドが定義されています．

```
\begin{displaymath}
 \mu, M \qquad
 \boldsymbol{\mu}, \boldsymbol{M}
\end{displaymath}
```

$$\mu, M \qquad \boldsymbol{\mu}, \boldsymbol{M}$$

(訳注 2) 節が新しく始まると定理の番号も再び 1 から始まります．

3.10　数式記号

数式モード中で，通常利用可能な記号を以下の表に示します．

表3.12～3.16に示された記号[6]を使うには，文書のプリアンブルでamssymbパッケージを読み込まなければなりません．また，AMS mathフォントがシステムにインストールされていなければ出力できません．もしAMSパッケージとフォントが使用しているシステムにインストールされていなければ，`CTAN:/tex-archive/macros/latex/packages/amslatex`を見て下さい．

表 3.1: 数式モードにおけるアクセント記号

$\hat{a}$	`\hat{a}`	$\check{a}$	`\check{a}`	$\tilde{a}$	`\tilde{a}`	$\acute{a}$	`\acute{a}`
$\grave{a}$	`\grave{a}`	$\dot{a}$	`\dot{a}`	$\ddot{a}$	`\ddot{a}`	$\breve{a}$	`\breve{a}`
$\bar{a}$	`\bar{a}`	$\vec{a}$	`\vec{a}`	$\widehat{A}$	`\widehat{A}`	$\widetilde{A}$	`\widetilde{A}`

表 3.2: ギリシャ小文字

α	`\alpha`	θ	`\theta`	o	`o`	υ	`\upsilon`
β	`\beta`	ϑ	`\vartheta`	π	`\pi`	ϕ	`\phi`
γ	`\gamma`	ι	`\iota`	ϖ	`\varpi`	φ	`\varphi`
δ	`\delta`	κ	`\kappa`	ρ	`\rho`	χ	`\chi`
ϵ	`\epsilon`	λ	`\lambda`	ϱ	`\varrho`	ψ	`\psi`
ε	`\varepsilon`	μ	`\mu`	σ	`\sigma`	ω	`\omega`
ζ	`\zeta`	ν	`\nu`	ς	`\varsigma`		
η	`\eta`	ξ	`\xi`	τ	`\tau`		

表 3.3: ギリシャ大文字

A	`\mathrm{A}`	H	`\mathrm{H}`	N	`\mathrm{N}`	T	`\mathrm{T}`
B	`\mathrm{B}`	Θ	`\Theta`	Ξ	`\Xi`	Υ	`\Upsilon`
Γ	`\Gamma`	I	`\mathrm{I}`	O	`\mathrm{O}`	Φ	`\Phi`
Δ	`\Delta`	K	`\mathrm{K}`	Π	`\Pi`	X	`\mathrm{X}`
E	`\mathrm{E}`	Λ	`\Lambda`	P	`\mathrm{P}`	Ψ	`\Psi`
Z	`\mathrm{Z}`	M	`\mathrm{M}`	Σ	`\Sigma`	Ω	`\Omega`

---

[6] これらの表は，David Carlisleが作成した`symbols.tex`とJosef Tkadlecが提案し，拡張変更したものです．

表 3.4: 関係演算子

以下に示すそれぞれのコマンドの前に`\not`コマンドを付ければ，対応する否定演算子を打ち出すことができます．

$<$	`<`	$>$	`>`	$=$	`=`
$\leq$	`\leq` or `\le`	$\geq$	`\geq` or `\ge`	$\equiv$	`\equiv`
$\ll$	`\ll`	$\gg$	`\gg`	$\doteq$	`\doteq`
$\prec$	`\prec`	$\succ$	`\succ`	$\sim$	`\sim`
$\preceq$	`\preceq`	$\succeq$	`\succeq`	$\simeq$	`\simeq`
$\subset$	`\subset`	$\supset$	`\supset`	$\approx$	`\approx`
$\subseteq$	`\subseteq`	$\supseteq$	`\supseteq`	$\cong$	`\cong`
$\sqsubset$	`\sqsubset` [a]	$\sqsupset$	`\sqsupset` [a]	$\Join$	`\Join` [a]
$\sqsubseteq$	`\sqsubseteq`	$\sqsupseteq$	`\sqsupseteq`	$\bowtie$	`\bowtie`
$\in$	`\in`	$\ni$	`\ni` , `\owns`	$\propto$	`\propto`
$\vdash$	`\vdash`	$\dashv$	`\dashv`	$\models$	`\models`
$\mid$	`\mid`	$\parallel$	`\parallel`	$\perp$	`\perp`
$\smile$	`\smile`	$\frown$	`\frown`	$\asymp$	`\asymp`
$:$	`:`	$\notin$	`\notin`	$\neq$	`\neq` or `\ne`

[a] この記号を出力するには，latexsym パッケージを使用する必要があります．

表 3.5: 二項演算子

$+$	`+`	$-$	`-`		
$\pm$	`\pm`	$\mp$	`\mp`	$\triangleleft$	`\triangleleft`
$\cdot$	`\cdot`	$\div$	`\div`	$\triangleright$	`\triangleright`
$\times$	`\times`	$\setminus$	`\setminus`	$\star$	`\star`
$\cup$	`\cup`	$\cap$	`\cap`	$\ast$	`\ast`
$\sqcup$	`\sqcup`	$\sqcap$	`\sqcap`	$\circ$	`\circ`
$\vee$	`\vee` , `\lor`	$\wedge$	`\wedge` , `\land`	$\bullet$	`\bullet`
$\oplus$	`\oplus`	$\ominus$	`\ominus`	$\diamond$	`\diamond`
$\odot$	`\odot`	$\oslash$	`\oslash`	$\uplus$	`\uplus`
$\otimes$	`\otimes`	$\bigcirc$	`\bigcirc`	$\amalg$	`\amalg`
$\bigtriangleup$	`\bigtriangleup`	$\bigtriangledown$	`\bigtriangledown`	$\dagger$	`\dagger`
$\lhd$	`\lhd` [a]	$\rhd$	`\rhd` [a]	$\ddagger$	`\ddagger`
$\unlhd$	`\unlhd` [a]	$\unrhd$	`\unrhd` [a]	$\wr$	`\wr`

[a] この記号を出力するには，latexsym パッケージを使用する必要があります．

表 3.6: 大型演算子

$\sum$	`\sum`	$\bigcup$	`\bigcup`	$\bigvee$	`\bigvee`	$\bigoplus$	`\bigoplus`
$\prod$	`\prod`	$\bigcap$	`\bigcap`	$\bigwedge$	`\bigwedge`	$\bigotimes$	`\bigotimes`
$\coprod$	`\coprod`	$\bigsqcup$	`\bigsqcup`			$\bigodot$	`\bigodot`
$\int$	`\int`	$\oint$	`\oint`			$\biguplus$	`\biguplus`

表 3.7: 矢印記号

$\leftarrow$	`\leftarrow` or `\gets`	$\longleftarrow$	`\longleftarrow`	$\uparrow$	`\uparrow`
$\rightarrow$	`\rightarrow` or `\to`	$\longrightarrow$	`\longrightarrow`	$\downarrow$	`\downarrow`
$\leftrightarrow$	`\leftrightarrow`	$\longleftrightarrow$	`\longleftrightarrow`	$\updownarrow$	`\updownarrow`
$\Leftarrow$	`\Leftarrow`	$\Longleftarrow$	`\Longleftarrow`	$\Uparrow$	`\Uparrow`
$\Rightarrow$	`\Rightarrow`	$\Longrightarrow$	`\Longrightarrow`	$\Downarrow$	`\Downarrow`
$\Leftrightarrow$	`\Leftrightarrow`	$\Longleftrightarrow$	`\Longleftrightarrow`	$\Updownarrow$	`\Updownarrow`
$\mapsto$	`\mapsto`	$\longmapsto$	`\longmapsto`	$\nearrow$	`\nearrow`
$\hookleftarrow$	`\hookleftarrow`	$\hookrightarrow$	`\hookrightarrow`	$\searrow$	`\searrow`
$\leftharpoonup$	`\leftharpoonup`	$\rightharpoonup$	`\rightharpoonup`	$\swarrow$	`\swarrow`
$\leftharpoondown$	`\leftharpoondown`	$\rightharpoondown$	`\rightharpoondown`	$\nwarrow$	`\nwarrow`
$\rightleftharpoons$	`\rightleftharpoons`	$\iff$	`\iff` (bigger spaces)	$\leadsto$	`\leadsto` [a]

[a] この記号を出力するには，latexsym パッケージを使用する必要があります．

表 3.8: 区切り記号

<table>
<tr><td>(</td><td><code>(</code></td><td>)</td><td><code>)</code></td><td>$\uparrow$</td><td><code>\uparrow</code></td><td>$\Uparrow$</td><td><code>\Uparrow</code></td></tr>
<tr><td>[</td><td><code>[</code> or <code>\lbrack</code></td><td>]</td><td><code>]</code> or <code>\rbrack</code></td><td>$\downarrow$</td><td><code>\downarrow</code></td><td>$\Downarrow$</td><td><code>\Downarrow</code></td></tr>
<tr><td>$\{$</td><td><code>\{</code> or <code>\lbrace</code></td><td>$\}$</td><td><code>\}</code> or <code>\rbrace</code></td><td>$\updownarrow$</td><td><code>\updownarrow</code></td><td>$\Updownarrow$</td><td><code>\Updownarrow</code></td></tr>
<tr><td>$\langle$</td><td><code>\langle</code></td><td>$\rangle$</td><td><code>\rangle</code></td><td>$\vert$</td><td><code>|</code> or <code>\vert</code></td><td>$\Vert$</td><td><code>\|</code> or <code>\Vert</code></td></tr>
<tr><td>$\lfloor$</td><td><code>\lfloor</code></td><td>$\rfloor$</td><td><code>\rfloor</code></td><td>$\lceil$</td><td><code>\lceil</code></td><td>$\rceil$</td><td><code>\rceil</code></td></tr>
<tr><td>/</td><td><code>/</code></td><td>$\backslash$</td><td><code>\backslash</code></td><td></td><td>. (dual. なし)</td><td></td><td></td></tr>
</table>

表 3.9: 大型区切り記号

$\lgroup$	`\lgroup`	$\rgroup$	`\rgroup`	$\lmoustache$	`\lmoustache`	$\rmoustache$	`\rmoustache`
$\arrowvert$	`\arrowvert`	$\Arrowvert$	`\Arrowvert`	$\bracevert$	`\bracevert`		

表 3.10: その他の記号

$\dots$	`\dots`	$\cdots$	`\cdots`	$\vdots$	`\vdots`	$\ddots$	`\ddots`
$\hbar$	`\hbar`	$\imath$	`\imath`	$\jmath$	`\jmath`	ℓ	`\ell`
$\Re$	`\Re`	$\Im$	`\Im`	$\aleph$	`\aleph`	$\wp$	`\wp`
$\forall$	`\forall`	$\exists$	`\exists`	$\mho$	`\mho` [a]	∂	`\partial`
$'$	`'`	$\prime$	`\prime`	$\emptyset$	`\emptyset`	∞	`\infty`
∇	`\nabla`	$\triangle$	`\triangle`	$\Box$	`\Box` [a]	$\Diamond$	`\Diamond` [a]
$\bot$	`\bot`	$\top$	`\top`	$\angle$	`\angle`	$\surd$	`\surd`
$\diamondsuit$	`\diamondsuit`	$\heartsuit$	`\heartsuit`	$\clubsuit$	`\clubsuit`	$\spadesuit$	`\spadesuit`
$\neg$	`\neg` or `\lnot`	$\flat$	`\flat`	$\natural$	`\natural`	$\sharp$	`\sharp`

[a] この記号を出力するには，latexsym パッケージを使用する必要があります．

表 3.11: 非数学記号

以下の記号は，テキストモード中でも使用できます．

†	`\dag`	§	`\S`	©	`\copyright`
‡	`\ddag`	¶	`\P`	£	`\pounds`

表 3.12: AMS の区切り記号

$\ulcorner$	`\ulcorner`	$\urcorner$	`\urcorner`	$\llcorner$	`\llcorner`	$\lrcorner$	`\lrcorner`

表 3.13: AMS のギリシャ文字とヘブライ文字

$\digamma$	`\digamma`	$\varkappa$	`\varkappa`	$\varGamma$	`\varGamma`	$\varDelta$	`\varDelta`
$\varTheta$	`\varTheta`	$\varLambda$	`\varLambda`	$\varXi$	`\varXi`	$\varPi$	`\varPi`
$\varSigma$	`\varSigma`	$\varUpsilon$	`\varUpsilon`	$\varPsi$	`\varPsi`	$\varPsi$	`\varPsi`
$\varOmega$	`\varOmega`	$\beth$	`\beth`	$\daleth$	`\daleth`	$\gimel$	`\gimel`

表 3.14: AMS の関係演算子

$\lessdot$	`\lessdot`	$\gtrdot$	`\gtrdot`	$\doteqdot$	`\doteqdot` or `\Doteq`
$\leqslant$	`\leqslant`	$\geqslant$	`\geqslant`	$\risingdotseq$	`\risingdotseq`
$\eqslantless$	`\eqslantless`	$\eqslantgtr$	`\eqslantgtr`	$\fallingdotseq$	`\fallingdotseq`
$\leqq$	`\leqq`	$\geqq$	`\geqq`	$\eqcirc$	`\eqcirc`
$\lll$	`\lll` or `\llless`	$\ggg$	`\ggg` or `\gggtr`	$\circeq$	`\circeq`
$\lesssim$	`\lesssim`	$\gtrsim$	`\gtrsim`	$\triangleq$	`\triangleq`
$\lessapprox$	`\lessapprox`	$\gtrapprox$	`\gtrapprox`	$\bumpeq$	`\bumpeq`
$\lessgtr$	`\lessgtr`	$\gtrless$	`\gtrless`	$\Bumpeq$	`\Bumpeq`
$\lesseqgtr$	`\lesseqgtr`	$\gtreqless$	`\gtreqless`	$\thicksim$	`\thicksim`
$\lesseqqgtr$	`\lesseqqgtr`	$\gtreqqless$	`\gtreqqless`	$\thickapprox$	`\thickapprox`
$\preccurlyeq$	`\preccurlyeq`	$\succcurlyeq$	`\succcurlyeq`	$\approxeq$	`\approxeq`
$\curlyeqprec$	`\curlyeqprec`	$\curlyeqsucc$	`\curlyeqsucc`	$\backsim$	`\backsim`
$\precsim$	`\precsim`	$\succsim$	`\succsim`	$\backsimeq$	`\backsimeq`
$\precapprox$	`\precapprox`	$\succapprox$	`\succapprox`	$\vDash$	`\vDash`
$\subseteqq$	`\subseteqq`	$\supseteqq$	`\supseteqq`	$\Vdash$	`\Vdash`
$\Subset$	`\Subset`	$\Supset$	`\Supset`	$\Vvdash$	`\Vvdash`
$\sqsubset$	`\sqsubset`	$\sqsupset$	`\sqsupset`	$\backepsilon$	`\backepsilon`
$\therefore$	`\therefore`	$\because$	`\because`	$\varpropto$	`\varpropto`
$\shortmid$	`\shortmid`	$\shortparallel$	`\shortparallel`	$\between$	`\between`
$\smallsmile$	`\smallsmile`	$\smallfrown$	`\smallfrown`	$\pitchfork$	`\pitchfork`
$\vartriangleleft$	`\vartriangleleft`	$\vartriangleright$	`\vartriangleright`	$\blacktriangleleft$	`\blacktriangleleft`
$\trianglelefteq$	`\trianglelefteq`	$\trianglerighteq$	`\trianglerighteq`	$\blacktriangleright$	`\blacktriangleright`

表 3.15: AMS の矢印記号

$\dashleftarrow$	`\dashleftarrow`	$\dashrightarrow$	`\dashrightarrow`	$\multimap$	`\multimap`
$\leftleftarrows$	`\leftleftarrows`	$\rightrightarrows$	`\rightrightarrows`	$\upuparrows$	`\upuparrows`
$\leftrightarrows$	`\leftrightarrows`	$\rightleftarrows$	`\rightleftarrows`	$\downdownarrows$	`\downdownarrows`
$\Lleftarrow$	`\Lleftarrow`	$\Rrightarrow$	`\Rrightarrow`	$\upharpoonleft$	`\upharpoonleft`
$\twoheadleftarrow$	`\twoheadleftarrow`	$\twoheadrightarrow$	`\twoheadrightarrow`	$\upharpoonright$	`\upharpoonright`
$\leftarrowtail$	`\leftarrowtail`	$\rightarrowtail$	`\rightarrowtail`	$\downharpoonleft$	`\downharpoonleft`
$\leftrightharpoons$	`\leftrightharpoons`	$\rightleftharpoons$	`\rightleftharpoons`	$\downharpoonright$	`\downharpoonright`
$\Lsh$	`\Lsh`	$\Rsh$	`\Rsh`	$\rightsquigarrow$	`\rightsquigarrow`
$\looparrowleft$	`\looparrowleft`	$\looparrowright$	`\looparrowright`	$\leftrightsquigarrow$	`\leftrightsquigarrow`
$\curvearrowleft$	`\curvearrowleft`	$\curvearrowright$	`\curvearrowright`		
$\circlearrowleft$	`\circlearrowleft`	$\circlearrowright$	`\circlearrowright`		

表 3.16: AMS の否定関係演算子と否定矢印記号

$\nless$	`\nless`	$\ngtr$	`\ngtr`	$\varsubsetneqq$	`\varsubsetneqq`
$\lneq$	`\lneq`	$\gneq$	`\gneq`	$\varsupsetneqq$	`\varsupsetneqq`
$\nleq$	`\nleq`	$\ngeq$	`\ngeq`	$\nsubseteqq$	`\nsubseteqq`
$\nleqslant$	`\nleqslant`	$\ngeqslant$	`\ngeqslant`	$\nsupseteqq$	`\nsupseteqq`
$\lneqq$	`\lneqq`	$\gneqq$	`\gneqq`	$\nmid$	`\nmid`
$\lvertneqq$	`\lvertneqq`	$\gvertneqq$	`\gvertneqq`	$\nparallel$	`\nparallel`
$\nleqq$	`\nleqq`	$\ngeqq$	`\ngeqq`	$\nshortmid$	`\nshortmid`
$\lnsim$	`\lnsim`	$\gnsim$	`\gnsim`	$\nshortparallel$	`\nshortparallel`
$\lnapprox$	`\lnapprox`	$\gnapprox$	`\gnapprox`	$\nsim$	`\nsim`
$\nprec$	`\nprec`	$\nsucc$	`\nsucc`	$\ncong$	`\ncong`
$\npreceq$	`\npreceq`	$\nsucceq$	`\nsucceq`	$\nvdash$	`\nvdash`
$\precneqq$	`\precneqq`	$\succneqq$	`\succneqq`	$\nvDash$	`\nvDash`
$\precnsim$	`\precnsim`	$\succnsim$	`\succnsim`	$\nVdash$	`\nVdash`
$\precnapprox$	`\precnapprox`	$\succnapprox$	`\succnapprox`	$\nVDash$	`\nVDash`
$\subsetneq$	`\subsetneq`	$\supsetneq$	`\supsetneq`	$\ntriangleleft$	`\ntriangleleft`
$\varsubsetneq$	`\varsubsetneq`	$\varsupsetneq$	`\varsupsetneq`	$\ntriangleright$	`\ntriangleright`
$\nsubseteq$	`\nsubseteq`	$\nsupseteq$	`\nsupseteq`	$\ntrianglelefteq$	`\ntrianglelefteq`
$\subsetneqq$	`\subsetneqq`	$\supsetneqq$	`\supsetneqq`	$\ntrianglerighteq$	`\ntrianglerighteq`
$\nleftarrow$	`\nleftarrow`	$\nrightarrow$	`\nrightarrow`	$\nleftrightarrow$	`\nleftrightarrow`
$\nLeftarrow$	`\nLeftarrow`	$\nRightarrow$	`\nRightarrow`	$\nLeftrightarrow$	`\nLeftrightarrow`

表 3.17: AMS の二項演算子

$\dotplus$	`\dotplus`	$\centerdot$	`\centerdot`	$\intercal$	`\intercal`
$\ltimes$	`\ltimes`	$\rtimes$	`\rtimes`	$\divideontimes$	`\divideontimes`
$\Cup$	`\Cup` or `\doublecup`	$\Cap$	`\Cap` or `\doublecap`	$\smallsetminus$	`\smallsetminus`
$\veebar$	`\veebar`	$\barwedge$	`\barwedge`	$\doublebarwedge$	`\doublebarwedge`
$\boxplus$	`\boxplus`	$\boxminus$	`\boxminus`	$\circleddash$	`\circleddash`
$\boxtimes$	`\boxtimes`	$\boxdot$	`\boxdot`	$\circledcirc$	`\circledcirc`
$\leftthreetimes$	`\leftthreetimes`	$\rightthreetimes$	`\rightthreetimes`	$\circledast$	`\circledast`
$\curlyvee$	`\curlyvee`	$\curlywedge$	`\curlywedge`		

表 3.18: AMS のその他の記号

$\hbar$	`\hbar`	$\hslash$	`\hslash`	$\Bbbk$	`\Bbbk`
$\square$	`\square`	$\blacksquare$	`\blacksquare`	$\circledS$	`\circledS`
$\vartriangle$	`\vartriangle`	$\blacktriangle$	`\blacktriangle`	$\complement$	`\complement`
$\triangledown$	`\triangledown`	$\blacktriangledown$	`\blacktriangledown`	$\Game$	`\Game`
$\lozenge$	`\lozenge`	$\blacklozenge$	`\blacklozenge`	$\bigstar$	`\bigstar`
$\angle$	`\angle`	$\measuredangle$	`\measuredangle`	$\sphericalangle$	`\sphericalangle`
$\diagup$	`\diagup`	$\diagdown$	`\diagdown`	$\backprime$	`\backprime`
$\nexists$	`\nexists`	$\Finv$	`\Finv`	$\varnothing$	`\varnothing`
$\eth$	`\eth`	$\mho$	`\mho`		

表 3.19: 数式モード中のアルファベット

例	コマンド	必要なパッケージ
ABCdef	`\mathrm{ABCdef}`	
ABCdef	`\mathit{ABCdef}`	
$\mathnormal{ABCdef}$	`\mathnormal{ABCdef}`	
$\mathcal{ABC}$	`\mathcal{ABC}`	
$\mathscr{ABC}$	`\mathcal{ABC}`	eucal（オプション：mathcal）,
	`\mathscr{ABC}`	eucal（オプション：mathscr）
$\mathfrak{ABCdef}$	`\mathfrak{ABCdef}`	eufrak
$\mathbb{ABC}$	`\mathbb{ABC}`	amsfonts もしくは amssymb

第4章　特記事項

大きな文書を作成する際には，索引，参考文献などの作成に LaTeX が大いに役立つでしょう．もっと詳しい説明や高度な使い方については，*LaTeX* マニュアル [1] や *The LaTeX* コンパニオン [3] を参照して下さい．

4.1　EPS ファイルの挿入

LaTeX は，画像や図といった浮動体を扱う基本的機能として，`figure` 環境や `table` 環境を利用します．

標準の LaTeX や LaTeX の拡張パッケージには，実際に図，画像を文書中に取り込む方法がいくつかあります．しかし，ほとんどの人にとってはそんな方法を理解するのは非常に面倒です．そこでこの冊子では，詳しくは説明しません．詳細は，*The LaTeX* コンパニオン [3] または *LaTeX* マニュアル [1] を参照して下さい．

図を文書中に取り込む簡単な方法としては，特別なソフトウェアを使用して図を作成し[1]，それを文書に貼り込む方法があります．もう一度言っておくと，LaTeX には図を文書中に取り込むための方法が多くのパッケージで用意されています．ここでは，扱いが簡単で広く使用されているという理由から Encapsulated PostScript（EPS）による図の取り扱い方についてのみ説明を行います．EPS 形式の図を扱うには，出力のための PostScript プリンタ[2]が必要になります．

図を文書中に取り込むための一連のコマンドは，D. P, Carlisle が作成した graphicx パッケージを読み込むことで使用可能になります．このパッケージは，“graphics” と呼ばれるパッケージ群の一部です[3]．

PostScript プリンタが使用できる環境にあり，使用しているコンピュータシステムに graphicx パッケージがインストールされていれば，次に示す順番に従って文書中に図を取り込むことができます．

1. まず，作成した図を EPS 形式でファイルに保存します[4]．

---

[1] このようなソフトウェアとして，XFig，CorelDraw!，Freehand，Gnuplot などがあります

[2] PostScript を出力する別の方法として，`CTAN:/tex-archive/support/ghostscript` から入手できる GhostScript があります．Windows の使用者であれば，GSvies もあれば便利でしょう．

[3] `CTAN:/tex-archive/macros/latex/packages/graphics`

[4] もし使用しているソフトウェアがファイルの保存に EPS 形式を指定できないのであれば，

2. 次に図を取り込もうとしている文書のプリアンブルに，次のコマンドを書いてgraphicxパッケージを読み込みます．

   `\usepackage[driver]{graphicx}`

   ここでオプション*driver*は，“dviファイルからpostscriptファイルに変換する”ために使用するアプリケーション名です．広く使われているものに，`dvips`があります．TEXでは，文章に図を取り込むための標準方法が決められていないので，使用するドライバ(訳注1)の名前を文書の執筆者がオプションで指定する必要があるのです．*driver*の名前をLATEXが知っていれば，graphicxパッケージは図に関する情報を`.dvi`ファイルに取り込むための正しい方法を選びます．そして，プリンタはその情報を解釈し，`.eps`ファイルを正しく文書中に取り込むことができるようになるのです．
3. 図のファイルを文書中に挿入するために，次のコマンドを使用します．

   `\includegraphics[key=value, ...]{file}`

   オプション引数には，変数*keys*とその値*values*をカンマで区切って複数与えることができます．*key*は，挿入する図の幅，高さ，回転角度を指定するために使用します．表4.1に主な変数を示しておきます．

表4.1: graphicxパッケージで使用できる主な変数名

`width`	文書中に出力する際の図の幅を指定する
`height`	文書中に出力する際の図の高さを指定する
`angle`	右回りの回転角度を指定する
`scale`	図の拡大・縮小率を指定する

次の例を見れば，引数の指定方法がわかるでしょう．

```
\begin{figure}
 \begin{center}
  \includegraphics[angle=90, width=0.5\textwidth]{test}
 \end{center}
\end{figure}
```

PostScript対応のプリンタドライバ（例えば，AppleのLaserWriterなど）をインストールし，このドライバで出力先をファイルにして印刷して下さい．運がよければ，出力されたファイルはEPS形式になっているはずです．ただし，EPSは1ページの図しか扱うことができません．プリンタドライバによっては，EPS形式で出力できるように対応しているものもあります．

(訳注1)出力装置にdviファイルを出力するソフトウェア．

これは，`test.eps` ファイルに描かれている図を文書中に取り込む例ですが，まず最初に図を右回りに 90 度回転させ，次に版面幅の半分になるように図の幅を変更します．この際，高さに関するパラメータは指定されていないので，縦横比は元の図のまま保たれます．幅と高さは，長さの単位を用いて指定することもできます．長さの単位に関しては，74 ページの表 5.5 を参照して下さい．参考文献 [8] や [11] を読めば，図の取り込みに関してもっと詳しく知ることができます．

4.2 参考文献

参考文献を作成するには `thebibliography` 環境を使用し，それぞれの文献を次のコマンドで指定します．

`\bibitem{`*marker*`}`

引数の *marker* は，文書中でその文献を参照する際に使用するキーワードです．また，文献は次のコマンドを用いて参照します．

`\cite{`*marker*`}`

参考文献の番号は，自動で付けられます．`\begin{thebibliography}` 環境の引数で，この番号を出力する最大幅を指定します．以下の例では，`{99}` と指定することによって，二桁を越える番号を持つ文献がないことを LaTeX に伝え，その幅内で番号を揃えて出力します．

```
Partl~\cite{pa} has
 proposed that \ldots

\begin{thebibliography}{99}
 \bibitem{pa} H.~Partl:
  \emph{German \TeX},
  TUGboat Vol.~9, No.~1 ('88)
\end{thebibliography}
```

Partl [1] has proposed that . . .

参考文献

[1] H. Partl: *German TEX*, TUGboat Vol. 9, No. 1 ('88)

長い文書の参考文献を作成するためには，BibTEX が必要になるかもしれません．BibTEX は，たいていの TEX システムと一緒に配布されており，参考文献データベースを読み込み，その中から作成中の文書内で参照したの引用文献を抜き出して参考文献を作成します．BibTEX が生成する参考文献の体裁

表 4.2: 索引の指定例

例	索引の見出し	注釈
`\index{hello}`	hello, 1	簡単な指定方法
`\index{hello!Peter}`	Peter, 3	‘hello’ の副索引
`\index{Sam@\textsl{Sam}}`	*Sam*, 2	索引フォントの変更
`\index{Lin@\textbf{Lin}}`	**Lin**, 7	同上
`\index{Jenny|textbf}`	Jenny, **3**	ページ番号フォントの変更
`\index{Joe|textit}`	Joe, *5*	同上

は，広く使われ確立されたデザインとなるように定義されたスタイルファイルに基づいています．

4.3　索引

多くの本で索引は，とても役に立つものです．LaTeX と `makeindex`[5]プログラムを使用すると，索引を非常に簡単に作成することができます(訳注 2)．ここでは，基本的な索引作成コマンドについてのみ説明します．詳細については，*The LaTeX* コンパニオン [3] を参照して下さい．

LaTeX で索引を作成するには，以下のコマンドを用いて，makeidx パッケージをプリアンブルで読み込まなければなりません．

```
\usepackage{makeidx}
```

そして，次のコマンド

```
\makeindex
```

を入力ファイルのプリアンブルに書くことによって，実際に索引を作成するコマンドが有効になります．

索引は，次のコマンドで作成し，

```
\index{key}
```

key が索引の見出しになります．索引に出力したい用語が文章中に現れるところでこのコマンドを入力します．表 4.2 は，引数 *key* の書き方をいくつかの例と共に示しています．

---

[5] 8 文字を越えるファイル名を扱うことのできないシステムでは，`makeidx` という名前になっていることもあります．

(訳注 2) 日本語による索引作成プログラムには，ASCII の `mendex` があります．

入力ファイルを LaTeX で処理すると，それぞれの `\index` コマンドは，その出現ページと共に適切な索引見出しを作成し，それを特別なファイルに書き出します．ファイル名は，LaTeX が処理する入力ファイルと同じ名前を持ち，拡張子は `.idx` となります．そして，この `.idx` ファイルを `makeindex` で処理します．

```
makeindex filename
```

`makeindex` は，*filename* と同じ名前を持つ拡張子 `.idx` ファイルをソートして，拡張子が `.ind` の索引ファイルを作成します．再び LaTeX で入力ファイルを処理すると，次のコマンドが書かれた場所に，ソートした索引ファイルを読み込んで索引を出力します．

```
\printindex
```

LaTeX 2_εで導入された showidx パッケージを用いると，入力した index コマンドの引数を全て文書の左マージンに出力します．これは，文書の校正や索引の確認を行うのにとても便利です．

4.4 凝ったヘッダ

Piet van Oostrum による fancyhdr パッケージ[6]を使用すると，いくつかの簡単なコマンドで，文書のヘッダやフッタを自由に設定することができます．このページ上部を見れば，このパッケージでできることがわかると思います．

ヘッダやフッタを設定するときの問題点は，ヘッダ等に書くための節や章の見出しの設定方法です．LaTeX は，これを二段階の方法で実現しています．まず，ヘッダやフッタの定義段階では，現在の章と節の見出しをそれぞれ `\rightmark` コマンドと `\leftmark` コマンドを用いて表します．この二つのコマンドの値は，`\chapter` コマンドや `\section` コマンドが処理される際に，常に書き換えられます．

柔軟にヘッダやフッタを設定できるように，`\chapter` コマンドなどでは `\rightmark` コマンドや `\leftmark` コマンドを直接書き換えることはできないようになっています．`\rightmark` コマンドや `\leftmark` コマンドを書き換えるには，`\chaptermark`，`\sectionmark`，`\subsectionmark` といった別のコマンドが `\chapter` コマンドなどから呼び出されるのです．

そのため，ヘッダに書かれる章見出しの出力様式を変更したい場合には，`\chaptermark` コマンドを再定義するだけでよいのです．

図 4.1 は，fancyhdr パッケージを使用して，この冊子と同じヘッダを作成する例を示しています．とにかく，脚注で示したところからパッケージ一式を入手し，その説明書を読んでみて下さい．

---

[6] `CTAN:/tex-archive/macros/latex/contrib/supported/fancyhdr` より入手可能です．

```
\documentclass{book}
\usepackage{fancyhdr}
\pagestyle{fancy}
% 上のコマンドで，章と節の見出しの小文字での入力を保証します．
\renewcommand{\chaptermark}[1]{\markboth{#1}{}}
\renewcommand{\sectionmark}[1]{\markright{\thesection\ #1}}
\fancyhf{}  % 現在のヘッダとフッタを無効にします．
\fancyhead[LE,RO]{\bfseries\thepage}
\fancyhead[LO]{\bfseries\rightmark}
\fancyhead[RE]{\bfseries\leftmark}
\renewcommand{\headrulewidth}{0.5pt}
\renewcommand{\footrulewidth}{0pt}
\addtolength{\headheight}{0.5pt} % 罫線のための高さを確保
\fancypagestyle{plain}{%
  \fancyhead{} % plain で定義されたページのヘッダと
  \renewcommand{\headrulewidth}{0pt} % 罫線をなしにする
}
```

図 4.1: fancyhdr パッケージの使用例

4.5　入力通りの出力を行うパッケージ

この冊子の第 2 章で `verbatim` 環境について説明しましたが，この節では verbatim パッケージについて説明します．verbatim パッケージは，LaTeX 本来の `verbatim` 環境における制限を取り除き，`verbatim` 環境を基に拡張，再定義したものです．特別目を見張るようなことではないのですが，verbatim パッケージでは新たな機能が加えられていますので，それについてここで説明しておきます．

verbatim パッケージを読み込むと，

```
\verbatiminput{filename}
```

コマンドが使用できます．このコマンドは，ちょうど `verbatim` 環境中に *filename* のファイル内容をそのまま書いたように，作成している文書中に別の文書を挿入して出力することができます．

verbatim パッケージは，一連の ‘tools’ の一部として配布されていますので，たいていのシステムにはインストール済みだと思います．このパッケージについての詳細は，参考文献 [9] を読んで下さい．

4.6　壊れやすいコマンドの保護

\caption や \section のようなコマンドの引数に与えられる文字列は，文書中に二度以上（例えば，文書の該当個所と目次に）現れることがあります．そのため，\section コマンドのような引数内ではそのまま使用できないコマンドがいくつかあります．このようなコマンドは，壊れやすいコマンドと呼ばれます．壊れやすいコマンドには，\footnote コマンド，\phantom コマンドなどがあります．これらの壊れやすいコマンドを使用するときには，コマンドの前に \protect コマンドを付けることで保護することができます．

\protect コマンドは，すぐ後ろに続くコマンドだけに影響するのであって，その引数まで保護するものではありません．多くの場合，余分に\protect コマンドを使用しても問題になりません(訳注 3)．

```
\section{I am considerate
          \protect\footnote{and protect my footnotes}}
```
(訳注 4)

(訳注 3)ですから，問題がありそうだと思ったら取りあえず \protect コマンドを書いておきましょう．でいいのかな?

(訳注 4)このままだと目次にも脚注が出力されますので，目次に脚注が必要なければオプション引数を使用して下さい．

第5章　LaTeXのカスタマイズ

これまでに説明してきたコマンドを使えば，多くの人たちにとって見やすい文書を作成できるでしょう．それらは手が込んでいるようには見えませんが，読みやすく見た感じのよい出版物の組版規則に従っています．

しかし，必要なコマンドや環境がLaTeXにない場合，あるいは既存のコマンドによる出力では満足できない場合もあるでしょう．

本章では，新しいコマンドを定義する方法や既存のコマンドを変更して異なる出力を得る方法などについて説明していきます．

5.1　新しいコマンド，環境，パッケージ

この冊子に書かれているコマンドは全て枠で囲ってあり，巻末の索引に書かれていることに気付かれた人もいるでしょう．ここでは，このようなことを行うのに必要なLaTeXコマンドを直接書く代わりに，新しいコマンドや環境を定義したパッケージファイルを作成して使用しています．この定義を用いると，次のように簡単に書くだけでコマンド名を枠で囲い，索引に出力することができます．

```
\begin{command}
\ci{dum}
\end{command}
```

```
\dum
```

この例では，コマンド名の周りを枠で囲む `command` という新しい環境と，コマンド名を出力しそのコマンド名を索引の見出しとして書き出す新しいコマンド `\ci` が使用されています．実際に，この巻末にある索引で `\dum` コマンドを探してみて下さい．そうすれば `\dum` という項目を見つけることができ，この `\dum` コマンドについて説明している全てのページ数が書かれていることがわかるでしょう．

コマンドを枠の中に入れて出力するのがイヤになれば，違った出力となるように `command` 環境の定義を変えることが簡単にできます．これは，コマンドを枠で囲うようにLaTeXコマンドを直接使用して出力を行った箇所を文書中から全て探し出し，いちいち変更するよりもはるかに簡単に行うことができるのです．

5.1.1 新しいコマンドの作成

新たにコマンドを作成するには，以下のコマンドを使用します．

`\newcommand{`*name*`}[`*num*`]{`*definition*`}`

基本的に，このコマンドは二つの引数を取ります．一つは作ろうとしているコマンド名 *name* であり，もう一つはそのコマンドの定義を書き込む *definition* です．ブラケット（角括弧，[ ]）の中に書かれる引数 *num* はオプション引数であり，この新しいコマンドが取る引数の数（9 までの数が可能）を指定します．このオプションが指定されていなければ，引数は 0 すなわち引数がないことを表します．

以下の二つの例は，コマンド定義の方法とその使い方を示しています．最初の例は，`\tnss` という名前の新しいコマンドを定義しています．これは，この冊子のオリジナル英文の表題である “The Not So Short Introduction to LaTeX 2_ε” という文字列を定義したものです．このようなコマンドは，文書中に何度も何度も繰り返し同じ文字列を書かなければならないときに使用すると便利でしょう．

```
\newcommand{\tnss}{The not
    so Short Introduction to
    \LaTeXe}
This is ``\tnss'' \ldots{}
``\tnss''
```

This is “The not so Short Introduction to LaTeX 2_ε” … “The not so Short Introduction to LaTeX 2_ε”

次の例は，引数を一つ取る新しいコマンドの定義を示したものです．`#1` の文字列の部分は，コマンドを使用する際に指定する引数に置き換えられます．もし二つ以上の引数を使用したければ，`#2` などというように書けばよいのです．

```
\newcommand{\txsit}[1]
 {This is the \emph{#1} Short
     Introduction to \LaTeXe}
% 文章本体：
\begin{itemize}
 \item \txsit{not so}
 \item \txsit{very}
\end{itemize}
```

- This is the *not so* Short Introduction to LaTeX 2_ε
- This is the *very* Short Introduction to LaTeX 2_ε

LaTeX では，すでに存在しているコマンドと同じ名前を持つ新しいコマンドを作ることはできません．そこで既存のコマンドを再定義する時のために，特別な `\renewcomannd` コマンドがあります．このコマンドは，`\newcommand` コマンドと同じ方法で使用することができます．

また，`\providecommand` コマンドを使いたくなることがあるかもしれません．このコマンドも `\newcommand` と同じなのですが，すでに同名のコマンドが存在しているときには，LaTeX 2_ε はその定義を無視して処理を続けるところが異なります．

LaTeX コマンドの直後のホワイトスペースについては，注意すべき点がいくつかあります．詳しくは，6 ページを参照して下さい．

5.1.2 新しい環境の作成

`\newcommand` コマンドと同じように，必要な環境を作り出すコマンドとして `\newenvironment` があります．`\newenvironment` コマンドは，以下のようにして使用します．

`\newenvironment{`*name*`}[`*num*`]{`*before*`}{`*after*`}`

`\newcommand` コマンドと同様に，`\newenvironment` もオプション引数が指定できます．引数 *before* には，環境内の文章を処理する前に行う内容を定義します．引数 *after* には，`\end{`*name*`}`コマンドが現れたときに行う処理内容を定義します．

以下に `\newenvironment` コマンドの使用例を示します．

```
\newenvironment{king}
 {\rule{1ex}{1ex}%
      \hspace{\stretch{1}}}
 {\hspace{\stretch{1}}%
      \rule{1ex}{1ex}}

\begin{king}
 My humble subjects \ldots
\end{king}
```

■ My humble subjects … ■

引数 *num* は，`\newcommand` コマンドと同じように使用します．LaTeX では，すでに存在している環境と同じ名前の環境は定義できません．すでに存在している環境を変更したいときには，`\renewenvironment` コマンドを使用します．定義の仕方は，`\newenvironment` コマンドと同じです．

ここで示した例で使用しているコマンドについては，後ほど説明します．`\rule` コマンドについては 73 ページ，`\stretch` コマンドについては 73 ページ，`\hspace` コマンドに関しては 73 ページを参照して下さい．

5.1.3 パッケージの作成

新たにたくさんの環境とコマンドを定義すると，文書のプリアンブルはとても長くなってしまいます．このような場合，定義したすべてのコマンドと環境を含んだ LaTeX パッケージを新たに作成するとよいでしょう．そうすれば，`\usepackage` コマンドを使用するだけで，作成中の文書でそれらの定義が使用できるようになります．

パッケージの作成は，基本的に文書のプリアンブルに書いてあった定義内容を，拡張子 `.sty` を持つファイルにコピーすればよいだけです．作成するパッ

```
% Tobias Oetiker によるパッケージの例
\ProvidesPackage{demopack}
\newcommand{\tnss}{The not so Short Introduction to \LaTeXe}
\newcommand{\txsit}[1]{The \emph{#1} Short
                        Introduction to \LaTeXe}
\newenvironment{king}{\rule{1ex}{1ex}hspace{\stretch{1}}}%
                      {\hspace{\stretch{1}}\rule{1ex}{1ex}}
```

図 5.1: パッケージの例

ケージの最初には，以下のコマンドを書いておきます．

`\ProvidesPackage{`*package name*`}`

\ProvidesPackage コマンドは，LaTeX にパッケージ名を伝え，パッケージを二度読み込もうとした際に気の利いたエラーメッセージを出力してくれます．図 5.1 は，これまでに示した例の中で定義したコマンドを含んだパッケージの例を示しています．

5.2 フォントの種類とそのサイズ

5.2.1 フォントを変更するコマンド

LaTeX は，文書の論理的な構造（節，脚注など）に基づいて適切なフォントとそのサイズを選びますが，直接それらを変更したくなることもあるでしょう．表 5.1，5.2 に，フォントやそのサイズを変更するためのコマンドを示します．それぞれのフォントの実際のサイズは，レイアウトデザインの問題であり，文書クラスとそのオプションの指定によって変わります．標準の文書クラスで定義されているサイズ変更コマンドとその実際のフォントサイズを表 5.3 に示します．

```
{\small The small and
\textbf{bold} Romans ruled}
{\Large all of great big
\textit{Italy}.}
```

The small and **bold** Romans ruled all of great big *Italy*.

LaTeX 2_ε で導入された重要な特徴のひとつに，独立したフォント属性があります．つまり，フォントサイズや種類を変更するコマンドを実行する際，それまでに指定されていたボールド，スラントといった属性を保ち続けることができるようになっています．

数式モードでは，一時的に数式モードから出て通常のテキストモードに移った後，これらのフォント変更コマンドを使用することになります．また別に，数式中でフォントを変更する特別なコマンドがあります．これを表 5.4 に示します．

フォトサイズコマンドに関しては，ブレース（中括弧）が重要な役割をします．これはグループを作るために必要で，グループ化を行うことで多くの LaTeX コマンドの有効範囲を制限することができます．

```
He likes {\LARGE large and
{\small small} letters}.
```

He likes large and small letters.

フォントサイズコマンドは行間隔も変更します．しかしこれは，フォントサイズコマンドの有効範囲内で段落が終了している場合にのみ適用されます．そのため，閉じブレース（中括弧：}）の位置には注意して下さい．次の二つの例によって，`\par` コマンドの位置と行間隔の関係を確認して下さい．

表 5.1: フォントの種類

`\textrm{...}`	roman	`\textsf{...}`	sans serif
`\texttt{...}`	`typewriter`		
`\textmd{...}`	medium	`\textbf{...}`	**bold face**
`\textup{...}`	upright	`\textit{...}`	*italic*
`\textsl{...}`	*slanted*	`\textsc{...}`	SMALL CAPS
`\emph{...}`	*emphasized*	`\textnormal{...}`	document font

表 5.2: フォントサイズ

`\tiny`	tiny font	`\Large`	larger font
`\scriptsize`	very small font		
`\footnotesize`	quite small font	`\LARGE`	very large font
`\small`	small font	`\huge`	huge
`\normalsize`	normal font		
`\large`	large font	`\Huge`	largest

表 5.3: 標準文書クラスにおけるフォントサイズ

サイズ	10pt (デフォルト)	11pt オプション	12pt オプション
`\tiny`	5pt	6pt	6pt
`\scriptsize`	7pt	8pt	8pt
`\footnotesize`	8pt	9pt	10pt
`\small`	9pt	10pt	11pt
`\normalsize`	10pt	11pt	12pt
`\large`	12pt	12pt	14pt
`\Large`	14pt	14pt	17pt
`\LARGE`	17pt	17pt	20pt
`\huge`	20pt	20pt	25pt
`\Huge`	25pt	25pt	25pt

表 5.4: 数式フォント

コマンド	入力例	出力
`\mathcal{...}`	`$\mathcal{B}=c$`	$\mathcal{B}=c$
`\mathrm{...}`	`$\mathrm{K}_2$`	K_2
`\mathbf{...}`	`$\sum x=\mathbf{v}$`	$\sum x=\mathbf{v}$
`\mathsf{...}`	`$\mathsf{G\times R}$`	$\mathsf{G\times R}$
`\mathtt{...}`	`$\mathtt{L}(b,c)$`	$\mathtt{L}(b,c)$
`\mathnormal{...}`	`$\mathnormal{R_{19}}\neq R_{19}$`	$\mathnormal{R_{19}}\neq R_{19}$
`\mathit{...}`	`$\mathit{ffi}\neq ffi$`	$\mathit{ffi}\neq ffi$

```
{\Large Don't read this! It is not
true. You can believe me!\par}
```

> Don't read this! It is not true.
> You can believe me!

```
{\Large This is not true either.
But remember I am a liar.}\par
```

> This is not true either. But remember I am a liar.

文章中で一つ以上の段落全部のフォントサイズを変更したいときには，フォント変更のコマンドを環境として使用するとよいでしょう．

```
\begin{Large}
 This is not true.
 But then again, what is these
 days \ldots
\end{Large}
```

> This is not true. But then again, what is these days . . .

このように環境として使用すると，ブレース（中括弧）の数を数えて対応関係に注意を払う必要がなくなります．

5.2.2　フォントに対する注意の喚起

この章の最初でも述べましたが，以上のように直接フォントの種類とサイズを変更するようなコマンドを文書中でむやみに使用することは危険です．これは，文書の論理的構造と視覚的構造とを分離するという LaTeX の基本的な考え方に反します．つまり，特別な情報を読者に伝えるためにいくつかの場所で同じフォント変更コマンドを使う場合には，フォント変更コマンドを直接使用する代りに，`\newcommand` を使って“論理的に扱えるコマンド”を定義するべきなのです．

```
\newcommand{\oops}[1]{\textbf{#1}}
Do not \oops{enter} this room,
it's occupied by a \oops{machine}
of unknown origin and purpose.
```

> Do not **enter** this room, it's occupied by a **machine** of unknown origin and purpose.

この手法には，注意を引くために `\textbf` 以外の別の視覚的表示を使うかどうかを，文書作成中にその場その場で指定したり，`\textbf` を使用しているすべての場所を探したりする必要がなく，さらには危険を知らせたり別の理由のためにどんな表示をすればよいかをその段階で理解していなくても，少し後の段階で決定できるという利点があります．

5.2.3 フォントに関する助言

フォント種類とフォントサイズに関する記述をまとめるために，ここでちょっとした助言をしておきましょう．

> **Remember!** *The* MORE fonts you use **in** a `document`, *the* more READABLE and ***beautiful*** *it* becomes.

5.3 スペース

5.3.1 行間隔

文書中で行間を広げたい場合，文書のプリアンブルで以下のコマンドに値を設定することで行間を変更することができます．

```
\linespread{factor}
```

“ワンハーフ” スペースにしたい時には `\linespread{1.3}` を，“ダブル” スペースで文書を書く際には `\linespread{1.6}` を指定します．通常，行間は広がっておらず既定値は 1 となっています．

5.3.2 段落のレイアウト

LaTeX には，段落のレイアウトを変更する二つのパラメータが存在します．作成している文書のプリアンブルで以下のように定義すると，段落のレイアウトを変更することができます．

```
\setlength{\parindent}{0pt}
\setlength{\parskip}{1ex plus 0.5ex minus 0.2ex}
```

この二つのコマンドは，段落最初のインデント量を零にして，二つの段落間のスペースを増やします．大陸ヨーロッパでは，このように段落間を空け，インデントしないレイアウト形式がよく使われます．ところが，このレイアウトは目次にも影響を及ぼすことに注意して下さい．上述のように指定すると，目次に現れる各行はこれまで以上にもっと行間のスペースが空いてしまいます．これを避けるためには，上述のコマンドを文書のプリアンブルから `\tablefocontents` の後に移動させるか，使用を見送るかすることになるでしょう．それというのも，本格的なほとんどの書籍では，段落最初の行はインデントされており，段落間に余分なスペースが取られることはないからです．

インデントしないようにした段落でインデントを行いたいときには，次の

コマンドを段落の最初で使用します[1]．

`\indent`

明らかにこれは，`\parindent` が零以外の値であるときにしか意味をなしません．

インデントされない段落を作成するには，段落の最初で次のコマンドを使用します．

`\noindent`

これは，節などのコマンドを用いずに，直接本文を書き始める際に使用できます．

5.3.3 水平方向のスペース

LaTeX は，単語間や文と文との間のスペースの大きさを自動的に決定します．水平方向のスペースを増やすには，次のコマンドを使用します．

`\hspace{length}`

たとえ行頭や行末に現れてもこのスペースを有効にしたい場合には，`\hspace` コマンドの代わりに `\hspace*`コマンドを使用します(訳注 1)．*length* には，単位の付いた長さを指定します．主な長さの単位を表 5.5 に示しておきます．

```
This\hspace{1.5cm}is a space
of 1.5 cm.
```

This　　　　　is a space of 1.5 cm.

次のコマンドは，特別に伸び縮みするスペースを生成します．

`\stretch{n}`

このコマンドを使用すると，その行で取れるだけのスペースを生成します．二つの`\hspace{\stretch{n}}`コマンドが同じ行に現れた場合，引数 n に指定された数値に応じたスペースが取られます．

```
x\hspace{\stretch{1}}
x\hspace{\stretch{3}}x
```

x　　　　x　　　　　　　　　　　　x

---

[1] 節の見出し直後の段落でインデントを行うには，'tools' に同梱の indentfirst パッケージを使用して下さい．

(訳注 1)つまり，`\hspace` コマンドは行頭や行末では無視されることがわかります．

表 5.5: TeX での単位

`mm`	ミリメートル ≈ 1/25 インチ	⊔
`cm`	センチメートル = 10 ミリメートル	└──┘
`in`	インチ = 25.4 ミリメートル	└──────────┘
`pt`	ポイント ≈ 1/72 インチ ≈ $\frac{1}{3}$ mm	‖
`em`	ほぼ現在使用中のフォントの文字 ‘M’ の幅	└─┘
`ex`	ほぼ現在使用中のフォントの文字 ‘x’ の高さ	⊔

5.3.4 垂直方向のスペース

段落，節，小節などの間のスペースは，LaTeX が自動的に決定します．必要があれば，段落間の垂直方向のスペースを次のコマンドで余分に空けることができます．

\vspace{*length*}

このコマンドは，通常二つの空行の間で使用します．もしページの最初もしくは最後でもこのスペースを取りたい場合には，`\vspace` コマンドの代わりにアスタリスク付きの `\vspace*`コマンドを使用して下さい．

ページの一番下の端に文章を出力したり，ページの中央に出力したりするために `\pagebreak` コマンドと一緒に `\stretch` コマンドを使用します．

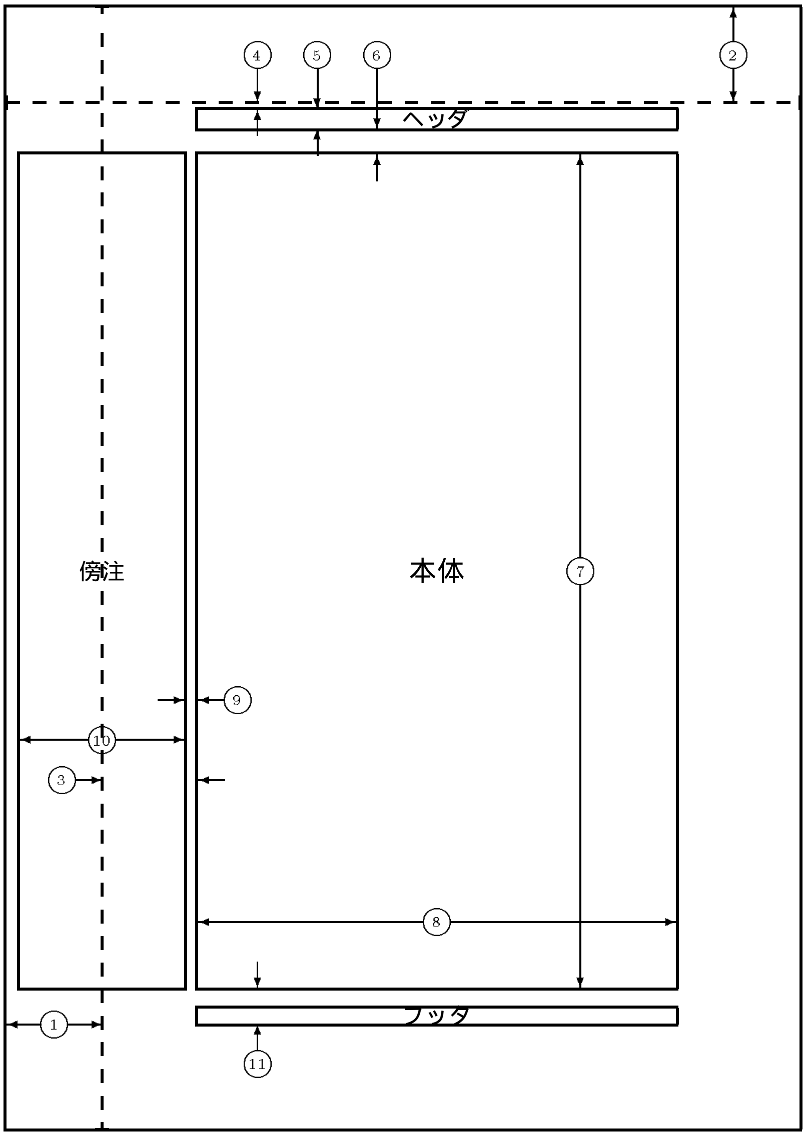
```
文章が\ldots

\vspace{\stretch{1}}
この文章は，ページの一番下に出力されます．\pagebreak
```

一つの段落中の行間や `table` 環境の各行の間隔を余分に空けたい場合には，次のコマンドを使用します．

\\\\[*length*]

5.4 ページレイアウト

LaTeX 2_εでは，`\documentclass` コマンドのオプションで出力する用紙サイズを設定でき，適切なマージンなどが自動で設定されますが，その値に満足できないこともあります．そんな場合，マージンなどの値を変更することが



1	1 インチ + `\hoffset`	2	1 インチ + `\voffset`
3	`\evensidemargin = 72pt`	4	`\topmargin = 5pt`
5	`\headheight = 15pt`	6	`\headsep = 19pt`
7	`\textheight = 628pt`	8	`\textwidth = 360pt`
9	`\marginparsep = 10pt`	10	`\marginparwidth = 124pt`
11	`\footskip = 27pt`		`\marginparpush = 5pt` （非表示）
	`\hoffset = 0pt`		`\voffset = 0pt`
	`\paperwidth = 597pt`		`\paperheight = 845pt`

図 5.2: ページレイアウトパラメータ

できます．図5.2に，変更することのできるパラメータを示します．この図は，‘tools’[2] 中の layout パッケージを用いて作られたものです．

ちょっと待った！...「ページの幅を少し広くしよう」と思い込む前に，もう少し考えてみましょう．LaTeX ではほとんどのことがそうであるように，ページレイアウトにもそのように定義されている理由があるのです．

確かに，市販の MS Word のページレイアウトと比較すると一行の幅はかなり狭くなっています．しかし，お気に入りの本[3]を手にとって，一行の文字数を数えてみて下さい．一行に，66文字以上はないでしょう．そこで，LaTeX の出力結果でも同じように文字数を数えて下さい．一行に66文字くらいであることがわかります．経験的に，一行にたくさんの文字を詰め込むととたんに読みづらくなることがわかっています．これは，ある行の終わりから次の行の最初まで目を移動するのが難しいためです．また，新聞が複数の列(訳注2)から構成されているのも同じ理由です．

そこで，もし作成している文書の幅を広げるならば，その文書を読む人に無理をさせることを覚えておいて下さい．ただ，十分に注意を払うのならばその方法を教えましょう...

LaTeX には，これらパラメータの長さを変更するために二つのコマンドが存在し，通常文書のプリアンブルで使用します．

一つ目は，パラメータに決まった値を直接設定するコマンドです．

`\setlength{`*parameter*`}{`*length*`}`

二つ目は，パラメータに設定し長さを増減させるコマンドです．

`\addtolength{`*parameter*`}{`*length*`}`

このコマンドは，現在の値を基準に設定することができるので，`\setlength` コマンドよりも使いやすいでしょう．一行の幅を 1cm 広くするには，以下のコマンドを文書のプリアンブルに書いて下さい．

```
\addtolength{\hoffset}{-0.5cm}
\addtolength{\textwidth}{1cm}
```

この場合，calc パッケージを使用すると，`\setlength` コマンドなどの引数中，もしくは数値を書き込むことのできる場所に加減乗除などの算術式が使用できるようになります．

---

[2] CTAN:/tex-archive/macros/latex/packages/tools

[3] 評判の良い出版社が発行している書籍のことです．

(訳注2) 日本の一般の縦組された新聞では，段ですね．

5.5　もっともっと長さについて

可能である限り，LaTeX で文書を作成する際には直接長さを指定することは避け，ページレイアウトの幅や高さを基準にして長さを設定します．図の幅に対しては，ページを埋めるために `\textwidth` を基にします．

以下の三つのコマンドは，引数 *text* にある文字列の高さ，深さ，幅を *command* に書かれたコマンドに設定します．

```
\settoheight{command}{text}
\settodepth{command}{text}
\settowidth{command}{text}
```

次に，これらのコマンドの使用例を示します．

```
\flushleft
\newenvironment{vardesc}[1]{%
  \settowidth{\parindent}{#1:\ }
  \makebox[0pt][r]{#1:\ }}{}

\begin{displaymath}
 a^2+b^2=c^2
\end{displaymath}

\begin{vardesc}{Where}$a$,
 $b$ -- are adjunct to the right
 angle of a right-angled triangle.

 $c$ -- is the hypotenuse of
 the triangle and feels lonely.

 $d$ -- finally does not show up
 here at all. Isn't that puzzling?
\end{vardesc}
```

$$a^2 + b^2 = c^2$$

Where: a, b – are adjunct to the right angle of a right-angled triangle.

c – is the hypotenuse of the triangle and feels lonely.

d – finally does not show up here at all. Isn't that puzzling?

5.6　ボックス

LaTeX は，ボックスを配置しまくることでページを作成します．それぞれの文字は小さなボックスであり，単語を形成するために別の文字とつなぎ合わされます．これらの単語は，また別の単語とつなぎ合わされます．この際，単語間には伸びたり縮んだりしてページを同じ長さの行で正確に埋め尽くすことができるゴムのような特別な糊が使用されます．

これが TeX で，実際に行われていることの最も簡単な説明です．要するに，TeX は糊とボックスを操っているということになります．文字だけがボックスを形成しているわけではありません．別のボックスを含んだボックスの中に仮想的にあらゆる要素を置くことができ，LaTeX はそれぞれのボックスをあたかも一つの文字として扱います．

これまでの章で既にいくつかのボックスを扱っているのですが，今まで説明してきませんでした．例えば，`tabular`環境や`\includegraphics`コマンドは共に一つのボックスを作ります．このことは，二つの表や図を横に並べることが簡単にできることを意味します．ただし，並べられた際の幅が`\textwidth`を越えないことを確認することは必要でしょう．

また次の二つの方法で，任意の段落を一つのボックスに収めることもできます．一つは

`\parbox[`*pos*`]{`*width*`}{`*text*`}`

コマンドで，もう一つは

`\begin{minipage}[`*pos*`]{`*width*`}` text `\end{minipage}`

環境です．オプション引数の *pos* パラメータは，前後の文章のベースラインに対するボックスの上下位置を設定するためのもので，`c`，`t`，`b`のいずれかを指定します．*width* はボックスの幅を決める引数で，長さを指定します．`minipage`環境と`\parbox`コマンドの大きな違いは，`\parbox`コマンドの中ではあらゆるコマンドと環境が使用できないのに対して，`minipage`環境内ではそのほとんどが使用できることにあります．

改行を含んだ段落全体を詰め込む`\parbox`とは別に，水平方向にだけ文章などを並べる一連のボックスコマンドもあります．一つは，既に知っているコマンドです．`\mbox`コマンドがそれに当たります．このコマンドは，ひと続きのボックスを別のボックスに詰め込んだり，LaTeXが二つの単語に分断しないようにするために使用されます．ボックスを別のボックスの中に置くことができるように，これらの水平ボックスも非常に柔軟に扱うことができます．

`\makebox[`*width*`][`*pos*`]{`*text*`}`

引数 *width* は出力されるときのボックスの幅を指定します[4]．この引数には，長さを表すパラメータのほかに，`\width`，`\height`，`\depth`，`\totalheight`といったコマンドも使用できます．これらのコマンドには，*text* で指定された文字列を組版したときに得られるボックスの数値がセットされます．*pos* パラメータは，次に示す一文字を取ることができます．すなわち，`c`（中央：**c**enter），`l`（左詰め：**l**eft），`r`（右詰め：**r**right），ボックス幅一杯に文字列を広げる`s`の4つです．

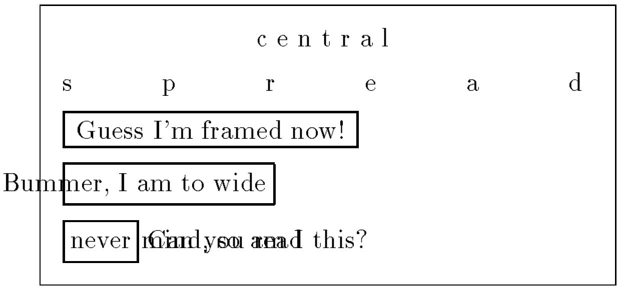
`\framebox`コマンドは，文字列を枠で囲む以外は`\makebox`コマンドと全く同じように使用することができます．

以下に，`\makebox`，`\framebox`コマンドを使用した例を示します．

---

[4] これは，ボックスの中に書かれる文章などの幅よりも短くすることもできます．すなわち幅を0ptと指定することもでき，周りのボックスに影響することなくボックスの中の文字列などを組むことができます．

```
\makebox[\textwidth]{%
   c e n t r a l}\par
\makebox[\textwidth][s]{%
   s p r e a d}\par
\framebox[1.1\width]{Guess I'm
   framed now!} \par
\framebox[0.8\width][r]{Bummer,
   I am to wide} \par
\framebox[1cm][l]{never
   mind, so am I}
Can you read this?
```



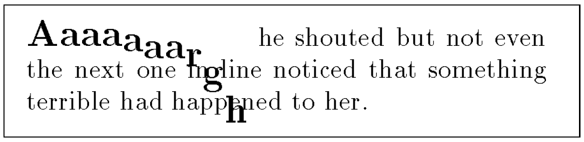
```
central
s p r e a d
Guess I'm framed now!
Bummer, I am to wide
never mind, so am I Can you read this?
```

水平方向の制御を理解したところで，次は垂直方向の説明に移りましょう[5]．LaTeX では，何でもないことです．

`\raisebox{`*lift*`}[`*depth*`][`*height*`]{`*text*`}`

上記のコマンドで，ボックスを上下方向へ移動させることができます．最初の三つのパラメータ内には，*text* 引数で与えられた文字列を組版したときのボックスの大きさに基づいて設定される `\width`，`\height`，`\depth`，`\totalheight` コマンドが使用できます．

```
\raisebox{0pt}[0pt][0pt]{\Large%
 \textbf{Aaaa\raisebox{-0.3ex}{a}%
 \raisebox{-0.7ex}{aa}%
 \raisebox{-1.2ex}{r}%
 \raisebox{-2.2ex}{g}%
 \raisebox{-4.5ex}{h}}}
he shouted but not even the next
one in line noticed that something
terrible had happened to her.
```



Aaaaaaargh he shouted but not even the next one in line noticed that something terrible had happened to her.

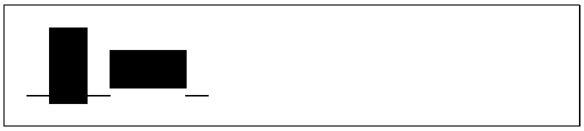
5.7 罫線と支柱

数ページ前に，次のコマンドがあったことに気付いた人もいるでしょう．

`\rule[`*lift*`]{`*width*`}{`*height*`}`

通常このコマンドは，単純な黒塗りのボックスを出力します．

```
\rule{3mm}{.1pt}%
\rule[-1mm]{5mm}{1cm}%
\rule{3mm}{.1pt}%
\rule[1mm]{1cm}{5mm}%
\rule{3mm}{.1pt}
```



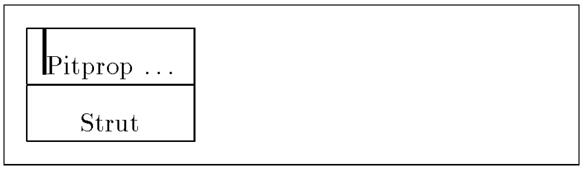
---

[5] 全制御は，水平，垂直の両方向の制御によってなされるだけです...

このコマンドは，垂直方向，水平方向の罫線を描くのに使用できます．例えば，この冊子の表紙に描かれている罫線はこの `\rule` コマンドを使用して描いています．

特別な場合として，高さのみがあり幅が零の罫線も描くことができます．組版では，支柱と呼ばれます．これは，ページ中の要素の高さがある値より小さくならないようにするために使われます．例えば，`tabular` 環境中で行の高さをある高さ以上にするために使用します．

```
\begin{tabular}{|c|}
 \hline
 \rule{1pt}{4ex}Pitprop \ldots\\
 \hline
 \rule{0pt}{4ex}Strut\\
 \hline
\end{tabular}
```



Pitprop …
Strut

参考文献

[1] Leslie Lamport. *LaTeX: A Document Preparation System.* Addison-Wesley, Reading, Massachusetts, second edition, 1994, ISBN 0-201-52983-1．［邦訳：阿瀬はる美訳『文書処理システム LaTeX 2ε』ピアソン・エデュケーション，1999 年，ISBN 4-89471-139-7．］

[2] Donald E. Knuth. *The TeXbook,* Volume A of *Computers and Typesetting,* Addison-Wesley, Reading, Massachusetts, second edition, 1984, ISBN 0-201-13448-9．［邦訳：斎藤信男監修，鷺谷好輝訳『改訂新版 TeX ブック』アスキー出版局，1992 年，ISBN 4-7561-0120-8．］

[3] Michel Goossens, Frank Mittelbach and Alexander Samarin. *The LaTeX Companion.* Addison-Wesley, Reading, Massachusetts, 1994, ISBN 0-201-54199-8．［邦訳：アスキー書籍編集部監訳『The LaTeX コンパニオン』アスキー出版局，1998 年，ISBN 4-7561-1813-5．］

[4] それぞれの LaTeX システムには，そのシステムに固有な事項について説明されている，いわゆる *LaTeX* ローカルガイドが存在しています．ファイル名は通常，`local.tex` となっています．運悪く，そのような文書を用意していない怠け者のシステム管理者だった場合，わからないことは，近くの LaTeX 伝道師に会いに行ってお尋ね下さい．

[5] LaTeX3 Project Team. *LaTeX 2ε for authors.* `usrguide.tex` というファイル名で，LaTeX 2εの基本ファイルと一緒に配布されています．

[6] LaTeX3 Project Team. *LaTeX 2ε for Class and Package writers.* `clsguide.tex` というファイル名で，LaTeX 2εの基本ファイルと一緒に配布されています．

[7] LaTeX3 Project Team. *LaTeX 2ε Font selection.* `fntguide.tex` というファイル名で，LaTeX 2εの基本ファイルと一緒に配布されています．

[8] D. P. Carlisle. *Packages in the ‘graphics’ bundle.* `grfguide.tex` というファイル名で，一連の ‘graphics’ パッケージと一緒に配布され，LaTeX の基本ファイルの入手先から入手できます．

[9] Rainer Schöpf, Bernd Raichle, Chris Rowley. *A New Implementation of LaTeX’s verbatim Environments.* `verbatim.dtx` というファイル名で，

一連の ‘tools’ パッケージと一緒に配布され，LaTeX の基本ファイルの入手先から入手できます．

[10] Graham Williams. *The TeX Catalogue.* このカタログは，TeX や LaTeX に関連した多数のパッケージファイルのほぼ完全な一覧となっています．`CTAN:/tex-archive/help/Catalogue/catalogue.html` から入手できます．

[11] Keith Reckdahl. *Using EPS Graphics in LaTeX 2ε Documents.* このファイルは，EPS ファイルに関するあらゆる事や今まで知りたかった事よりも多くの事柄，さらに LaTeX 文書内での EPS ファイルの扱い方について説明しています．`CTAN:/tex-archive/info/epslatex.ps` から入手できます．

索引