LaTeX 入門

水谷正大 mizutani@ic.daito.ac.jp

2013 年度 β 版

目次

概要

電子組版システムである TEX を DEC のコンピュータ科学者である L. Lamport が使いやすいようにマクロパッケージ化して発展してきた LATEX システムを紹介します。

急速に発展し、しかもその重要度をますます高めている Web テクノロジーは、印刷物として出版される前に情報をすみやかに普及させています。しかし、Web が克服しなければならない問題の 1 つに表示の品質があり、まだ決定的な打開策は提案されていません。長い年月を経て洗練されてきた印刷物の品質やレイアウト、数式、化学式や楽譜などの情報をどのように Web で取り扱うべきかの多くの議論がなされています。

LATEX システムが優れている理由として

- 印刷文書作成支援システムとして完成度が高く、目次、相互参照や索引など高度な編集作業が自動化できる
- 文書ファイルが編集・文書処理が可能なテキストファイルでありながら，出力結果とファイルが 1 対 1 に対応している
- 多くの文書様式のためのスタイルファイルが用意されており，同じ文書内容であってもスタイルファイルを変更するだけで文書レイアウトを変えることができる

などがあげられます。

1 TEX はどのように発音するのか

D.Knuth は TEX のバイブルである自身の著作 [1] で TEX の発音について述べています。

> “TEX という名称は....$\tau\epsilon\chi$ の大文字で綴る。... TEX の χ を x と発音せず、ギリシャ語の chi のように発音する。そのため、TEX の χ は、blecchhh という言葉の語尾と同じ響きになる。スコットランド語の loch とかドイツ語の ach のように ch と発音したり、またはスペイン語の j やロシア語の kh というような発音をする。コンピュータに向かって、息を吐きかけるように正しく発音すれば、端末は少しばかり曇るかもしれない。”

では、日本語ではどう発音するのでしょうか。“テック” とか “テフ” と発音している人もいます。したがって、LATEX は “ラテック” とか “ラテフ” となります。アメリカでは “レイテック” に聞こえるように発音しているようです。いずれにせよ正体は*なぞ*につつまれたままです。

2 組版ソフトウェア TEX ファミリー

TEX はスタンフォード大学の Donald E.Knuth によって開発された組版システムで、コンピュータによる文書の作成から組版・出版までを想定した文書作成ソフトウエア環境で、Knuth が TEX ソースプログラムを公開してきたために多くの OS に移植され現在にいたっています。TEX の現在における Version は 3.1415926 で、Knuth 本人によって「打ち止め」宣言されています[*1]。TEX ファミリーには、ここで詳しく取り上げる LATEX（アスキーが日本語化した pTEX や pLATEX）以外にもアメリカ数学会の $\mathcal{AMS}$–TEX や自在に化学式が使える XƎTEX や楽譜記号が記述できる MusiXTeX などいくつかの仲間があり、LATEX システムからそれぞれ専用のマクロパッケージを読み込むことによって簡単に利用することができます。

TEX ファミリーは世界の標準的組版ソフトウェア (typesetting software) の一つとしての地位を確立して

[*1] TEX システムのコアの開発は Knuth によって打ち止めされましたが、これに基づいた LATEX などの TEX ファミリーや各種のマクロパッケージの開発や各種う OS へのポーティングは現在でも盛んに行われています。

おり、Z 書籍や雑誌などの出版事業に積極的に使われています。

2.1 日本語 LaTeXS システムの入手

TeX の入手やインストールなど詳しい最新情報に関しては奥村晴彦氏が運営している TeXWiki `http://oku.edu.mie-u.ac.jp/~okumura/texwiki/` が最善です。是非参照してください。TeX 文書の文法や詳しい作成方法については、同じ奥村氏による著書 [4] が日本での標準的テキストとして広く読まれています。付録のディスクには各 OS ごとの代表的な TeX ディストリビューションが付属しているので、ネットワークからのダウンロードが困難なユーザーには重宝します。

ここで紹介する代表的な OS ごとの TeX システムの入手とインストール方法はすべて TeXWiki にありますから随時参照してください。

Linux　最近の Linux ディストリビューションでは TeX システム一式が標準でインストールされます。

Mac　TeXWiki/Mac に各種の方法が説明されています。バイナリをインストールする方法ならどのやり方でもインストールは非常に簡単です。筆者は Mac のパッケージ管理ソフトウエアでる MacPort を導入して、ソースからコンパイルしてインストールしています。

Mac に TeX をインストールしたら、TeX 文書作成統合環境である「TeXShop」も併せてインストールしましょう。また、PowerPoint や Keynote などスライド専用ソフトエアに TeX システムで生成した数式などを画像として簡単に貼りつけられる「LaTeXiT」または「TeX2img」もお忘れなく。

Windows　Mac に比べて少々面倒ですが、説明通りに従えば難しくはありません。TeXWiki/MS Windows(インストーラ) の 'MS Windows' からリンクされている TeXWiki インストール (Windows) に各種の方法が説明されています ()。さらに簡便な方法として 'インストーラ' からリンクされている TeXWiki TeX installers for Windows の「TeX インストーラ 3」 `http://www.ms.u-tokyo.ac.jp/~abenori/mycreate/abtexinst.html` が便利でしょう。

津田でもインストールされている TeX 文書作成統合環境である「WinShell」もお忘れなく。TeX システムで生成した数式などを画像として PowerPoint スライドなどに貼りつけるための「TeX2img」や「IguanaTex」もお忘れなく。

TeX で生成した数式を画像として貼り付けるために各 OS で動作する単体アプリを使わず、Web アプリケーションを利用することもできます（TeXWiki/プレゼンテーションツール)。

TeXclip　`http://maru.bonyari.jp/texclip/`

TeXCrop　`http://www.fukudat.com/texcrop/`

3 LaTeX システムの使い方

現在の多くの TeX ユーザは、エディタ機能を備えた上でエラー処理、プレビューおよび修正作業の多くを自動化する TeX 統合環境を利用して、TeX ファイルから最終的に PDF ファイルの作成を目的としています。PDF ファイルなら、事実上の必携ソフトウエアである Adobe Reader を使ってどんなユーザでもプレビュー可能で、しかもスマートフォンやタッチ式デバイスで電子書籍として読むことも可能です。

3.1 LaTeX システムの作業の流れ

LaTeX システムの詳細に立ち入る前に、LaTeX ファイルを作成し LaTeX システムを使って印刷するまでの手続きの流れを示します。

以下の手続きの流れはどんなコンピュータを使う場合でも同じです。TeX 統合環境を使えば，これらの大部分が隠蔽されて自動処理されるので、以下の TeX 作業は実際には難しくはありません。

1. 適当なテキストエディタを使って[*2] 、拡張子 `.tex` の付いたテキストファイル（LaTeX 文書ファイルとか TeX ファイルと呼びます）を LaTeX 規則にしたがって作成し、これを保存する。
2. TeX ファイルを コマンド `platex` によってコンパイルして、DVI ファイルの作成を試みる。
3. LaTeX システムからエラーが報告されたときには、処理を中断してエディタに戻り TeX ファイルを修正・保存してから再びコンパイルする。作業 2. と作業 3. をエラーがなくなるまで繰り返す。
4. 印刷する前にプレビューア (previwer) で DVI ファイルの印刷イメージを確かめる。TeX 統合環境を使うユーザは、コンパイルの後に生成された DVI から自動的に PDF ファイルが生成され、この作業をスキップできます。
5. `dvipdfmx` などのコマンドで DVI ファイルから PDF ファイルを生成してプレビューする[*3]。
6. 生成した DVI/PDF ァイルを配布あるいは印刷する。

LaTeX システムにおけるこの処理の流れを図 1 に示します。

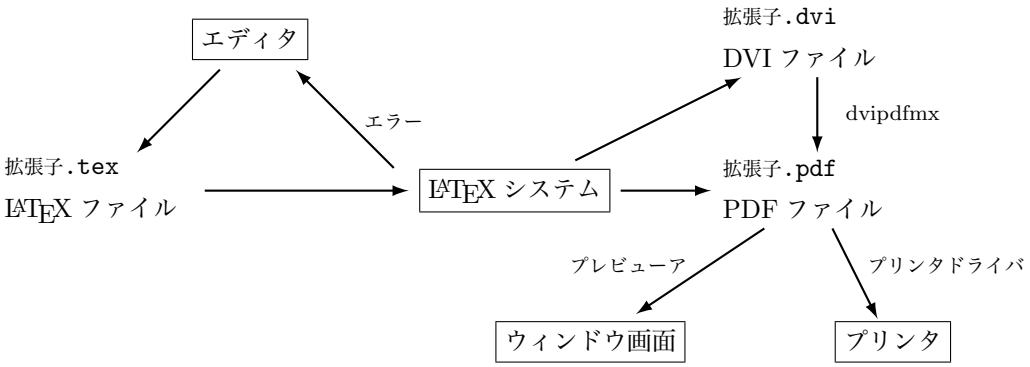

図 1　LaTeX システムにおける処理の流れ

TeX システムの利用者はワープロソフトウェアなどを使って文書を印刷する作業に比べて、図 1 のように‘文法的’に正しい LaTeX ファイルを作成して DVI ファイルを生成するという余計な手間を経ねばなりません。それであっても、TeX ユーザはさまざまなレイアウトで非常に美しい文書を作成できるという代償を得

[*2] Windows/Mac で利用できる TeXworks、Mac 専用の TeXShop や Windows 用に WinShell などの TeX 統合環境ソフトウエアはエディタ機能を内蔵していますが、手馴れたテキストエディタがあればそれも併用して使うことが可能です。

[*3] DVI ファイルを経由せずにコマンド `pdflatex` などによって直接 pdf ファイルを生成することも可能ですが、日本語野処理に問題がある場合がある。

るのです。

3.1.1 DVI ファイル

DVI ファイルとは装置に独立（DeVice Independent）な印刷イメージファイルで、使用しているコンピュータや印刷しようとするプリンタの種類とは無関係なバイナリファイルです。DVI ファイルには印刷に必要なフォントや各ページ上の座標位置などの情報が納められています。ただし、5 節で後述するように、DVI ファイルにはポストスクリプト画像ファイルの情報は含まれていません。画像情報を含むすべての情報を 1 つのファイルに収めるには、5 節で説明するように、`dvipsk` を使ってポストスクリプトファイルに変換するか、あるいは 5.4 節で触れるように `dvipdfmx` を使って PDF ファイルに変換する必要があります。

DVI ファイルが使用する装置に無関係であることは次の二つの意味があります。

- 使用したコンピュータに独立
  Windows や Macintosh を使って TEX システムから DVI ファイルさえ作成すれば、そのファイルを Linux に持ってきても（あるいは、その逆でも）、PDF ファイルを生成したり、プレビューしてプリンタから出力できます。
- 印刷しようとするプリンタに独立
  あらかじめどのようなプリンタで印刷するかによらないで作業をすることができます。手元にある個人用のプリンタの出力結果と高価な印刷機からの出力結果の差は、その出力品質だけです。

つまり、どんなコンピュータであろうとも DVI ファイル（さらに、それからポストスクリプトあるいは PDF ファイル）を作成してしまえば、そのファイルを高精度な印刷機を持つ印刷所に持ち込んで出力を頼めば、自分のパソコンで確認したレイアウトどおりの最高水準の印刷物が得られることになります。

3.1.2 PDF ファイル

PDF ファイルは Adobe 社によって規定されたページ出力のためのファイル形式です。PDF ファイルであれば無料配布されている Adobe Reader などを使ってどんなコンピュータでもその内容を表示できます。LATEX システムでは、コマンド `dvi2pdfmx` を使って、次のようにして DVI ファイル *sample.dvi* から PDF ファイルを生成します。

```
% dvipdfmx sample.dvi
```

Adobe Reader が多くの PC での必携アプリケーションである今日の事情を考えると TEX システムによって DVI ファイルではなく、PDF ファイルを最終生成ファイルとするのが今日の TeX 利用の姿です。

3.2 印刷出力までの手順

ここでは簡単な LATEX 文書をエディタで作成し、これをプリンタから出力するまでの具体的手続きを紹介します。

節 3.2.3 以降の各段階の処理をコマンドによって行う様子を非常に面倒だと感じるかもしれません。多くの TeX ユーザがそうであるように、統合環境を使えばプレビューまでの処理は自動化され、事実上 one click で済んでしまします。ただし、高度な TEX 機能を利用する場合には、ここで説明するようなコマンドによる処理が必要になる場合もあります。

Windows では [アクセサリ]-[コマンドプロンプト] からコマンドプロンプトウィンドウ（俗称 DOS 窓）を、Mac では [アプリケーション内のユーティリティにあるターミナル app を実行して、コンピュータに実行させる命令（コマンド）を文字列として入力することができます。図 1 の処理の流れさえ把握していれば、OS の違いによる混乱は起こらないでしょう。

3.2.1 LaTeX 文書の作成の実際

テキストエディタを使って作成する LaTeX ファイルは常に `.tex` という拡張子を付ける必要があります。簡単な LaTeX ファイルの例として、次のように入力したファイルを *sample.tex* として保存してみましょう (実はワザと誤りを入れています)。

```
\documentclass{jsarticle}% この jsarticle が現在の事実上の標準である
\begin{document}          % 全角文字『}』を使った誤り
初めての\TeX{}文書です。
どんなふうに
仕上がるかな?

とにかくエディタで文書さえ書いておけば、後で\LaTeX{}ファイルにすることは簡単。
やっぱり文章は見てくれより{\Large 中身}
が大切だからね。

でも、こんなこと
\[
\int_0^\infty \frac{\sin ax}{x} dx=
        \frac{\pi}{2}\quad (a>0)
\]
が書けるとなると、\textbf{外見も大切}かなあと思ってしまうよね。
\end{document}
```

ここで表示されている半角のバックスラッシュ記号 '\' について一言。Windows や Macintosh などの日本語環境では通常この半角バックスラッシュ記号は Yen キーで入力され、¥ というように表示されるものです。以下では、自分の環境に合わせて '\' を '¥' と読み替えて下さい。

この簡単な例は、**LaTeX 文書ファイルの必須要素**を示しています。

- まず `\documentclass{jarticle}` を書いて文書スタイルを指定する。`jsarticle` は標準的な日本語の論文スタイルです[*4] (出版社や学会から様々な文書スタイルが提供されています)。
- 文書スタイルの指定の下に `\begin{document}` を書く。`\documentclass{....}`と`\begin{document}`

[*4] この `jsarticle` は奥村晴彦氏が改良した日本語用論文スタイルで、以前の `jarticle` に替わって事実上の標準となっています。TeXWiki からダウンロードできます

の間を行間部分をプリアンブルといい、通常はさまざまな情報を書きます（11 ページの 4.1 節以降で紹介していきます）。

- 文書本体は `\begin{document}` の後から書き始める。
- 本文の最後に `\end{document}` を書く。

なお、ファイル中の記号 '%' は、それ以降から行末までをコメントとして扱うために使われています（3.3.2 節）。コメントは仕上がり文書には反映されませんが、メモや修正事項などを記入したり LaTeX 文書を分割して個別の文書ファイルのデバッグをするなどさまざまな用途があり、たいへん便利です（3.3.2 節）。

3.2.2 DVI ファイルの作成

保存した LaTeX ファイル *sample.tex* を LaTeX システムによってコンパイルして DVI ファイル *sample.dvi* を作成するには次のようにします。

```
% platex sample.tex
```

このとき LaTeX 構文上のエラーがあるとき、LaTeX システムは途中で処理を中断し、違反あるい処理が破綻したファイルの行番号をエラーメッセージとともに示し、"?" のプロンプトによって利用者にこれ以降の処理を尋ねてきます。実際には、その箇所でエラーが起こったというよりも、それ以前にあった誤入力によって矛盾が積み重なってエラーとして顕在化する場合が多く、エラーの修正にはその行以前にさかのぼって探査する必要があります。

"?" のプロンプトは、このエラーをユーザがどのように対処するかを LaTeX システムに指示するために、キーボードからのコマンド "入力待ち状態" を表しています。上の sample.tex の場合では次のようなエラー状態となります（以下の例は、MacOS で MacTeX を利用した場合です）。この例では、`\begin{document}` のように、半角の '}' でなければならないのに、日本語モードで入力した全角の '｝' が使われてしまったためにエラーが発生しました。

```
... 利用している OS や使っている TeX システムに応じたメッセージが続く
This is e-pTeX, Version 3.1415926-p3.3-110825-2.4 (utf8.euc) (TeX Live 2012)
 restricted \write18 enabled.
(./test.tex
pLaTeX2e <2006/11/10> (based on LaTeX2e <2011/06/27> patch level 0)
......
(/usr/local/texlive/2012/texmf-dist/tex/platex/jsclasses/jsarticle.cls
Document Class: jsarticle 2010/03/14 okumura
)
Runaway argument?
{document｝ どんなふうに仕上がるかな?
! Paragraph ended before \begin was complete.
<to be read again>
                   \par
l.5
```

```
? ■      ← コマンド入力待ち
```

処理中のすべてのメッセージは、拡張子.log のついたログファイル sample.log として自動的に保存されますから必要に応じて随時参照します。

このようにエラーが原因で処理がとまってしまった場合、LaTeX システムに入力できる代表的な対処コマンドキーには次のものがあります。

- h　エラー原因をアドバイスしてくれる（あまり役に立たない）
- x　処理を強制終了する（処理を中止してエディタで修正する場合は、このコマンドを使います）
- q　エラーや警告を無視して処理を実行してしまう
- ?　コマンド一覧の表示
- ↵　エラーに構わず次の処理を行なう

日本語の全角記号や全角空白入力によって生じるエラーは少なくありません。そのような日本語独特の課題によって生じるエラーを発見しやすくするためには、半角空白文字や全角空白文字を区別して表示をさせるといったテキストエディタ機能を存分に利用するとよいでしょう。

エラーがあったとき、通常は 'x' を入力して LaTeX 処理を中断し、起動してあるエディタ画面に戻って修正・保存します。エラー箇所を修正したファイルを保存して、それを再びコンパイルして DVI ファイル作成を試みます。こうして LaTeX 構文エラーがなくなるまで以上の作業を繰り返します。最終的には LaTeX システムから次のようなメッセージ（以下の例は、MacOS で MacTeX を利用した場合)。

```
... 利用している OS や使っている TeX システムに応じたメッセージが表示される
This is e-pTeX, Version 3.1415926-p3.3-110825-2.4 (utf8.euc) (TeX Live 2012)
 restricted \write18 enabled.
.....
(./test.tex
pLaTeX2e <2006/11/10> (based on LaTeX2e <2011/06/27> patch level 0)
.........
(/usr/local/texlive/2012/texmf-dist/tex/platex/jsclasses/jsarticle.cls
Document Class: jsarticle 2010/03/14 okumura
) (./test.aux) [1] (./test.aux) )
Output written on test.dvi (1 page, 1028 bytes).
Transcript written on test.log.
```

が表示され、DVI ファイル sample.dvi が生成されたことがわかります（確認するにはどうすればよいかわかりますか)。この例では短い文ですから 1 ページしかありませんが、長い文のときには [1][2][3]··· とページ数が増えていきます。

3.2.3 DVI ファイルのプレビュー

現在の LaTeX システムではプレビューするには 3 通りの方法があります。1 つ目は DVI ファイルをプレビューするソフトウエアを使うやり方と、2 つ目は DVI ファイルを PostScript ファイルに変換（5.5 節）してからプレビュー・印刷するやり方、3 つ目は DVI ファイルを PDF ファイルに変換（5.4 節）してからプレビュー・印刷するやり方です。

多くの TeX ユーザはエディタ機能を備えた統合環境を利用したり、エラー処理、プレビューおよび修正作業を自動化するスクリプトを利用して、今日では PDF ファイルを生成してプレビューする 3 つ目の方法が標準的となっています。したがって、以下の DVI ファイルのプレビューに関する記述は事実上不要でしょう。実際、LaTeX 利用者や Linux 利用者以外のユーザに DVI ファイルや PostScript ファイルメールに添付してプレビューできるシステム環境を整えている利用者は多くはありません。

生成された DVI ファイル *sample.dvi* ファイルからいきなり印刷せずに、プレビューアを使って印刷イメージを確認します。作成された DVI ファイルのプレビューは、Windows では「dviout for Windows」が有名です。Linux ではプレビューアとして xdvi を使って DVI ファイルの印刷イメージを次のようにして確かめることができます。

```
% xdvi sample.dvi
```

5 節で述べるように、PostScript 画像（正確には Encapsulated PostScript ファイル）が張り込まれている DVI ファイルを画像といっしょにプレビューするためには Ghostscript のインストールが必要です。節 2.1 で紹介した TeX システムのインストールに従ったならば、既にインストールされているはずです。

3.2.4 DVI ファイルの印刷

プレビューで印刷イメージを確認し終わって、それが期待どおりであればプリンタに印刷します。

Windows ではプレビューア dviout for Windows、Mac では xdvi などの DVI プレビューアから直接印刷できます。Linux などからポストスクリプトプリンタを使う場合には、DVI ファイルを Postscript コードに変換する dvipsk（または dvips）を使って次のようにして印刷することができます。

```
% dvipsk sample.dvi
```

今日の TeX 作業のゴールは Adobe Reader でプレビューすることができる PDF ファイルの生成です。PDF ファイルは DVI ファイルから次のコマンドで生成します（5.4 節）。

```
% dvipdfmx sample.dvi
```

プレビューすると、次のような結果が得られるでしょう。

初めての TEX 文書です。どんなふうに仕上がるかな?
とにかくエディタで文書さえ書いておけば、後で LATEX ファイルにすることは簡単。
やっぱり文章は見てくれより中身が大切だからね。
でも、こんなこと

$$\int_0^\infty \frac{\sin ax}{x} dx = \frac{\pi}{2} \quad (a > 0)$$

が書けるとなると、外見も大切かなあと思ってしまうよね。

3.3 LATEX ファイルと表示

3.3.1 LATEX コマンド

5 ページの LATEX ファイル例には、たとえば ‘\LaTeX{}’ というような特別な書き方を含んでいます。DVI ファイルを作成して印刷するとこれは ‘LATEX’ のように印刷されます。

このように LATEX ファイルでは、バックスラッシュ記号 ‘\’ に続いたある特別な文字列を使って特別な印刷結果や組版上の効果を得ることがあります。これらを LATEX コマンドといい、決められた書き方をしなければなりません。LATEX システムが前もって定めていて勝手には変更できない言葉を予約語といいます。

> たとえば、‘\TeX{}’ とすると ‘TEX’ という結果になります。‘\TeX’ はコマンドとして予約されていますが、‘\tex’ や ‘\Tex’ はコマンドとして予約されていないので LATEX システムはエラーを出します。また、“\LaTeX 文書は美しい” のように予約語 ‘\LaTeX’ に続けて他の言葉を書くと、‘\LaTeX 文書’ などが LATEX コマンドとしてシステムに解釈されることになります。しかし、そのようなコマンドは LATEX システムには登録されていないのでエラーとなります。

LATEX ファイル中の文字列のどこが LATEX コマンドであるかを明示するために、次のような工夫をするとよいでしょう。

- ‘\LaTeX{}文書は美しい’ のようにコマンドの直後に ‘{}’ を追加する
- ‘\LaTeX␣文書は美しい’ のようにコマンドの直後に半角空白を挿入する
- ‘{\LaTeX}文書は美しい’ のようにコマンドの有効範囲を定めるために ‘{’ と ‘}’ で囲んで ‘\LaTeX’ がコマンドであると明示する。

3.3.2 コメントの活用

LATEX では記号 ‘%’ 以降から行末までコメントとして扱われます。不要となった文章やファイルに関する補助情報などをコメントにしておくと後で活用できます。また、コメント機能には LATEX ファイルの分割（38 ページ）を使って文書の内容を制御する大切な役割があります。

ここで、行末というのはエディタ上で Return キー（あるいは Enter キー）を押して改行記号を入力した箇所を意味します。エディタから見れば、改行記号から改行記号までの文字列が論理的な 1 行とみなされます。

3.3.3 改行と段落の取り扱い

上の例で、エディタ上での LATEX ファイル内容とその印刷結果（またはプレビュー画面）を見比べてわかるように、LATEX ファイルでの改行と印刷出力での対応は次のようになっています。

- LATEX ファイル内の単なる改行は、印刷出力では改行とはならない。文章途中でいくら改行しても、同

じ段落を構成する一連の文章とみなされる。

- 印刷出力において段落を改めるためには、LaTeX ファイルで一つ以上の空行（行頭で改行すること）を入れるか、または行末でコマンド‘\par’を記入する。このとき、段落始めとなる文章が行頭から書いてあっても、出力されると適当にインデント（字下げ）が行なわれる。
- LaTeX ファイル内で行末にコマンド‘\\’を記入すると、出力では強制改行される。このとき、改行後の行頭文字はインデントされない。

このように LaTeX システムでは、段落と段落の区切りには（一つ以上の）空行またはコマンド\par がその役目を果たします。したがって、LaTeX ファイルの単なる改行では段落が改まらないことを利用して、エディタで LaTeX ファイルを作成するときには「1 文で改行して 1 行とする」ように書くとテキスト編集の能率が向上するでしょう。

3.3.4　半角文字と全角文字

日本語モードで入力した文字を**全角文字**といい、テキスト画面上では下の表のように全角文字は半角文字の倍の文字幅となっています。

半角文字	{abcdefg}
全角文字	｛ａｂｃｄｅｆｇ｝

コンピュータにとっては全角文字と半角文字は異なる文字として識別されています。したがって、エディタを使っているときに半角文字か全角文字のどちらの種類の文字を入力しているかを常に意識しておかなければなりません。

LaTeX ファイルでは、上の例でエラーの原因となったような‘\’や‘{’,‘}’などの特殊文字や記号はすべて半角文字で書かなければなりません。特に空白文字には気を付けます。モニタに表示されるとき、半角空白‘␣’と全角空白‘　’は区別がつきにくいからです。半角空白文字と全角空白文字を区別して表示できるようなエディタを使うとよいでしょう。

3.4　LaTeX の特殊記号

LaTeX システムで特殊な役割をする記号文字が定められています。LaTeX ファイルにおいて特殊な意味を持つ記号には次のものがあります。詳しくは節 4.13（20 ページ）を参照してください。

```
%　{　}　&　#　$　^　~　\　_
```

これらを、その記号自体として LaTeX 文書として印刷するためには、表 1 のように、バックスラッシュ記号 \ をつけて特殊記号の意味をエスケープさせるか、LaTeX コマンド『\verb|...|』を使います。

4　文書のレイアウト

文書は、それを印刷したときにその内容がわかりやすいように文字の大きさや配置を工夫することが必要です。これを文書の**レイアウト**といいます。

出力	入力	出力	入力	出力	入力
%	\%	{	\{	}	\}
&	\&	#	\#	$	\$
^	\verb|^|	~	\verb|~|	\	\verb|\\|
_	_				

表 1　LaTeX における特殊記号の出力

4.1　表題の出力

文書に表題を付けるには、たとえば次のようにします。

```
\documentclass{jarticle}% 以下、日本語論文スタイル用には jsarticle が推奨です
\title{サルかに合戦顛末記}
\author{足柄金太郎}
\date{昔々}
\begin{document}
\maketitle % 表題出力コマンド
......
\end{document}
```

4 つのコマンド \title{..}, \author{..}, \date{..}, および \maketitle は 1 組になって LaTeX の表題要素を構成します。これらの表題要素の順番は違っていても構いませんが、\documentclass{...} と \begin{document} の間のプリアンブルの部分に記述します。実際に表題を出力するためには、上のように \begin{document}の次の行に LaTeX コマンド\maketitle を記述しなければなりません。表題要素がプリアンブル部に記述してあっても、\maketitle が本文になければ表示されません。

表題については次の点に留意してください。

- \maketitle を書かなければ表題は出力されません。
- 出力される表題要素はタイトル、著者、日付の順に表示されます。
- 表題にかかわるコマンドから省略したい項目のコマンドを省略することは “できません”。ただし、タイトル、著者、日付のいずれかを省力したい場合は、該当するコマンドの引数{...}内に何も記入しないで ‘{}’ とします。
- \date{\today} とすると、コンパイルした日付となります。\date{..}内に好みの書式で日付を書き込んでも構いません。

4.2　目次の作成

7 節で説明するように、LaTeX では\documentclass{.....}に指定する文書スタイルに応じて「文書の論理構造」を指定するコマンド使うことによって明瞭な文書構造を持つ文書を簡単に作成することができます。

ここでいう文章の論理構造あるいは（単に文章構造ともいいます）とは、たとえば第一章の表題、第一節の表題など書籍の目次などにみられるような文書をブロックに区切ることを指しています。このとき、コマンド `\tableofcontents` を一行書くだけで自動的に文書の目次を作成することができます（7.3.2 節（36 ページ）参照）。

4.3 書式と環境

文章のある部分を一定の書式で記述したい場合があります。たとえば、指定した文章範囲を引用文だとわかるように記述するとか、中央に寄せて記述するとか、箇条書となるようになどです。このような目的のために、LaTeX では環境 (environment) という方法を用意しています。

文のある部分を指定した環境下に置くには次のような書式で記述します。環境の中で書体を変えても、その影響は環境の外には及びません。

```
\begin{環境名}
  環境に支配される文章
\end{環境名}
```

以下に、LaTeX 環境のいくつかを紹介します。

4.4 文の引用

文章中に ‘他の文章’ や誰かの ‘発言’ などををそのまま借用するために、その文の両端のマージンを余分にとってレイアウトするとわかりやすいことがあります。このような文章の表記を文の引用といいます。LaTeX には quote と quotation の二つの引用環境があります。

4.4.1 quote 環境

quote 環境の中の文章は、 l 行の長さが短くなり左右に同じだけの空白が入ります。段落の切れ目は空白行で表しますが、“空白行がそのまま残り段落の最初の文字の字下げは行なわれません”。また、quote 環境の前後にはやや広めの空白が確保されます。たとえば、次のような言葉を引用してみましょう。

```
\begin{quote}
さるかに合戦の勃発についての歴史上の驚異を探査するための我々の現在の方法の多くを提供したのは猿蟹大学の浦島田太郎の洞察であった。
実際、彼は合戦をその時代における歴史システムのダイナミックスとして理解することの重要性を強調したのであった。

彼の多くの示唆はその後、他の人により精密化され拡張されたが、我々が彼の創造力と洞察に
負っていることは誇張し過ぎることはない。

\hfill 花咲爺、『さるかに合戦の考古学』
\end{quote}
```

このとき出力は次のようになります。

> さるかに合戦の勃発についての歴史上の驚異を探査するための我々の現在の方法の多くを提供したのは猿蟹大学の浦島田太郎の洞察であった。実際、彼は合戦をその時代における歴史システムのダイナミックスとして理解することの重要性を強調したのであった。
> 彼の多くの示唆はその後、他の人により精密化され拡張されたが、我々が彼の創造力と洞察に負っていることは誇張し過ぎることはない。
>
> 花咲爺、『さるかに合戦の考古学』

4.4.2 quotation 環境

quotation 環境は quote 環境とほぼ同じです。しかし、空白行で表される段落の切れ目では、通常の文と同じように “字下げ” によって段落の切れ目を表します。このとき quote 環境と違って空白行は表れません。たとえば、上の文例を quotation 環境で使うと出力は次のようになります。

> さるかに合戦の勃発についての歴史上の驚異を探査するための我々の現在の方法の多くを提供したのは猿蟹大学の浦島田太郎の洞察であった。実際、彼は合戦をその時代における歴史システムのダイナミックスとして理解することの重要性を強調したのであった。
> 彼の多くの示唆はその後、他の人により精密化され拡張されたが、我々が彼の創造力と洞察に負っていることは誇張し過ぎることはない。
>
> 花咲爺、『さるかに合戦の考古学』

4.5 文を寄せる

文章を揃えて左寄せにしたり、中央寄せや右寄せに配置したい場合があります。このようなときには、それぞれ **flushleft 環境**、center 環境そして **flushright 環境**を使います。文を寄せる際には、強制改行コマンド ‘\\’ を利用することがあります。

> 強制改行の場合には、改行後の行頭の文字の字下げは行なわれません。3.3.3 節で説明したように、LaTeX では空白行または \par は段落の切れ目となり、改行された上で新段落の行頭文字が字下げされます。

center 環境　改行で区切られた文や指定した図表を中心に配置することを中寄せといいます。
flushleft 環境　改行で区切られた文や指定した図表を左に配置することを左寄せといいます。
flushright 環境　改行で区切られた文や指定した図表を右に配置することを右寄せといいます。

これらの文を寄せる環境を使って次のように書いてみます。

```
\begin{flushright}
\LaTeX{}は世界中で\\
利用されている\\
文書整形の\\
定番です
\end{flushright}
```

```
\begin{center}
誰にでもできる\\
簡単な\\
入門
\end{center}
\begin{flushleft}
Linux、\\
Macintosh や\\
Windows でも OK です
\end{flushleft}
```

このとき出力は次のようになります。

LaTeX は世界中で
利用されている
文書整形の
定番です

誰にでもできる
簡単な
入門

Linux、
Macintosh や
Windows でも OK です

4.6 擬似タイプ入力

コンピュータプログラムを掲載したり、文書の一部をタイプしたときと同じように出力したいときがあります。また、LaTeX コマンドを説明する場合には、特殊記号を含んだ LaTeX ファイルの生原稿を出力する必要があります。このような目的のために、生原稿をそのまま出力する環境として **verbatim 環境**を使います。たとえば、本節の文頭の生原稿を出力するには次のように書きます。

```
\begin{verbatim}
\subsection{擬似タイプ入力}
....
\LaTeX{}コマンドを説明する場合には、特殊記号を含んだ\LaTeX{}ファイルの生原稿を出力する
必要があります。
このような目的のために、生原稿を\textbf{そのまま出力}する環境として
```

```
\textbf{verbatim 環境}....
\end{verbatim}
```

このとき出力は次のようになります。

```
\begin{verbatim}
\subsection{\texttt{verbatim}環境}
\LaTeX{}コマンドを説明する場合には、特殊記号を含んだ\LaTeX{}ファイルの生原稿を出力する必要があります。
また、ソースプログラムを掲載することも必要になるでしょう。
\end{verbatim}
```

そのまま出力したい‘生原稿部分’を文中で表すには、‘\verb’コマンドを使います。\verb の直後の文字（たとえば‘|’や‘+’）が次に表れるまでに囲まれた文字列が、たとえそれが LaTeX コマンドであろうとも本来の意味機能を停止して、そのまま出力されます。たとえば

```
\verb|\LaTeX{}|文書は美しい
```

と書けば、‘\LaTeX{}文書は美しい’と出力されます。

環境名において verbatim*、または文中で‘\verb*’のように * を加えたときには指定された範囲の生原稿がそのまま出力されるのは同じですが、半角空白文字が‘␣’と表されるので使い分けると便利でしょう。

4.7 箇条書

箇条書をするための環境として **itemize** 環境、**enumerate** 環境および **description** 環境の 3 つがよく利用されており、それぞれ単純箇条書、列挙箇条書、見出し付箇条書と呼びます。

4.7.1 itemize 環境

箇条項目に先立って‘\item␣’をつけて箇条書きします。\item の後に空白‘␣’があることに注意してください。単純箇条書では、各箇条項目の前に印‘•’が付きます。単純箇条書は箇条項目の順番を入れ替えても問題が起こらないときに使います。箇条項目の順番に意味があるときには列挙箇条書を採用すべきです。

- itemize 環境では箇条項目に • が付きます。
- enumerate 環境では箇条項目に番号が付きます。
- description 環境では箇条項目に見出しを付けることができます。

と出力するためには、次のように書きます。

```
\begin{itemize}
\item itemize 環境では箇条項目に$\bullet$ が付きます。
\item enumerate 環境では箇条項目に番号が付きます。
\item description 環境では箇条項目に見出しをつけることができます。
\end{itemize}
```

4.7.2 enumerate 環境

列挙箇条書環境でも、単純箇条書とまったく同じように箇条項目に先立って '\item␣' を付けます。列挙箇条書では、各箇条項目の前に列挙した順に番号が振られます。

箇条項目の順番を入れ替えても問題がない場合には単純箇条書にすべきです。単純箇条書にするか列挙箇条書にするかは、その理由を考えてから適切な箇条書環境を選んでください。

1. itemize 環境では箇条項目に • が付きます。
2. enumerate 環境では箇条項目に番号が付きます。
3. description 環境では箇条項目に見出しをつけることができます。

と順序数を付けて出力するには、次のように書きます。

```
\begin{enumerate}
\item itemize 環境では箇条項目に$\bullet$ が付きます。
\item enumerate 環境では箇条項目に番号が付きます。
\item description 環境では箇条項目に見出しをつけることができます。
\end{enumerate}
```

4.7.3 description 環境

箇条項目に先立って '\item[...]␣' を付けて [...] 内に箇条項目の見出しを書いて、見出し付き箇条書とします。

単純箇条書　itemize 環境では箇条項目に • が付きます。
列挙箇条書　enumerate 環境では箇条項目に番号が付きます。
見出し付き箇条書　description 環境では箇条項目に見出しを付けることができます。

と出力するには、次のように書きます。

```
\begin{description}
\item[単純箇条書] {\tt itemize}環境では箇条項目に$\bullet$ が付きます。
\item[列挙箇条書] {\tt enumerate}環境では箇条項目に番号が付きます。
\item[見出し付き箇条書] {\tt description}環境では箇条項目に見出しを付けることができます。
\end{description}
```

4.8 ネストされた環境

¥LaTeX の環境では、次のように環境内の文の中に環境をネスト、つまり環境文を多重化させることができます。

```
\begin{環境 A}
   ......
   \begin{環境 B}
      ......
      \begin{環境 D}
         .......
      \end{環境 D}
      ......
   \end{環境 B}
   ......
   \begin{環境 C}
     ........
   \end{環境 C}
\end{環境 A}
```

上の例では、環境 A の文書の中に環境 B と環境 C が使われています。さらに、環境 B の中には環境 D が使われています。このように環境の中でさらに環境を使うことを**環境のネスト**といいます。

ネストさせる環境は、原則としてどんなものでも構いません。たとえば、単純箇条書の環境を次のようにネストさせてみましょう。

```
\begin{itemize}
\item 単純箇条書第 1 レベル
    \begin{itemize}
    \item 単純箇条書第 2 レベル
        \begin{itemize}
        \item 単純箇条書第 3 レベル
            \begin{itemize}
            \item 単純箇条書第 4 レベル
            \end{itemize}
        \end{itemize}
    \end{itemize}
\item 単純箇条書第 2 レベル
\end{itemize}
```

すると、次の出力結果が得られます。

- 単純箇条書第 1 レベル
  - 単純箇条書第 2 レベル

        - 単純箇条書第 3 レベル
            - 単純箇条書第 4 レベル
- 単純箇条書第 2 レベル

同様に、列挙箇条書の環境を次のようにネストさせてみましょう。

```
\begin{enumerate}
\item 列挙箇条書第 1 レベル
    \begin{enumerate}
    \item 列挙箇条書第 2 レベル
        \begin{enumerate}
        \item 列挙箇条書第 3 レベル
            \begin{enumerate}
            \item 列挙箇条書第 4 レベル
            \end{enumerate}
        \end{enumerate}
    \end{enumerate}
\item 列挙箇条書第 2 レベル
\end{enumerate}
```

すると、次の出力結果が得られます。

1. 列挙箇条書第 1 レベル
    (a) 列挙箇条書第 2 レベル
        i. 列挙箇条書第 3 レベル
            A. 列挙箇条書第 4 レベル
2. 列挙箇条書第 2 レベル

これらの箇条書きを入れ子にしたときに、その深さに応じて自動的につけられる項目記号は再定義することができます（奥村 [4, 第 5 章箇条書き]）。

いままでの例からもわかるように、LaTeX は文のレイアウトや後で述べるように文書クラスを定めることによって最高度に美しい整形出力を実現するシステムです。最初からエラーのない文書を書くことは誰にもできません。LaTeX システムでのコンパイルとエラーの修正を繰り返しながら徐々に完成していくのが一般的な過程です。そのために、できる限りエラーの見つけやすい文書書法を心掛けましょう。LaTeX ファイルでは、その印刷結果が同じであってもかなり自由に書くことができます。プログラム言語を記述するときと同じ態度ですが、字下げ等を使って文書の論理構造を明示する書き方を踏襲するなどの工夫をするとよいでしょう。

4.9 脚注を入れる

LaTeX では、簡単に脚注を付けることができます脚注には文中で

`\footnote{..}`
を使って `{..}` 内に脚注を記述します。脚注をイタズラに多用すると読み手の視線が散乱して読むリズムが狂うことがあるので注しましょう。脚注[*5]では、本文の該当箇所に脚注を表す印が付けられ、同一ページ下のフッタの中に脚注文が表示されます。ここで書いた脚注は次のように書きました。

> 脚注
> `\footnote{`これが脚注です。あまり多用すると文章が読みづらくなります。`}`
> では、本文に脚注を表す印が付けられ、同一ページ下のフッタの中に脚注文が表示されます。

4.10 LaTeX で使われる文字

4.11 基本文字サイズ

出版の世界では、文字の大きさをポイントという単位で表すことがあります。これは 1 インチを約 72 ポイントと考えて文字の大きさを計るやり方です。

LaTeX 文書の印刷仕上がりは、特に指示をしない場合には欧文文字が 10 ポイントの大きさで印刷されます。これでは見にくいと感じるならば、欧文文字を 11 ポイント、または 12 ポイントで印刷するオプションを指定することができます。このためにはファイル先頭の `\documentclass{...}`の箇所で次のようにオプションを指定します。

11 ポイント	`\documentclass[11pt]{...}`
12 ポイント	`\documentclass[12pt]{...}`

4.12 フォントの種類

印刷の世界では同一種類の活字 1 セットをフォント (Font) といいます。文書として効果的な出力結果を得るために、フォントの大きさや種類を変えることは出版の世界で広く行なわれてきました。LaTeX でもフォントの種類やその大きさを変更することができます。ただし、フォントを変更しなければならない理由をよく考えるべきで、むやみに多くのフォントを多用すると読みづらい結果になってしまうことがあります。

4.12.1 日本語フォント

標準的に利用できる LaTeX の日本語フォントは、原則として明朝体族とゴシック体族の 2 書体です。特に指定しなければ、日本語全角文字は明朝体で印刷されます。文書中の日本語文字列をゴシック体 (Gothic) に変更するには, 次のように指定します。

ここの部分が`\textgt{`ゴシック体`}`になる

あるいはボールド体 (太字) を指定する `\textbf{...}` を使っても同じ効果が得られます（論理的には、この`\textbf{...}`を使うほうがベターでです)。

[*5] これが脚注です。あまり多用すると文章が読みづらくなります。

4.12.2　英文フォント

LaTeX で利用できる英文書体には次の表 2 にあるフォントがあります[*6]。文書中の英文文字列のフォントを変更するには表 2 のようにして文字列範囲を指定します。

出力	入力	出力	入力
Roman	`\testrm{Roman}`	**Boldface**	`\textbf{Boldface}`
Emphasis	`\emph{Emphasis}`	Sans Serif	`\textsf{Sans Serif}`
Italic	`\textit{Italic}`	typewriter	`\texttt{typewriter}`
Small Caps	`\textsc{Small Caps}`		

表 2　英文フォントを指定するコマンド

4.13　シンボル・特殊記号の表現

TeX システムは本来、数式が入り混じった文書の組版を目指して D. Knuth によって開発されました。TeX ファミリで組版される数式は、どんな複雑なものでも美しく出力されます。ここでは数式の取り扱いは述べませんが、奥村 [4] などを参照してください。

LaTeX で用意されている豊富な記号を利用するためには数式モード内で記号指定をします。数式モードとは `$` で始まり `$` で閉じる特別な状態を意味しています。たとえば、♡ を表すためには　`$\heartsuit$` と記入します。

4.13.1　ギリシャ文字

ギリシャ文字を使う場合にも数式モード内で表 3 にあるように指定します。

[*6] jarticle、jreport や jbook などの文書クラスを指定した場合、数式で使われるフォントとして Computer Modern が使われます。

出力	入力	出力	入力	出力	入力
α	\alpha	β	\beta	γ	\gamma
δ	\delta	ϵ	\epsilon	ε	\varepsilon
ζ	\zeta	η	\eta	θ	\theta
ϑ	\vartheta	ι	\iota	κ	\kappa
λ	\lambda	μ	\mu	ν	\nu
ξ	\xi	$\o$	\o	π	\pi
ϖ	\varpi	ρ	\rho	ϱ	\varrho
σ	\sigma	ς	\varsigma	τ	\tau
υ	\upsilon	ϕ	\phi	φ	\varphi
χ	\chi	ψ	\psi	ω	\omega
Γ	\Gamma	Λ	\Lambda	Σ	\Sigma
Ψ	\Psi	Δ	\Delta	Ξ	\Xi
Υ	\Upsilon	Ω	\Omega	Θ	\Theta
Π	\Pi	Φ	\Phi	$\sum$	\sum
$\prod$	\prod				

表 3　ギリシャ文字群

4.13.2　記号

ギリシャ文字以外にも、表 4 にあるように多くの記号やシンボルが数式モードで使うことができます。

出力	入力	出力	入力	出力	入力
$\pm$	\pm	$\mp$	\mp	$\times$	\times
$\div$	\div	$*$	\ast	$\star$	\star
$\circ$	\circ	$\bullet$	\bullet	$\cdot$	\cdot
$\diamond$	\diamond	$\bigtriangleup$	\bigtriangleup	$\bigtriangledown$	\bigtriangledown
$\triangleleft$	\traiangleleft	$\triangleright$	\traiangleright	$\bigcirc$	\bigcirc
$\dagger$	\dagger	$\leq$	\leq	$\geq$	\geq
$\equiv$	\equiv	$\sim$	\sim	$\simeq$	\simeq
$\approx$	\approx	$\neq$	\neq	$\propto$	\propto
$\perp$	\prep	$\parallel$	\parallel	$\leftarrow$	\leftarrow
$\Leftarrow$	\Leftarrow	$\rightarrow$	\rightarrow	$\Rightarrow$	\Rightarrow
$\leftrightarrow$	\leftrightarrow	$\Leftrightarrow$	\Leftrightarrow	$\longleftarrow$	\longleftarrow
$\Longleftarrow$	\Longleftarrow	$\longrightarrow$	\longrightarrow	$\Longrightarrow$	\Longrightarrow
$\uparrow$	\uparrow	$\Uparrow$	\Uparrow	$\downarrow$	\downarrow
$\Downarrow$	\Downarrow	$\updownarrow$	\updownarrow	$\Updownarrow$	\Updownarrow
$\nearrow$	\nearrow	$\searrow$	\searrow	$\swarrow$	\swarrow
$\nwarrow$	\nwarrow	$\angle$	\angle	$\flat$	\flat
$\natural$	\natural	$\sharp$	\sharp	$\backslash$	\backslash
∂	\partial	∞	\infty	$\triangle$	\triangle
$\clubsuit$	\clubsuit	$\diamondsuit$	\diamondsuit	$\heartsuit$	\heartsuit
$\spadesuit$	\spadesuit				

表 4　LaTeX で利用できる記号・シンボル（一部）

$\mathcal{AMS}$-TeX 用のスタイルファイル amsmath と amssymb [*7] を使うためにプリアンブル部に

```
\usepackage{amsmath,amssymb}
```

と記述すると、さらに利用出来る数学記号がぐんと増加します。たとえば、ギリシャ文字 π (`$\pi$`) の太字を`$\boldsymbol{\pi}$`で $\boldsymbol{\pi}$ とできることや、不等号記号も `$\leqq$`で $\leqq$、`\geqq` で $\geqq$、`$\lneqq$`で $\lneqq$、`\gneqq` で $\gneqq$ などと多彩な記述が可能になります。数式を多用する文書を書くときには $\mathcal{AMS}$-LaTeX での記述を使うとぐんと作業が楽になるため、調べてみる価値があります。たとえば、

```
\[
\cfrac{1}{\sqrt{2}+ \cfrac{1}{\sqrt{2}+
\cfrac{1}{\sqrt{2}+\dotsb }}}
\]
```

[*7] $\mathcal{AMS}$-LaTeX のスタイルファイルの入手には TeXWiki/AMS-LaTeX や本家のアメリカ数学会 `http://www.ams.org/publications/authors/tex/amslatex` から。

と書くだけで、次のような連分数を書くことができる。

$$\cfrac{1}{\sqrt{2}+\cfrac{1}{\sqrt{2}+\cfrac{1}{\sqrt{2}+\cdots}}}$$

4.13.3　特殊文字とアクセント記号

特殊文字は表 5 のようにして出力します。‘ı’ や ‘ȷ’ のように、ドットのない英字を出力できることに注意してください。これらは、以下のようにアクセント記号を組み合わせて利用します。またハイフン ‘-’ と 2 分ダッシュ ‘–’ と全角ダッシュ ‘—’ の使い分けにも留意してください。

出力	入力	出力	入力	出力	入力
%	`\%`	{	`\{`	}	`\}`
#	`\#`	$	`\$`	$	`\$`
ˆ	`\^{}`	˜	`\~{}`	_	`\_`
\	`$\backslassh$`	§	`{\S}`	¶	`{\P}`
©	`{\copyright}`	†	`{\dag}`	‡	`{\ddag}`
£	`{\pounds}`	¥	`Y\llap=`	œ	`{\oe}`
Œ	`{\OE}`	æ	`{\ae}`	Æ	`{\AE}`
å	`{\aa}`	Å	`{\AA}`	ø	`{\o}`
Ø	`{\O}`	ß	`{\ss}`	¿	`?‘`
¡	`!‘`	ı	`{\i}`	ȷ	`{\j}`
‘	‘	’	’	“	‘‘
”	’’	-	-	–	--
—	---	LaTeX	`\LaTeX`	TeX	`\TeX`

表 5　特殊記号の印刷

フランス語などでは、通常のアルファベット以外にアクセント記号を組み合わせます。たとえば、`\c{c}` で cedilla を表し、`fa\c{c}ade` などと書きます。その他のアクセント記号を表 6 にまとめておきます。

名前	入力	出力	名前	入力	出力	名前	入力	出力
grave	`\‘{a}`	à	tilde	`\~{o}`	õ	check	`\v{s}`	š
acute	`\’{e}`	é	bar	`\={y}`	ȳ	long	`\H{j}`	ȷ̋
hat	`\^{o}`	ô	dot	`\.{p}`	ṗ	tie-after	`\t{\i u}`	ı͡u
umlaut	`\"{u}`	ü	breve	`\u{\i}`	ĭ	dot-under	`\d{h}`	ẖ

表 6　LaTeX のアクセント記号

5　画像ファイルの取り込み

LaTeX 文書中に別に用意した画像ファイルを取り込んで一緒に出力することができます。その具体的な情報は、たとえば TeXWiki の [TeX 入門/図表] にあります。

最近の TeX 環境では、取り扱う画像ファイルは後で述べる EPS 形式が原則ですが、さまざまな形式の画像を貼り込めるようになっています。ただし、そのためには画像の配置情報である BoundingBox のパラメータ（画像のピクセル位置）を TeX システムに知らせなければなりません。

画像ファイルにはさまざまな形式、たとえば PNG, JPEG, PDF や EPS 形式などがあります。たいていの場合、ファイル拡張子によって `*.png`, `*.jpg`, `*.pdf`, `*.eps` などのように区別されるようになっているはずです。EPS 形式画像はその情報をファイル内に保持しています。一方、PNG, JPEG, PDF 形式の画像ファイルは BoundingBox 情報を保持していないため、PNG, JPG および PDF 形式の画像を LaTeX で貼り込むためには、貼り込み指定時に画像ファイルの BoundingBox 情報を直接指定するか、以下で説明するように BoundingBox 情報を別ファイルに書き出す前処理を行う必要があります。

携帯や iPhone などのスマートフォン、ファミリー用途のデジカメでは JPEG 形式で画像ファイルが保存されます。一方、精緻なイラストなどは、拡大縮小してもジャギーの現れないように Adobe Illustrator や GIMP などのソフトウエアを使って Postscript 形式で作成することがあります。PNG, JPG 形式の画像ファイルは Adobe PhotoShop/Illustrator や GIMP を使って、BoundingBox 情報を持つカプセル化 Postscript 形式（EPS 形）に変換保存することができます。

5.1　PNG/JPEG 形式の画像

TeX Wiki[TeX 入門/図表] にその実際が説明してあります。再掲すると、以下のようです。426×496 ピクセルの PNG または JPEG 形式の画像、たとえば `apple.jpg` をを貼り込むためには、次のように書きます。

```
\documentclass{jarticle}% jsarticle の利用が推奨です
...
\usepackage[dvipdfm]{graphicx}% プリアンブル部で画像の読込/加工用のパッケージ
                              %    の読み込みを宣言
....
\begin{document}
.....
\includegraphics[bb=0 0 426 496,width=5cm]{apple.jpg}
....
\end{document}
```

この例のように、プリアンブルで画像読み込み用のパッケージ読み込みを宣言し、本文中でコマンド `\includegraphics{画像ファイル}`によって指定した画像を読み込みます。ここでは `[ ]` を使ってオプションを指定しています。`bb` は BoundingBox 情報で画像配置範囲 $(0, 0)$ から $(426, 496)$ つまりサイズ 426×496

ピクセルの画像を読み込んで、それを width で指定した幅で貼り込むという指定をしています。BoundingBox やコマンド\includegraphics のオプションについては改めて 5.2 節や 5.3 節で詳しく述べます。

しかしながら問題なのは、PNG 形式や JPEG 形式の画像の読み込みの際、オプション bb で指定した画像サイズ情報（BoundingBox 情報）を予め知らねばならないことです。画像ソフトを使って貼り込みたい画像を開けばその画像サイズを知ることができ、それを使って b 情報を指定することはできます。別の方法として、\includegraphics のオプションに bb 情報を指定せずに、bb 情報を別ファイルに書き出すプログラム使う方法があります。

最近の標準的な TEX システムには、ここでも前提としている出力ドライバ dvipdfm (実行ファイル名は dvipdfmx.exe) にはプログラム ebb が同梱されています。これを使って、次のようにして BoundingBox 情報を拡張子が .bb のファイルに書き出します。

```
% ebb apple.jpg
```

すると、ファイル apple.bb に BoundingBox 情報が書き込まれます。この処理を行っておけば、bb 情報をオプションで指定せずに、\includegraphics[width=5cm]{apple.jpg}とするだけで、上の例と同じ結果を得ることができます。プログラム ebb を使う場合、Windows ではコマンドプロンプト (または「DOS 窓」ともいいます）を実行して使います（「アクセサリ」内の「コマンドプロンプトで開くか、[スタート/ファイル名を指定して実行] から cmd と入力)。Mac OS ではユーティリティフォルダ内にある「ターミナル」を起動して使います。

Winows の場合、代表的なプレビューア dviout [*8] に付属のプログラム CreateBB.exe (dviout.exe と同じフォルダにある) を使うと BB ファイルの生成がぐんと楽になります。GUI 操作によって指定した JPEG ファイルなどの BoudingBox 情報を拡張子.bb が付いた BB ファイルを生成してくれます（[File Type..] で「JPG File」にチェックを入れておきます)。

5.2 EPS 形式の画像

ここでは拡大縮小などが自由にでき、仕上がり結果が美しい PostScript 形式（正確には EPS 形式）の画像ファイルを張り込む方法を説明します。LATEX で取り込める画像ファイルの形式は、出力ソフトによって異なります。dviout for Windows では多くの形式の画像ファイルを取り込むことができますが、3.2.3、5.4 節でも述べたように、EPS ファイルを張り込むのが Macintosh、Linux などに共通する一般的方法です。

5.2.1 PostScript と EPS

PostScript とは Adobe System 社が開発したプリンタ出力のためのインタープリタ型のページ記述言語です。PostScript は、文字はもちろんのこと、イラストや写真などのグラフィックスを高度なレベルで記述する能力があり、品質を損なうことなく縮小・回転や変形などの画像操作を行なえるなどの強力な機能を備えています。PostScript で記述されたファイルはテキストファイルで、これを PostScript ファイル（あるいは単に PS ファイル）と呼びます。PS ファイルの拡張子として .ps をつけます。

Encapsulated PostScript（あるいは単に EPS）ファイルとは PS ファイルの一形式で PS 形式で単独

[*8] dvioiut で貼り込んだ PJEG/PNG ファイルをによってプレビューするためには Susie プラグインが必要です詳細はたとえば、「dvioiut に Susie プラグインを入れる」http://www10.ocn.ne.jp/~tasusu/tex/susie.html を参照してください。

の画像としてカプセル化された PS ファイルです。言い換えれば、PS ファイルからページ情報を取り去ったものといえるでしょう。EPS ファイルにプレビューのための別形式の画像ファイルを含ませることができるソフトウエアがありますが、LaTeX で利用するためには EPS 形式で保存する際にはプレビュー画像を含まない形式にします。EPS ファイルの拡張子として `.eps` または `.ps` をつけます。

PS ファイル EPS ファイルともテキストファイルです。EPS ファイルは必ず次のようなヘッダコメントから始まっています。

```
%!PS-Adobe-3.0 EPSF-3.0
%%Creator: GraphicConverter
%%Title: WINTEX.ps
%%BoundingBox: 0 0 128 62
  ..............
```

1 行目がバージョンコメントで、そのファイルが PostScript ファイルであり EPS ファイルのバージョンを表しています。EPS ファイルで最も重要な情報はコメント '`%%BoundingBox:`' で、次のような行になっています。

`%%BoundingBox:` $x_0\ y_0\ x_1\ y_1$

このコメントにはそのファイルで描く画像の位置配置と大きさを示すコメントが書かれています。こうすることによって描画領域をを定め、その外側を描かないようにするのです。画像の外枠の左下の座標が (x_0, y_0)、右上の座標が (x_1, y_1) であることを意味しており、結果的に画像サイズは横幅が $x_1 - x_0$、縦幅が $y_1 - y_0$ であることを表しています。単位は ポイント $= 1/72$ インチ です。

5.2.2 EPS ファイルの作成・変換

非 EPS 対応の描画ソフトで作成した画像データやデジタル写真画像を LaTeX 文書に張り込むために EPS 形式のファイルに変換しておくと便利です。有名なドローソフトウエアに Adobe Illustrator が、写真レタッチソフトに Adobe PhotoShop がありますが、いずれも高価です。Linux, Mac および Windows でも共通で使える画像ソフト Gimp（GNU Image Manipulation Program）が非常に重宝します[*9]。ただし、現在のヴァージョンの GIMP では EPS 形式や PDF 形式のファイルを読み込むことはできません（原則的には EPS や PDF 以外の様々な形式の画像を読み込んで EPS 形式で保存することができます。PDF 形式で保存もできません)。

5.3 EPS ファイルの張り込み

LaTeX ページに EPS ファイルを取り込むためには、次のようにプリアンブル部に`\usepackage` コマンドを使ってグラフィックスのための graphicx パッケージの読み込みを記述し、本文中で画像を張り込みたい箇所に `\includegraphics` コマンドを使って目的の EPS ファイルを指定します。

[*9] Gimp の入手は **http://www.gimp.org/**から。

```
\documentclass{jarticle}% jsarticle の利用が推奨です
...
\usepackage[dvipdfm]{graphicx}% 画像ファイルの読み込み、加工用のパッケージ
....
\begin{document}
.....
\includegraphics[オプション]{ファイル名}
....
\end{document}
```

ファイル指定は、‘この’LaTeX ファイルから目的のファイルまでの相対ディレクトリパス（または絶対パス）を区切り記号 ‘/’ を使って指定します。たとえば、sin 曲線を表わす EPS ファイル `sin.eps` を用意しておいて、次のように文中で EPS ファイル `psfile/sin.eps` を取り込む命令を記述します。

```
元のサイズです。\includegraphics{psfile/sin.eps}
高さ指定もできます。\includegraphics[height=3cm]{psfile/sin.eps}\\
拡大もできます。\includegraphics[scale=1.5]{psfile/sin.eps}
変形も\includegraphics[width=3cm,height=4cm]{psfile/sin.eps}この通りです
```

すると図 2 のような出力が得られます。

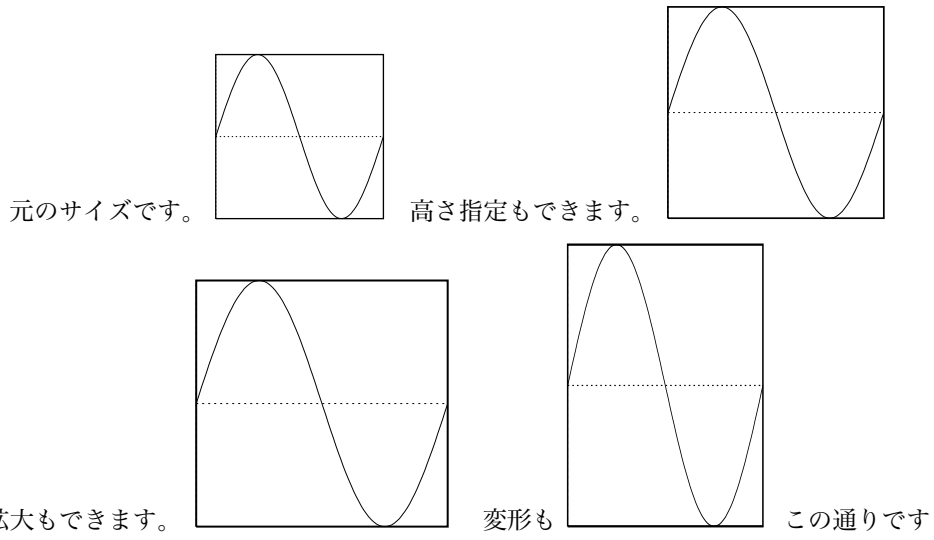

図 2　EPS ファイルの張り込み。ポストスクリプトファイルは縦横のスケールを変えても品質は変わりません。

図 2 からもわかるように、`\includegraphics{...}`で張り込まれる画像は “文中の 1 文字” として扱われます。また、拡大や変形によっても画像の品質には変化がないという PostScript の特質もうかがえます。

画像を張り込んだ個所を LaTeX 文書内で図として扱うためには、次のように **figure** 環境を使います。

```
\begin{center}
\includegraphics[scale=1.5]{psfile/sin.eps}
\end{center}
\caption{$\sin$曲線の EPS ファイルを 1.5 倍した図}
\label{sin-curve}
```

12 節で説明するように、figure 環境内で `\label{sin-curve}` とラベル名を指定しておくと、任意の文章中から `\ref{sin-curve}` によって図番号を、`\pageref{sin-curve}` によって登場ページ数を参照することができます。

EPS ファイル `epsfile.eps` を読み込む `\includegraphics` のオプションの使用法の一部を以下に示します。オプション指定をしなければ、EPS で指定されている画像の大きさで張り込むことになります。オプション指定を複数行う場合、カンマ ',' で区切りますが、カンマの前後には余分な空白を入れてはいけません。

大きさの指定　EPS ファイルの画像の大きさを指定することができ、高さを示す `height` と幅を示す `width` のキーワードが使えます。

1. `\includegraphics[height=4cm]{epsfile.eps}`
2. `\includegraphics[width=3cm]{epsfile.eps}`
3. `\includegraphics[width=3cm,height=4cm]{epsfile.eps}`

1 番目と 2 番目のように、高さまたは幅のどちらかを指定したときには、張り込まれる EPS 画像の大きさがそれに適合するように拡大・縮小されます。つまり、EPS 画像 `epsfile.eps` の大きさを幅 x、高さ y だとすると、1 番目で `height` を h とすると `width`$= h \times x/y$ として、2 番目では `width` が w とすると `height`$= w \times y/x$ として張り込まれます。　3 番目のように、`width` と `height` の両方を指定したときはその大きさになるように縦横に変形されます。

倍率の指定　画像全体の倍率は `scale` で指定します。

- `\includegraphics[scale=1.5]{epsfile.eps}`

この場合には元の大きさが縦横に 1.5 倍されます。

5.4　DVI から PDF ファイルへの変換

PDF（Portable Document Format）は Adobe System 社が開発した PostScript 言語から派生したページ記述言語です。PDF 形式のファイルを表示・印刷するビューアである Acrobat Reader は各種の OS 用に無料で配布されているために PDF ファイルを表示することはほとんど問題がなく、Web でも PDF ファイルにリンクを張っている場合も少なくありません。

EPS ファイルを張り込んでいる場合、DVI ファイル内部には画像データは含まれてはおらず、プレビューするときに Postscript プレビューアである Ghostscript が起動して EPS ファイルを読み込んで画像表示を実現します。以下に説明するように DVI ファイルを PDF ファイルに変換する場合には、画像情報も同時に取り込んで 1 つの PDF ファイルにするので、LaTeX 出力ファイルとして配布する場合には PDF に変換してお

くとよいでしょう。

Windows、Macintosh、Linux 共に、DVI ファイル（たとえば `latex.dvi` ）から PDF ファイル（`latex.pdf`）に変換するには、出力ドライバ dvipdfm(コマンド名 `dvipdfmx`) を次のように使って PDF ファイル `latex.pdf` を生成します。ただし、PostScript 画像が張り込まれている DVI ファイルを PDF ファイルに変換するためには Ghostscript が必要です。dvipdfm、Ghostscript ともに Linux、Windows、Macintosh OS X の各版が入手できます、

```
% dvipdfmx latex.dvi
```

TEX 統合環境として WinShell を使っている場合、この処理をユーザプログラムとして登録することによってボタンを押すだけで DVI ファイルから変換して Adobe Reader のような PDF ビューアーで確認することができます。

5.5 DVI から PS ファイルへの変換

DVI ファイルを dvipsk を使って張り込まれた EPS 画像ファイルデータとともに PostScript ファイルに変換することも可能です。ここでは詳しい説明をしませんので、奥村 [4, 第 13 章] などを参考にしてください。

PostScript ファイルをプレビューするための Ghostscirpt がインストールされている Windows や Macintosh パソコンは LATEX 利用者以外は多くはないでしょう。LATEX 文書情報として伝えたいのであれば、PDF ファイルかまたは LATEX ファイルそれ自体を配布する方が、読めない DVI や Postscript ファイルを受け取るよりもましというわけです。

6 簡単な作表

作表の基本は tabular 環境を使います。例えば、

1	2	3
左側	中央部	右側
Apple	Pine	Orange

のように表を作成することができます。文中の表部分は次のように書かれています。

```
\begin{tabular}{lcr}
\hline
1      &        2  &   3\\
\hline
左側 & 中央部 & 右側\\
Apple &  Pine & Orange\\
\hline
\end{tabular}
```

6.1　図表の出力位置

tabular 環境を使ったままでは、本文中に表が大きな 1 文字のように扱われてしまうので、通常は次のように table 環境内に置きます。このとき、tabular 環境をさらに center 環境ではさむと中央に位置させることができます。

1	2	3
左側	中央部	右側
Apple	Pine	Orange

表 7　簡単な表

連番が付く図表のために、LaTeX では図の場合には figure 環境、表の場合には table 環境が用意されており、以下の書式に従います。

```
\begin{table/figure}[出力位置指定]
画像の張り込みや作図または tabular 環境などによる作表
図表を中心位置に配置するには center 環境で挟む
\caption{図表の説明文}
\label{ラベル名}
\end{figure/table}
```

figure/table 環境は float 環境と呼ばれる仲間に属し、その出力位置は状況によって半自動的に決定され、ページ内または後のページへと動きます（それゆえ float なのです）。表 8 に figure/table 環境で使われる図表の‘出現位置指定’で使われるパラメータ文字の意味を示しました。

位置指定	文字の意味
h	できるだけ tabular の出現場所に出力する
t	できるだけページの先頭に出力する
b	できるだけページの末尾に出力する
p	図表だけからなるページを作成して，そこに出力する

表 8　figure/table 環境で使われる出力位置指定パラメータとその意味

この位置指定は、float な図表の出力位置に対する努力目標としての意味しかないことに注意してください。たとえば、[htb] と指定すると、できれば‘この’位置に、次いでページトップに、それでも無理ならページボトムにという意味になります。したがって、TeX では\label{ラベル名}でラベル名を指定し、\ref{ラベル名} でそのラベルを参照して図表の番号を取得しながら文を書くようにするのがよいのです。

図表の位置をここだと‘絶対指定’したければ、プリアンブル部で

```
\usepackage{here}
```

とパッケージ here を読み込んだ上で、figure/table の出力位置指定で [H] とすれば、「その」場所に出力されるようになります。

6.2 tabular 環境の書式

tabular 環境は次のような書式を持ちます。

tabular 環境

```
\begin{tabular}{位置書式}
.....
表項目の並び
.....
\end{tabular}
```

ここで、位置書式（左右寄せパラメータ）には次のような指定が可能です。表の各行の項目数は位置書式で指

記号	意味	備考
l	項目を左寄せ (Left) にする	小文字の L
c	項目を中寄せ (center) にする	
r	項目を右寄せ (Right) にする	
|	項目間に縦罫線を引く	縦棒
||	項目間に 2 重縦罫線を引く	縦棒 2 本

表 9　tabular 環境の位置書式

定した数以下でなければなりません。項目間は記号 & で区切り、各行の終わりには 2 つのバックスラッシュ（強制改行コマンド） \\を書きます。強制改行までの項目数が不足していれば，残りは空白項目として扱わます。

tabular 環境内の項目として表 10 のものが利用できます。

6.3 作表における技巧

tabular の列の位置書式（左右寄せパラメータ）を指定する引数内で p{ 幅指定 } を使って幅指定した段落モードにすることができます。段落の幅は、たとえば 0.3\textwidth とするとテキスト幅の 30% となります。

\hline	行の先頭（または最後）だけに書くことができる。\hline が置かれている「場所」から，表を横切る横罫線を表の幅の最後まで引く。\hline\hline と続けると 2 重横罫線になる。
\vline	縦罫線
\cline{start-end}	start 番目から end 番目までのカラムに横罫線を引く。
\multicolumn{num}{pos}{item}	複数のカラムにまたがる項目を作成するときに使う。num はまたがるカラム数で、それで確保された空間に pos で指定された水平方向の位置に、項目 item を置く。pos は l（L 小文字）, c, r の何れかの 1 文字を含み，|（縦棒） を含んでも構わない。

表 10　tabular 環境における特別な表要素

雨の日の注意事項	傘と雨具、レインシューズをはくのはもちろんですが、雨の日は足元が滑りやすくなり危険なことが多くなります。
晴天の日の注意事項	十分な水分を取り、着替を持参する。ただし、水分を過剰に摂りすぎたり、喉越しはよいけれど栄養価の低いものばかり食べてしまったり、体に負担をかける飲食には注意します。

表 11　段落モードを利用した表（2 列目が段落モード。幅は横幅の 40% とした）

表 7 では、1 つの表だけを置いたが、表が小さければ次にようにすることも可能です。tabular の列の左右寄せパラメータを指定する引数内で p{} を使って幅指定した段落モードにした上で（真ん中の段落は&&で空の列要素としている)、それぞれの列要素として tabular 環境で作成した 1 行の表（tabular の並び要素が tabular）としています。それぞれの段落内で\centering を使って中央揃えとしてます。

1	2	3
左側	中央部	右側
Apple	Pine	Orange

表 12　左側の表

1	2	3
左側	中央部	右側
Apple	Pine	Orange

表 13　右側の表

最後に、\multicolumn と\cline を使った例を挙げておきましょう。

シュークリームの材料		
品名	分量	用途
バター	100g	シュー生地
塩	2g	
小麦粉	100g	
卵	3 個	
卵黄身	4 個分	カスタードクリーム
砂糖	100g	
小麦粉	50g	
ミルク	500cc	
バニラエッセンス	少々	
洋酒	少々	

表 14　シュークリームのレシピ

7　LaTeX での文書作成

LaTeX には印刷レイアウトを美しく制御できるばかりでなく、論理構成を明確にした文章の作成を支援するシステムとしても非常に優れています。LaTeX では文書構造を定める論理構成要素を指定することができ、文書作成の骨格を明瞭に整えることができます。

ワードプロセッサは、モニタに表示されているものが最終的に出力されるものと同じであるべきだという考え方に基づいた方式を採用しています。この方式を “What You See Is What You Get” の頭文字を取って **WYSIWYG** 方式といいます。WYSIWYG 方式は一般利用者にとって使いやすいのですが、文書レイアウトと文書構成要素とを視覚的に関連させているため、特定のワードプロセッサで作成された文書（それらは単純なテキストファイルではありません）は別のシステムでの再利用を非常に困難にしています。

7.1　文書構造

LaTeX では文章の論理構造を明確に組み立てながら文書構造を明瞭にした文書を書くことが可能です。あるテーマについて LaTeX システムを利用して文書や論文を作成する場合、次のような過程を経て文書を作成することができるのです。

1. 文書の全体を構想し、文書構造コマンドを使ってまず文の骨組み (目次構成）を指定する（36 ページ）。
2. それぞれの文構造に応じて文章を書き足す。
3. 再び、文書全体の構成とバランスを考慮しながら、文章や文構成の入れ替え・削除などの編集を行なう。
4. 文章表現やレイアウトなどを調整して文書として完成する。
5. 目次を作成（36 ページ）し、必要に応じて索引をつける（54 ページ）。

一つの文書は論理的階層としていくつかの部分に分割でき、LaTeX では次のように考えています。文書は、図 3 のように、部 (part) が最上位の文構造単位であり、それから下位に向かって章 (chapter)、節 (section)、さらに項 (subsection)、そして目 (subsubsection) の論理単位からなっていると考えます。また、さらに細かく段 (paragraph) と小段 (subparagraph) という単位もあります。

LaTeX システムでは、図 3 のような文書の論理構造をラベルして、その見出しが指定できる文構造コマンドを用意しています。これらの文構造コマンドを使うと、コンパイルして DVI ファイル作成の際に、文構造で指定した見出しにその論理構成レベルに応じて自動的に通し番号を付けてくれます。この見出し番号は、文構造タグの階層構造（図 3）を反映して、ピリオドで区切られた数字の並びとして表示されます。編集時に文章構造を入れ換えて文構造タグの順番が変更されたときでも、見出し番号の付替えは LaTeX システムが自動的に行ないます。文書作成者は文章の構成だけに専念しながら文を組み立てればいいのです。

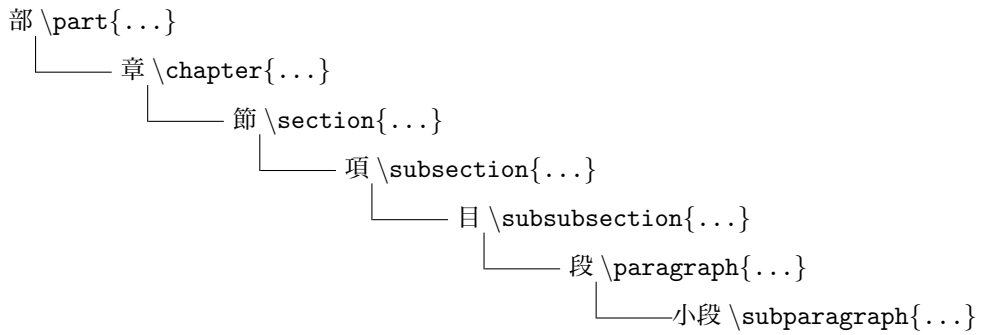

図 3　文章の文構造とその見出しを生成する LaTeX の文構造コマンド

これらの章や節などの文構造を指定する文構造コマンドには '*' を付けることができ、次のように使うこともできます。

```
\part*{...}          \chapter*{...}        \section*{...}
\subsection*{...}    \subsubsection*{...}  \paragraph*{...}
\subparagraph*{...}
```

こうした*付きコマンドを使って書かれた文書では出力時には見出し番号が付きません。また、これらの見出しは目次にも現れません。

LaTeX では、文構造を指定するコマンド以外にも表や図であることを特定する環境も用意しています（12.3 節）。

7.2　LaTeX の文書クラス

いままでの例にあったように LaTeX ファイルの冒頭部分

`\documentclass{jarticle}`あるいは`\documentclass{jsarticle}`

は文書クラス jarticle あるいは jsarticle を指定しています。図 4 にある 2 つの文書 A と B はいずれも表題に続いて、セクションが 1 つ以上続き、それぞれのセクションは 0 個以上のサブセクションに別れていて、また各サブセクションは 0 個以上のサブサブセクションが続いています。表題に続く要約はオプションです。このよう場合には 2 つの文書は同じ文書クラスに従っている、つまりオブジェクト指向の言葉でいえば 2 つの文書は同一クラスのインスタンスだと考えるのです（LaTeX では jarticle クラスに相当してます）。

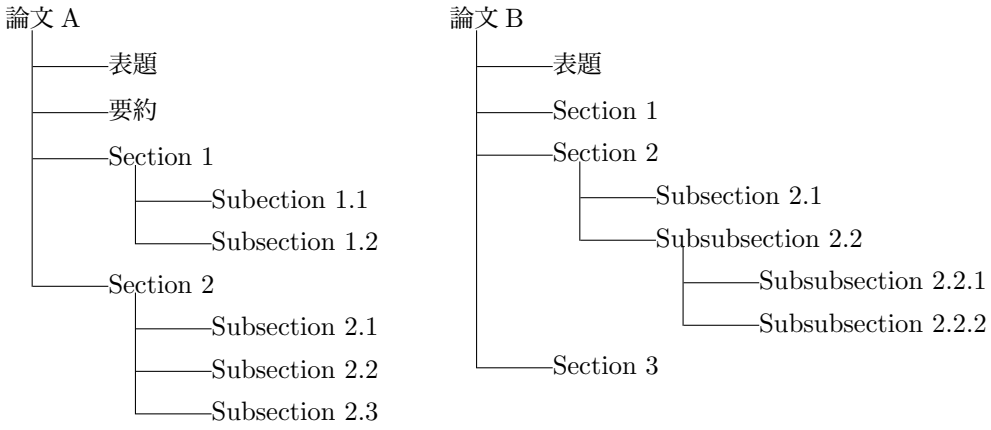

図 4　論文 A と論文 B は同じ文書クラスに属している

LaTeX 文書を作成する場合、文書内容から判断して、節や項、目となるべき箇所に文構造コマンド `\section{..}`、`\subsection{..}`および`\subsubsection{..}`などを配置し、その`{...}` の部分に見出し (heading) を記入しながら文章の全体構成を行なっていきます。

文書クラス jarticle で利用できる文書構造指定は`\section{...}`以下に限られ、`part{...}`や`\chapter{..}` は利用できません。文書クラスとして `jbook.sty` や `jreport.sty` を選んだときには、`\chapter{...}`のようなさらに大きな文構造を指定することができます大きな文構造を持つ文書であっても、文書量として多くなるわけではないことに注意してください。たとえば、jsarticle を使って数百ページの論文を書くことも、jbook を使って数十ページの文書を書くこともできるのです。

LaTeX では、文書構造を特定するために文書クラスがあるだけではなく、それに応じた文書レイアウトも定めています。同じ文書構造をもっていても`\documentclass{.....}`に指定する文書クラスを変えるとレイアウトを変更することができます。たとえば、横書き日本語論文クラス jarticle の替わりに、tarticle を指定すれば縦書き文書となります。tarticle、treport、tbook はいずれも、それぞれ横書き文書クラス jarticle、jreport、jbook の縦書きに対応した文書クラスです。こうした文書クラスは（TeX に関する十分な知識があれば)）著者の好みに応じて自由にカスタマイズすることが可能です（奥村 [4])。

文書クラスとは別に書式や体裁を便利に整えるために非常に多くのマクロパッケージが公開されており、これらを利用して LaTeX の表現力をさらに高めることも可能です。学会誌に投稿する研究論文のために、その雑誌のために指定された文書スタイルを使うように要請されることがあります。また、出版社では独自の文書スタイルファイルを用意して出来上がりの文書のあり方を定めています。

LaTeX システムの利用者は、一度文書クラスを決めてしまえば、活字の大きさやバランスなど “仕上がり” に関する詳細を一切気にすることなく、文書内容そのものの作成・編集だけに集中すればよいのです。

7.3　文書作成の実際

以下、文書クラスとして jarticle を指定した場合を例に、具体的な LaTeX を使う文書作成の概要を説明します。

7.3.1 文書の表題

文書には必ず表題を付けるようにします。4.1 節（11）で紹介したように、LaTeX では表題 (title) 出力のための 4 つのコマンド\title{..}、\author{..}、\date{..} および \maketitle が 1 組になって表題要素を構成しています。\maketitle を除いて、これらの表題要素を次のように内容指定してプリアンブルに記述します。

\title{..}	表題を指定
\author{..}	著者を指定
\date{..}	日付を指定
\maketitle	表題を出力（必須）

文書には表題が必要です。表題要素を指定すると表題としての文書情報がマークされることになり、文書情報処理の際に重宝します。仮に、表題を出力しない場合には\maketitle の行をコメントアウト（行頭に % を書く）にすればいいだけです。ファイルの冒頭に作成した文書に関する簡単なメモと日付などをコメントしておくことも大切です。

7.3.2 目次の出力

LaTeX では、文の論理構造を指定する文構造コマンドの情報に基づいて見出しが登場するページ数を対応させて、自動的に目次 (contents) を作成することができます。目次を作成するための LaTeX コマンドは \tableofcontents です。

目次を出力するときは、\begin{document}と\end{document} ではさまれた範囲で、目次を出力したい場所に 1 行

```
\tableofcontents
```

と書くだけです。この目次コマンドは、表題部の直後の本文に先立つ場所に記入するのが普通です。

目次情報の入った文書出力のための DVI ファイルを作成するためには、ファイルを最低 **2** 回コンパイルしなければなりません。最初のコンパイルでシステムは文書ファイルから文構造タグにある見出し情報を取り出し、これを目次情報として拡張子 '.toc' が付いた目次ファイル（Table Of Contents ファイル）を作成します。この段階で作成される DVI ファイルには目次情報は含まれていません。ファイルをもう 1 度コンパイルすると、この目次ファイルを読み込んで目次情報も取り込んだ DVI ファイルが作成されます。これが目次を出力するためにファイルを最低 2 回コンパイルしなければならない理由です。文章構造を変更した場合も同じく 2 回以上のコンパイルが必要です。

7.3.3 LaTeX ファイルのスケルトン

以上をまとめて、LaTeX 文書を作成する際のファイルの一般的様子を以下に示します。

```
% 2nd Sep. 2010 作成                    『%』から行末までの文はコメントとなる
% 12  Sep. 2010 修正                    作成した日付を入れる
% コンパイル時の注意や、文書の目的なども記載しておくとよい
\documentclass{jarticle}% jsarticle の利用を推奨          文書クラスの指定
```

```
% プリアンブル部                          本文中で使うコマンドなどの約束事などを記す
\usepackage{graphicx}                      利用するパッケージを指定
\usepackage{amsmath,amssymb}
\title{タイトル}                          \title{}, \author{}, \date{}, \maketitle
\author{著者名}                           で 1 組と考える
\date{日付}
....
% プリアンブルはここまで
\begin{document}                           これ以下が本文の始まり
\maketitle                                 表題の出力
\tableofcontents                           目次の作成。2 回以上 jlatex をかける

\begin{absrtact}
        論文の要約が必要なら、\begin{abstract}と\end{abstract}で
        囲まれたこの部分に書く。やや小さな字で印刷される。
\end{abstract}

\section{節の題名}
        節の内容を書きます。いくら長くても短くても構わない。
        次に \section{..} タグが現れるまでの文がこの節に属する。

\subsection{項の題名}
        必要なら副節を付けることもできる。
\subsection{..}は直前の\section の下位に属する。

\subsubsection{目の題名}
        \subsubsection{..}は直前の\subsection の下位に属する。
%\subsubsection{目の題名}                  文書構造の推敲跡をコメントで残す

\section{節の題名}
        ここから新たな節が始まります。節に付けられる番号は
        \section{..}が登場してくる順番にふられる。
        \subsection{..}や\subsubsection{...}についても同じ。
         .........

\section{節の題名}
        以下、同様に文を書いていく。
        .........
\end{document}                             ここで本文が終わる
```

このように、LaTeX では文書としての文構造コマンドによって該当する見出し部分マークすることで最高度に美しい整形出力を実現するシステムです。誰にでも読みやすくわかりやすい文章を書くための基本は、

- 文章の意味内容や文構造をよく考え
- 文構造コマンド\section{..}, \subsection{..}や\subsubsection{..}などをうまく活用し、
- 適切な見出しつけて

文章を作成し、

- 文書にふさわしいタイトルを考え
- 文書の構造を一望できる目次をつけて

文書を完成することです。この文書作成支援システムとしての LaTeX の特性を利用しながら文書を書くことは、的確な表現力を身に付けるための有効な訓練の一つとなるでしょう。

7.4 ファイルの分割

LaTeX で作成する文書が長くなってくると、編集作業に支障をきたしたり、またエラーを取り除くためのコンパイル処理時間も長くなってしまいます。このようなときにはファイルを分割し、基本となるファイルに分割したファイルを読み込んで、文書の構造化を促進しながら編集作業を容易にすることができます。基本となるファイルをルートファイルまたはマスターファイルといいます。とくに書籍や長大な論文レポートの作成などには、効率的な LaTeX ファイルの処理のためには、ファイル分割とそれにかかわるファイル管理が鍵となります。

ファイルを分割するときにはコマンド\input{..}か\include{..}を使います。ここでは\input{分割 TeX ファイル}を使う場合だけを説明します。以下の例のように\input コマンドは{ }内にファイルを入れる必要はなく、“\input␣ファイル名” でもよい（ただし、半角の空白 ‘␣’ を空けること）。

たとえば、次のようなルートファイル root.tex を作成します。

```
\documentclass{jarticle}
\begin{document}
\tableofcontents
%\input introduction.tex
\input development.tex
%\input turn.tex
%\input{conclusion.tex}
%\input{../appendix.tex}
%\input book/biblio.tex
\end{document}
```

このファイルをコンパイルするとき\input で指定したファイルが読み込まれるのです。読み込まれるファイルの中でもさらに\input{...}が使われていても構いません。しかし、\input コマンドをこうしてネストさ

せるとファイル管理が複雑になりますから注意してください。

この例では、\inpute の前にコメント記号 % が付いている行を含んでいるので、読み込まれるファイルは development.tex だけとなります。このようにファイルを分割し、さらにコメント機能を使って、指定したファイルだけを処理の対象としてエラーを取り除くことができます。

\inpute{../appendix.tex} や \input{book/biblio.tex} のように、読み込まれるファイルの指定はルートファイルからの相対ディレクトリパスによって指定します。ディレクトリパスの区切りは、Linux や Windows や Macintosh であっても、記号 / を使います。

8　パッケージの利用

4.3 節でわずかに取り上げた LaTeX で標準に定義されている文書クラスで利用できる環境だけでは、きめ細かい文書レイアウトを行うことが難しい場合があります。このような場合には、別に用意されたマクロパッケージを次のようにプリアンブル部に \usepackage{..} コマンドを使って読み込みます。パッケージに関する詳しい情報は TeXWiki を参照してください。画像ファイルの張り込みのために利用した graphicx パッケージについてはすでに 5.3 節（26 ページ）で説明しました。

```
\documentclass[クラスオプション]{文書クラス}
\usepackage{パッケージ名}
....
\usepackage{パッケージ名}
....
\begin{document}
....
```

8.1　段組文書

段組文書とするには、2 つの方法があります。1 つ目は、文書スタイルでオプション twocolumn を指定して 2 段組とする方法です。

```
\documentclass[twocolumn]{jarticle}
```

この場合には文章全体が 2 段組で出力されます。

パッケージ multicol を使うと、文中の任意の場所を多段組文書とすることができます。たとえば、指定した範囲にある文書を 2 段組にするには、multicol パッケージを読み込んで次のように書きます。multicols 環境に渡すパラメータが多段数を指定します。

```
\documentclass{jarticle}% jsarticle の利用を推奨
\usepackage{multicol}
```

```
...
\begin{document}
.......
文章を書く。ここに書かれた文章は 1 段組で出力される。
.......
\begin{multicols}{2} % 2 段組を指定
........
    ここに文章を書くと、2 段組で出力される。
    段数は \begin{multicol}{カラム数} で指定する
........
\end{multicols}
ここ以降に書かれた文章は 1 段組で出力される。
.......
\end{document}
```

多段組文書は、乱用すると印刷バランスが崩れてしまい読みにくくなってしまいます。日本語では、本文のフォントサイズを小さくしない限り多段組は 2 か 3 段が見た目にも読みやすさでも限界でしょう。段組の必要性と効果をよく考えてから利用してください。

8.2 パッケージの入手とインストール

パッケージ利用は LaTeX の世界を大きく広げます。標準的な TeX システムでは既に主なパッケージファイル群がインストールされています。TeX システムがコンピュータのどの場所にインストールされるかは TeX インストーラに依存します。Windows の「TeX インストーラ 3」では `C:\w32tex/share/`以下（津田では `C:\usr/local/share/`以下）にあるフォルダ `texmf/`内の `tex` または `ptex` 以下にインストールされています。『LaTeX コンパニオン』[2] には、代表的なパッケージの利用法が詳しく紹介されています。「LaTeX でこんなことができたなら」と思った場合に参考になるでしょう。日本語 TeXWiki でもさまざまなマクロパッケージのインストール法や使い方が集められていますので参考にしてください。

パッケージを含む TeX 関係のソフトは CTAN(Comprehensive TeXArchive Network) とと呼ばれるサイト群に集められています[*10]。パッケージ名が分かってかっている場合には ‘tex␣パッケージ名’ などで検索してパッケージをダウンロードします。

パッケージを定義している本体のファイルはスタイルファイルとも呼ばれ、拡張子 ‘`.sty`’ または ‘`.cls`’ が付きます。スタイルファイルはパッケージを必要とする TeX ファイルと同じフォルダに置けばよいのですが、それでは不便なので、以下で説明するしかるべき場所に配置（インストール）し、コマンド `mktexlsr` でその場所を TeX システムに記録しておくと、任意の場所にある TeX ファイルからパッケージを呼び込むことがで

[*10] 入手したいパッケージは
TeXUser Group `http://www.tug.org/`
から、または日本のミラーサイトとして
RING サーバ `http://www.ring.gr.jp/pub/text/CTAN/`
から入手することができます。

きます。

ダウンロードする際には、スタイルファイル (.sty,.cls) だけが配布される場合や関連ファイル一式として配布される場合があります。それらの関係は次のようになっています：

.sty, .cls　パッケージの本体ファイル。
.dtx　パッケージ本体と説明文書をパックしたファイルで、ins ファイルが必要。
.ins　dtx ファイルから、スタイルファイル本体や説明文書をアンパックするためのファイル。dtx ファイルとセットでダウンロードする。

パッケージとしてパックされたファイルたとえば、package.dtx の場合、一緒にダウンロードした package.ins が同じフォルダにあることを確認したうえで

```
% platex package.ins
```

を実行して、アンパックします。これによって、(場合によっては複数の) sty/cls ファイルや説明文などが生成されたことを確認します。説明文書が含まれている場合には dtx ファイルから

```
% platex package.dtx
```

によって dvi ファイルが生成されます（何度か platex を適用する必要があるかもしれません)。

こうして得られたパッケージファイル群を TEX システムにインストールするには、まず TEX システムがその場所を探索可能な場所に置きます。通常はそのフォルダごと、たとえば texmf/tex/misc や texmf/ptex/misc に移動します（texmf/tex 内に自由にフォルダを作成して構いません)。そして、これが重要なことですが、その後にかならず コマンド

```
% texhash
```

を実行します（以前は mktexlsr でしたが最近は texhash)。これによって、TeX システムが探索可能なパッケージファイル群などを記述したファイル ls-R が生成されます。こうすることによって、TEX 文書がどこにあってもプリアンブル部に\usepackage{..}と書いてパッケージを読み込む際に、同じフォルダにスタイルファイル sty/cls を置かなくて済むのです（8.3 節参照)。

8.3　TEXMFHOME の利用

TeX システムのインストール時には、環境変数 TEXMFHOME が設定されます。TEXMFHOME とは、TeX ユーザ個人が自由に使うことのできるフォルダ texmf を置くことができる場所で、定められたフォルダ構成にしたがって、TeX システムが有する標準的なパッケージ以外のパッケージを CTAN(The Comprehensive TEX Archive Network) [*11] などから入手したパッケージを置いておくことができます。

もし TEXMFHOME を利用しないとすると、TeX の本文ファイルに必要な標準以外のパッケージ（つまり、スタイルファイル）をその TeX ファイルと同じフォルダに置いておく必要があります。新たな TeX ファイルを作成するために必要なパッケージファイルを同じフォルダに置くことになって（同じパッケージファイルを重複して配置することになるでしょう)、ファイル管理上、無用の混乱を来してしまいます。TeX ファイ

[*11] http://www.ctan.org

ルに必要な標準以外パッケージは、以下で説明する方法にしたがってフォルダ texmf 内のサブフォルダ tex 以下（つまり、texmf/tex 以下）に配置しましょう（必要なパッケージが現在の TeX システムに存在しているのかどうかを調べる方法も以下で説明します)。

TeX システムでは、TeX システムにとってたいへん重要なさまざまな情報，特に TeX ユーザにとって必要なさまざまなパッケージ（スタイルファイル）が収められている場所が複数あり、それらは **TEXMF** ツリーと呼ばれています。これらのコンピュータ内の場所は、TEXMF で始まる環境変数で指定されています。

8.3.1 TEXMFHOME の場所

TEXMFHOME の場所は、コンピュータの OS や TeX システムの配布形態によって異なります。大抵の TeX インストーラでは、ユーザのホーム領域が TEXMFHOME の場所に割り当てられていて[*12] 、そこに次に従ってフォルダ tt texmf を作成します。

Windows

- Windows XP の場合: C:\Documents and Settings\ユーザー名
- Windws Vista の場合: C:\Users\ユーザー名
- Windows 7/8 の場合: C:\Users\ユーザー名

Mac

- Macport で TeX システムを構築した場合、TEXMFHOME は ~/（ユーザホーム）に設定される。
- MacTeX によって TeX システムをインストールした場合には、TEXNFHOME は ~/Library に設定される。ただし、MacOS 7(Lion) から フォルダ ~/Library は不可視になっています。MacOS 10.8 Mountain Lion のユーザライブラリを可視化する方法などを参考に見えるようにしてから、Desktop でフォルダ TEXMFHOME/texmf を作成してから、 ~/Library にドラッグするようにすると間違えないでしょう。

8.3.2 TEXMFHOME の使い方

上で説明したそれぞれの TeX システムに応じた TEXMFHOME に texmf という名前のフォルダを作成します（これを以降で TEXMFHOME/texmf と表記します)。ただし、フォルダ TEXMFHOME/texmf は次のような構造をになっていなければなりません。texmf 内にサブフォルダ tex を、そのサブフォルダ tex フォルダがなければならず、必要ならその中に platex や latex など任意のフォルダを置くことができます)。

```
TEXHOME（←インストール環境に応じて場所が決まってます）
    texmf
       |--tex
        |-- 直接パッケージを置いてもよい
            |--platex <-- 日本語に関わるパッケージ（フォルダごとでもよい）
            |--latex  <-- 一般のパッケージパッケージ（フォルダごとでもよい）
            |--misc   <-- 何か他のパッケージ
```

TEXMFHOME/texmf/の中には標準以外の TeX パッケージ（スタイルファイル）を置きます（必要なフォルダごと置いて構いません)。。

[*12] TeX インストーラ 3 http://www.math.sci.hokudai.ac.jp/ abenori/soft/abtexinst.html もそうです。

すると、TeX システムは TEXMFHOME/texmf/tex/ を含む TEXMF ツリー内を検索して、それが存在すれば、TeX ファイルで \usepackage{...} で宣言したパッケージとして読み込んでくれます。とても便利！

> 大学のシステムのように，ユーザがホーム以外に自由にパッケージを追加できない場合、TEXHOME に texmf フォルダを作成しておくことはとても大切です。そのとき、追加するパッケージは TEXHOM/texmf/tex に置かねばならないことを再度強調しておきます（TEXMFHOME/texmf/ 内にパッケージを置くと探すことができません)。

8.3.3 TeX システム内のパッケージを探す

必要なパッケージ名がわかっているときに、現在の TeX システムにそのパッケージが存在しているかどうかを知るには、コマンド kpsewhich を次のように使います（下の記号「%」 はコマンドプロンプトであり、入力する必要はありません)。もしパッケージが存在しない場合には、インターネット経由でパッケージを入手して、自分の TEXHOM/texmf/tex/ 内に置きます。次の例は、MacTEX の場合に、パッケージ（スタイルファイル）fourier.sty や pxjahyper.sty が TeX システムの TEXMF ツリー内にあるかを調べた例です。

```
% kpsewhich fourier.sty<
/usr/local/texlive/2012/texmf-dist/tex/latex/fourier/fourier.sty

% kpsewhich pxjahyper.sty
/Users/masahiro/Library/texmf/tex/misc/PXjahyper-master/pxjahyper.sty
```

この例では、それらのスタイルファイルは存在して、そのファイルの場所は それぞれ/usr/local/texlive/2012/texmf-dist/ および /Users/masahiro/Library/texmf/tex/misc/PXjahyper-master/pxjahyper.sty にあると表示されました。

TeX システムがどの TEXMF ツリーを検索するかを確認するには、この kpsewhich コマンドを次のように使います。

```
% kpsewhich -var-value TEXMF
{C:/Documents and Settings/masahiro/texmf</font>,C:/w32tex/share/texmf-projects,
C:/w32tex/share/texmf-local,C:/w32tex/share/texmf} <- Windows XP の場合

{/Users/masahiro/Library/texlive/2012/texmf-config,/Users/masahiro/Library/texlive/2012/texmf-var,
/Users/masahiro/Library/texmf,!!/usr/local/texlive/2012/texmf-config,
!!/usr/local/texlive/2012/texmf-var,!!/usr/local/texlive/2012/texmf,
!!/usr/local/texlive/2012/../texmf-local,
!!/usr/local/texlive/2012/texmf-dist} <- MacTeX の場合
```

自分で作成した TEXMFHOME/texmf が検索対象のフォルダになっていることをまず確かめてください。繰り返しますが，パッケージは上で説明したように TEXMFHOME/texmf/tex/ 等の中に置いておかねばなりません (パッケージを含むフォルダ全部でもよい)。

9　スライドおよびポスターの作成

TEX システムを使ってプレゼンテーションのためのスライドを作成する数多くのパッケージが提案されてきました。TEX システムで作成するスライドは今日では PDF ファイルとして作成し、Adobe Reader でフルスクリーンとして表示してプロジェクタを通してプレゼンテーションをおこないます。PDF ファイルとしてスライド（やそのハンドアウト）を作成すると、特定の OS や有料ソフトウエアに依存せずに配布ができるという利点があります。

ここではパッケージ Beamer [*13] をつかったスライドとポスターの作成を簡単に紹介します。

9.1　Beamer でスライド

Beamer パッケージを使ったスライドは、現在 LATEX で作成するスライドの事実上の標準となっており、多くのデザインテーマの提供（それらを改造して自分専用のテーマ作成も可能です）やページリンク機能およびアニメーションなど、専用のプレゼンテーションソフトウエアに匹敵するインタラクティブなスライドを作成することもできるようになっています。

以下のソースは `platex` でコンパイルし、DVI ファイルを `dvipdfmx` によって PDF ファイルを生成してスライドを作成する Beamer ソース例です（スペース節約のために 2 段組としました）。

一行目の `\documentclass[dvipdfm]{beamer}` で beamer パッケージを読み込んでいますが、オプションで dvipdfm を指定していることに注意して下さい。英語のみの LATEX 文書の場合、海外では pdflatex を使って直接 PDF ファイルを生成するのが標準的なのですが、pdflatex はまだ日本語に対応していません。したがって、日本語 LATEX 文書では platex + dvipdfmx を使わざるを得ず、beamer ではこのオプション `[dvipdfm]` を指定する必要があります[*14]。

また、2,3 行目で `\usepackage{{hyperref,PXjahyper}` でパッケージ hyperref と pxjahyper を読み込んでいます。これで、hyperref + dvipdfmx の組み合わせで日本語を含む「しおり」をもつリンクが埋め込まれた PDF スライドを作成することができます。Beamer テーマとして、この例では Madrid を使っていますが、多くのテーマが標準で用意されています。いろいろ試みてください。

Beamer で作成するスライドでは、スライド 1 枚分を次のように frame 環境内で記述します。

```
\begin{frame}{スライドタイトル}
...
ここに 1 枚分を書く
...
\verb+\end{frame}
```

ここで frame 環境び続いて `{スライドタイトル}` は、各スライドの上部にスライドタイトルを表示するためです（`{スライドタイトル}`自体を省略するとページにはスライドタイトルが表示されません）。また、frame 環境の外側では、`\section{..}` や `\subsection{..}` も通常の LATEX ファイルのように使うことができ、

[*13] `The beamer package http://www.ctan.org/tex-archive/macros/latex/contrib/beamer/.`

[*14] LATEX サンプルが上手くコンパイルできない場合にも、documentclass でこの `[dvipdfm]` オプションを指定してみて下さい。

\tableofcontents によって目次スライドを作成することができます（下の例でも使っています。参考にして下さい)。

beamer スライドソース例

```
\documentclass[dvipdfm]{beamer}
￥usepackage[dvipdfmx]{hyperref}%hyper リンク
\usepackage{PXjahyper}% 日本語しおり
\usetheme{Madrid}% 他のテーマも試してみよう

\usepackage[english]{babel}%for English
\usepackage{amsmath,amssymb}%AMS 記号用
\usepackage{mathptmx}%math 用 Adobe Times Roman
\usepackage{helvet}%for normal english
\usepackage{courier}%\texttt{..}で courier
\usepackage[T1]{fontenc}% おまじない (1)
\usepackage{lmodern}% おまじない (2)
\usepackage{graphicx}% 各種画像用
% 箇条書きを段階的にみせたいとき
%\beamerdefaultoverlayspecification{<+->}

\title[Makin Slides using Beamer]
{\LaTeX{}+Beamer でスライド作成}
\subtitle{\LaTeX{}によるプレゼンテーション}

\author[Taro Meiji]{明治太郎}
\institute{明治大学理工学研究科}
\date[June 8 2013]{2013 年 6 月 8 日}
\subject{\LaTeX{}+Beamer}

\begin{document}
\begin{frame}
\titlepage
\end{frame}

\begin{frame}<beamer>
\frametitle{Agenda}
\tableofcontents
\end{frame}

\section{はじめに}
\begin{frame}{何を問題としているか}
\begin{itemize}
\item こんなこと
\item あんなこと
\item しかも\alert{そんなことまで}
\end{itemize}
\end{frame}

\section{手始めに}
\subsection{Block の使い方}
\begin{frame}{さまざまな Block}
\begin{block}{ブロック}
これが block 環境だ。
\end{block}

\begin{example}
これは example block である。
\end{example}

 \begin{alertblock}{警告ブロック}
 alert block 環境ではこうなる。
\end{alertblock}
\end{frame}

\subsection{式を表示する}
\begin{frame}{\LaTeX{}だから数式は得意だ}
Pauli 行列の性質は次のようだ。
\begin{equation}
[\sigma_x, \sigma_y]=2i\sigma_z,
[\sigma_y, \sigma_z]=2i\sigma_x,
[\sigma_z, \sigma_x]=2i\sigma_y
\end{equation}

ブロック環境でも数式を書ける
\vspace{0.5cm}

\begin{block}<+->{内積の定義}
関数 $\phi(x)$と$\psi(x)$の内積
\begin{equation}
\langle\phi, \psi\rangle =
 \int \phi^*(x) \psi(x) dx
\end{equation}
\end{block}
\end{frame}

\section{図表の貼り込み}

\subsection{図}

\begin{frame}{PNG 画像}
PNG 形式の画像も、
bb ファイルを用意しておけば、この通り。
\begin{figure}
```

```
\includegraphics[scale=0.3]
   {image/lorenz_flow.png}
\end{figure}
\end{frame}

\subsection{作表}
\begin{frame}{\LaTeX{}で作表してみる}
自分で\LaTeX{}コードで作表するのはチトきつい。
\begin{table}[htb]
\begin{center}
\begin{tabular}{l|r|l}
\hline
\multicolumn{3}{c}{シュークリームの材料}\\
\hline
\multicolumn{1}{c|}{品名} & 分量 & 用途\\
\hline
バター & 100g & シュー生地 \\
\cline{1-2}
塩 & 2g & \\
\cline{1-2}
小麦粉 & 100g & \\
\cline{1-2}
卵 & 3 個 & \\
\hline
卵黄身 & 4 個分 & カスタードクリーム\\
\cline{1-2}
砂糖 & 100g & \\
\cline{1-2}
小麦粉 & 50g & \\
\cline{1-2}
ミルク & 500cc & \\
\cline{1-2}
バニラエッセンス & 少々 & \\
\cline{1-2}
洋酒 & 少々 & \\
\hline
\end{tabular}
\end{center}
\caption{シュークリームのレシピ}
\label{tbl-cream}
\end{table}
\end{frame}

\section{結語}
\begin{frame}
\frametitle{わかったこと}
\begin{enumerate}
\item \LaTeX{}はとても便利
\item Beamer は Cool
\begin{itemize}
\item \LaTeX{}と beamer だけで
   プレゼンテーションが可能
\item Nothing else?
\end{itemize}
\end{enumerate}
\vspace*{1.5cm}
\onslide
質問などは \href{mailto:hogehoge@meiji.ac.jp}
{\texttt{hogehoge@meiji.ac.jp}}にどうぞ
\end{frame}
\end{document}
```

9.2 Beamer でポスター

パッケージ beamerposter [*15] は、Beamer 機能を拡張してポスターを作成するパッケージです（Beamer が使える環境が前提）。

次は Beamerposter の基本的な使い方の骨格を示しています。`\documentclass[dvipdfm]{beamer}` とするのは通常の Beamer ファイルと同じです。次の行で、Beamerposter パッケージを用紙の使い方や A 版用紙サイズなどを指定して `\usepackage[orientation=portrait, size=a0, scale=1.4]{beamerposter}` のように読み込みます。用紙を縦置きにする場合は `orientation=portrait`、用紙を横置きにする場合には `landscape` とします。用紙サイズは `size=a0` として A4 から A0 まで選ぶことができます。

`\begin{document}` から始まる本文では、1 つだけの frame 環境が使われることに注意して下さい（Beamer では frame 環境ごとにページが生成されたことを思い起こして下さい。ポスターは 1 枚だけです）。この ‘大

[*15] The beamerposter package `http://www.ctan.org/tex-archive/macros/latex/contrib/beamerposter`.

きな’ 1 つだけの frame 環境内に、block 環境（alertblock も使えます）で記述したい事柄をブロック見出しを明記しながら記述するのです。

```
\documentclass[dvipdfm]{beamer}
\usepackage[orientation=portrait, size=a0, scale=1.4]{beamerposter}
\usetheme{使用するテーマ}
...
必要なパッケージ
...
\begin{document}
\begin{frame}
\begin{block}{見出し 1}
...Beamer スタイルで書く
\end{block}
\vfill
\begin{frame}
\begin{block}{見出し 2}
...Beamer スタイルで書く
\end{block}
....
....
\end{frame}
\end{document}
```

The beamerposter package に付属する `example.tex` を見てみると、用紙の向きに応じて段組を使って block 環境を使っています（block 環境は指定した幅を使い切りますから、用紙幅が大きければブロックは横に広がって縦に薄くなってしまい読みづらくなりますね)。次は、block 環境の幅をポスター幅（`\linewidth`）の 0.48 倍として、ポスターの記述を 2 段組としている例です。

ポスター幅を 2 段で使う例

```
\begin{columns}[t]
\begin{column}{.48\linewidth}
\begin{block}{見出し}
...
\end{block}
......
block 環境を適当回数繰り返す
......
\end{column}
\begin{column}{.48\linewidth}
\begin{block}{見出し}
```

```
...
\end{block}
......
block 環境を適当回数繰り返す
......
\end{column}
```

block 環境が出現するたびに、左端段の上から下へ、右隣の段の上から下へと移動して表示されます。もちろん、1 枚のポスター内で 1 段組、2 段組、3 段組と組み合わせて利用することが可能です。The beamerposter package に付属するポスター例では、縦置きポスターで「1 段 +1 段 +2 段」で block を表示しています。一方、RicePoster パッケージ（Rice 大学用に Beamer Poster パッケージをカスタマイズ)[*16] の例 `RicePosterExample.tex` では横置きパスターを 3 段組にし、真ん中の段を途中でさらに 2 段に分けています。

10 縦組文書

日本語の縦組用の文書スタイルには tarticle と tbook の 2 つがある。次は縦組論文の場合で、通常の論文スタイルと`\documentclass{tarticle}`の一行だけが異なっている。

```
￥documentclass{tarticle}
... プリアンブル部
￥title{題名}
￥author{著者名}
￥date{日付}
￥begin{document}
￥maketitle

... 本文
.......
\end{document}
```

10.1 ルビをふる

漢字などにルビをするためのパッケージには ruby.sty [*17]または furikana.sty [*18]などがある。

ruby.sty

```
\usepackage{ruby} % プリアンブル部で宣言
```

[*16] LATEXresources for Rice students `http://ricebeamer.dynaman.net`.
[*17] ruby.sty `http://www.nls.ics.saitama-u.ac.jp/~tohru/ja/Exports/External/Chosho/ruby.sty`.
[*18] furikana.sty `http://homepage3.nifty.com/xymtex/fujitas2/texlatex/tategumi/furikana.sty`.

```
\ruby{熟語}{ふりがな}
```

furikana.sty

```
\usepackage{furikana} % プリアンブル部で宣言

\kana{熟語}{ふりがな}
```

10.2 脚注

横組文書における脚注\footnote{...}は、その '列' の下でなく、縦組では左端または最後ページに追い込まれて具合が悪い。この不具合を修正するパッケージに kyakuchu.sty [*19] があり、次の書式に従う。まず、マーク付きで脚注本文を kyakuchutext[マーク]{脚注文}で定義しておき、それ以降の本文の箇所で\kyakuchumark{マーク}によってマークを参照して脚注を付けるのである。

kyakuchu.sty

```
\usepackage{kyakuchu} % プリアンブル部で宣言

\kyakuchutext{脚注マーク 1}{脚注文}
\kyakuchutext{脚注マーク 2}{脚注文}
...
\kyakuchumark{脚注マーク 1}脚注を付けたい本文
...
文献\kyakuchumark{脚注マーク 2}で解説....
```

パッケージ kyakuchu を使った具体例を以下にします。\footnote{..}も使っているので、その効果を検討されたい。

```
￥documentclass[a4j]{tarticle}
￥usepackage{furikana}
￥usepackage{kyakuchu}
￥begin{document}
歌枕
\footnote{
歌枕とは、和歌に引証される地名のこと。
}として、
\kyakuchutext{A1}{福島県白河市にあった奥州街道の関所。}
\kyakuchutext{A2}{芭蕉「おくのほそ道」
萩原 恭男 校注、岩波文庫七九（一九九一）。}
\kyakuchutext{A3}{蓑笠庵 梨一「奥細道菅菰抄」(おくのほそみちすがもしょう。
文献\ref{A2}に付録として掲載）の注釈が、
```

[*19] kyakuchu.sty http://homepage3.nifty.com/xymtex/fujitas2/texlatex/tategumi/kyakuchu.sty.

```
典拠を明らかにしている。}
\kyakuchumark{A1}白河の関は古来有名である。
ここより外は\kana{陸奥}{みちのく}として、
人々の旅情をかきたてる場所であった。
松尾芭蕉\kyakuchumark{A2}「奥の細道」
の白河 (白川) の関の条には、この歌枕を読み込んだ
\kyakuchumark{A3}古歌の一節がさりげなく引用されている。
\end{document}
```

11 文献リストの活用

文書の作成で参考にした文献などは、その都度文書中にその出典を明らかにする必要があります。文書を作成する場合、参考情報を開示することは文書作成における重要なマナーの 1 つです。LaTeX では参考文献リストを作成し、その文献番号を文中で利用することができます。参考文献番号を文書中で引用するために、まず **thebibliography** 環境を使って文書内で引用する参考文献のリストを文献参照ラベル`\bibitem[オプション]{ラベル名}`を使って作成しておきます。文中で文献番号を参照する個所で `\cite{ラベル名}` と書くと、該当する参照ラベル名に LaTeX システムが置き換えます。

11.1 参考文献リストの作成

参考文献リストは thebibliography 環境を使って次のように書きます。普通は文書の最後に文献リストをおくのが普通です。

```
\begin{thebibliography}{n}
\bibitem[opt]{key1} 文献情報 1
\bibitem[opt]{key2} 文献情報 2
.....
.....
\end{thebibliography}
```

- thebibiliography 環境の引数 n には、参考文献リストにある文献数の桁数に相当する適当な数 n を入れます。1, 2, 3,... 桁に応じてそれぞれ 9, 99, 999... を書きます。
- `\bibitem` の引数 *key* には、文書中で参考文献を引用するときの引用ラベル名を指定します。ラベル名では大小文字は区別されます。
- `\bibitem` のオプション引数 `[ ]` を指定しない場合、参照ラベル名は、`\bibitem` の登場する順番に番号 [1]、[2]、[3]、... となります。

  参照ラベル名が番号だけでも特に混乱がなければ、以下の例の`\bibitem{urashimada}`のようにオプション引数を付ける必要はありません。

- オプション引数として \bibitem[花坂]{hanasaka} のように指定すると、文献参照ラベル名がこのオプション引数になります。
  同じ著者の文献が多数あるときなどでは\bibitem[花坂 1973]{hana73}、\bibitem[花坂 1984]{hana84}などとすると便利なこともあります。

具体的には次のように書きます。

```
現代の‘さるかに合戦’研究がその父をもつとすれば、それは浦島田太郎 (1905--1989) である。
その姿は今日では氏の集大成というべき研究書\cite{urashimada}に求めることができる。
.......
浦島田太郎に学んだ花咲爺が\cite[p.53-55]{hanasaka}で感慨をこめて述懐しているように、
.......
さるかに合戦の歴史的方法の現代的位置づけは\cite{asigara}に総括されている。

\begin{thebibliography}{99}
\bibitem{urashimada} 浦島田太郎、『さるかに合戦研究序説』、猿蟹大学出版会、1985 年.
\bibitem{asigara} Kintaro Asigara, \textit{The Battle of Saru-Kani and historical
methods}, Histrical Review, \textbf{31}(1989), pp125-209.
\bibitem[花坂]{hanasaka} 花咲爺、『さるかに合戦の考古学』、石海書店、1972 年.
\end{thebibliography}
```

参考文献リストの参照が文書中に取り込まれるためには、やはり文書ファイルを最低 2 回コンパイルしなければなりません。.aux ファイルに書き出された参考文献に関する情報を読み込むためです。

11.2 文献の引用

文書中で参考文献リストにある文献を参照ラベル名を使って引用するには、引用ラベル名 *key* を指定して次のよう書きます。

```
\cite[remark]{key}
```

- オプション引数 *remark* を省略すると、文書中に引用される参照ラベル名は、上の\bibitem で指定した通りになります。
- オプション引数 *remark* を指定したときには、文書中の参照ラベル名はたとえば ‘[3, *remark*]’ のように、文献番号に加えてオプション引数の内容が追加されます。書籍の該当ページ数などを加えたい場合には便利です。

BibTeX という別のソフトウェアを使って文献データベースを作成しておき、そこから参照文献リストを取り出すことも可能です。詳しくは参考書籍 [4] などを参照してください。

12　相互参照

LaTeX は章（chapter）や節（section）あるいは数式や図表などにシステムが自動的に番号を付けることができます。

このような章や節など位置や図表のある場所に適当な参照ラベルを付けることによって “文書中の任意の場所” でその参照ラベルが定義されている章・節番号や図表番号をを参照したり、さらには該当ページ数を出力することができます。11 節ですでに見たように、参考文献リストを作成しておき、それを文献ラベルとして文書中から引用して文献番号などを自動的に出力することも可能です。これらの機構をラベルの相互参照といいます。さらに、13 で説明するように、文書内に索引情報を埋め込んでおき、索引項目をその読み方で並べ替えて登場ページ数とを一覧表示する索引作成を自動的化することもできます。

LaTeX を出版編集システムとして見たときの大きな優位性は、目次の自動生成（36 ページ）だけでなく相互参照や索引作りが容易なことがあげられます。LaTeX の持つこの機能により編集作業の画期的な効率化と文書レイアウトの精密化が同時に達成され、本格的な文書の電子出版が可能となったのです。

12.1　相互参照の方法

LaTeX システムでは、文章の中で章や節、数式、図や表の番号や、それら（や他に特に指定した特定箇所）が登場したページを相互に参照することができます。このためには、参照したい箇所に参照ラベル名をつけてマークしておき、このラベル名によって相互参照します。

参照ラベル名を定義するには、参照場所としたい文書中で次のように `\label{..}` コマンドを使います。

`\label{`参照ラベル名`}`参照ラベルに付けられた番号を出力させる。

この`\label` コマンドは印刷出力には何の影響も及ぼしません。参照ラベル名は LaTeX 文書全体で一意的でなければならず、重複してはいけません。

文書中でこの参照ラベルを参照するには次のようにします。

`\ref{`参照ラベル名`}`	参照ラベルに付けられた番号を出力
`\pageref{`参照ラベル名`}`	参照ラベルが登場したページ数を出力

相互参照結果を文書中に取り込むためには、最低 2 回の文書ファイルのコンパイルが必要になります。1 回目のコンパイルで参照番号や参照ページなどの参照情報が拡張子 `.aux` の付いたファイルに書き込まれます。次いで 2 回目のコンパイル時に `.aux` ファイルから参照情報を読み込んで文書中に取り込まれるのです。

以下に、参照ラベルをマークする場所と参照方法について具体的に説明しておきます。

12.2　章・節番号の参照例

文構造タグ`\chapter{...}`、`\section{...}`などを使って得られる章や節の見出し番号を参照するには、たとえば次のようにします。

```
\section{さるかに合戦の背景\label{background}}
\subsection{猿の対猿関係\label{relation}}
あるいは
\section{さるかに合戦の背景}
\label{background}
\subsection{猿の対猿関係}
\label{relation}
```

このように記述しておくと、文書中でこれらの参照ラベルを次のように利用できます。

```
.... さるかに合戦は決して突発的なものではなくいくつかの伏線があった。
第\ref{background}節では合戦の背景について考察する予定である。
とくに第\ref{relation}項（\pageref{relation}ページ）で言及されること
であるが、猿の日ごろからの対猿関係は合戦の引き金要因として主要な伏線
をなしたいることがわかる。
```

12.3 図表の参照

本書では図表の作成については触れませんが（奥村 [4] などを参照してください）、LaTeX で作成した図表には自動的に図および表番号がふられます。この図表番号を参照するには参照ラベルをその図表環境の内部にマークします。図や表であることを定める figure や table 環境では、図表本体以外に図表を説明するためのコマンド`\caption{...}`コマンドを使うことができます。参照ラベルのコマンドは、次の例のように図表環境の終了`\end{tabel}`または `\end{figure}`の ‘直前’ でラベルします。

```
\begin{table}[htbp]
....
\caption{バニラアイスのレシピ}
\label{icecream}
\end{table}
```

このように記述しておくと、文書中でこれらの参照ラベルを次のように利用できます。

```
節\ref{dessert}でデザートについて学びました。.....
表\ref{icecream}（\pageref{icecream}ページ）のレシピに従って自家製アイスクリームを
作ってみるのもたのしいでしょう。
```

以上、参照ラベルコマンドとその参照の仕方をまとめると表 15 のようになっています。

参照される対象	参照方法	参照ラベルの指定
章・節などの番号	\ref{ラベル名}	\label{ラベル名}
登場ページ	\pageref{ラベル名}	\label{ラベル名}
図や表	\ref{ラベル名}	\label{ラベル名}
数式番号	\ref{ラベル名}	\label{ラベル名}
文献	\cite[remark]{ラベル名}	\bibitem[opt]{ラベル名}

表 15　LaTeX における相互参照ラベルの指定とその参照方法

13　索引の作成

LaTeX では MakeIndex という索引作成のソフトウェアを利用して索引の作成を自動的に行なうことができます。索引の作成は通常の書籍編集において最も手間のかかる作業の一つです。ここでは MakeIndex を ASCII が日本語化した mendex を使った索引の作り方を説明します。

13.1　索引作成の手順

索引を作成するには LaTeX ファイルには次の記載が必要です。

- ▷ プリアンブルにパッケージ makeidx を読み込む
- ▷ 続けてプリアンブルに \makeindex を宣言
- ▷ 本文中で索引項目を\index{よみ@読み}によって指定
- ▷ 文中で索引を出力する位置にコマンド\printindex を記入

具体的には次の形式のファイルを作成します。

```
\documentclass{文書クラス}
\usepackage{makeidx}
\makeindex
....
\begin{document}
....
\index{さくいんこうもく@索引項目}
....
\printindex
\end{document}
```

文書中で索引項目を\index{..@..}によって指定してあっても、文書の出力には何の影響もありません。したがって、将来索引を必要とする可能性がある場合は当然として、文書処理においては\index{..@..}を

検索時のキーワードとして利用できるという利点もあるために、できる限り\index{..@..}を使って索引項目を選び出して書いておくとよいでしょう。

索引の自動作成を達成するためには、次の手順に従って必要な回数のコンパイル作業が必要です。

1. ファイルをコンパイル（目次の挿入するときには、ページ数がずれるので最低 2 回コンパイル）し、拡張子 .idx の付いた索引情報ファイルを作成します。
2. 次に日本語 MakeIndex ソフトウェアである mendex を使って idx ファイルを処理して項目がアルファベット順と 50 音順に並べてページ番号に対応させた拡張子.ind の付いた索引ファイルを作成します。文書ファイルが latexfile.tex の場合には、latexfile.idx ファイルがあることを確認してから、次のように makeindex または mendex コマンドを実行します (どちらのコマンドを使うか、または両方とも使えるかは利用している TEX システムに依存します)。

   % mendex(or makeindex) *latexfile.idx*

   索引ファイル latexfile.ind が作成されたことを確認します。
3. もう一度、コンパイルします。索引ファイルの ind ファイルを読み込んで、\printindex が記入された位置に索引項目とその出現ページが順に印刷されます。
4. さらにもう一度コンパイルすると目次に索引ページが載ります。目次をいれて、目次に索引ページを入れるためには最低でも 4 回のコンパイル作業が必要です。

13.2 索引項目の指定

索引項目として指定するには文中で次の \index コマンドを使います。

```
\index{索引項目の読み方@索引項目}
```

ただし、索引項目がカタカナや漢字を含む場合には上のように書きましが、半角アルファベットやひらがなだけの索引項目は index{索引項目} とだけ書きます。

索引では、最初に英文字で始まる項目が “アルファベット順” に並び、次いで和文文字で始まる項目が “50 音順” に並びます。したがって、索引作成のポイントはコマンド index{..@..}の使い方、とくにその索引項目の読み方の指定にあります。

英数字記号　半角英数字 ‘だけ’ が索引項目のときは上で注意したように、索引項目をそのままを指定します。記号などが混じる場合、たとえば\LaTeX という索引項目の読み方は ‘LaTeX’ でも ‘latex’ でも構いません。読み方については大小文字の区別はありません。

和文　カタカナや漢字が索引項目とときには、文章の ‘揺れ’ に注意します。たとえば、“コンピュータ” と “コンピューター” とは違う索引項目になります。また、同じ索引項目に異なる読みを付け場合には、索引の取り扱いは別になります。

13.3 索引作成の文書例

索引を作成する LaTeX ファイルの例を次に示します。

```
\documentclass{jarticle}% jsarticle の利用を推奨
\usepackage{makeidx}
\makeindex
....
\begin{document}
\maketitle
\tableofcontents
.....
.... さるかに合戦
\index{さるかにかっせん@さるかに合戦}
に関する多種多彩な側面を深く検討することによって、従来の民話
\index{みんわ@民話}
的歴史認識
\index{れきしにんしき@歴史認識}
から得られないあららしい研究方法の獲得を説明することができる。
例えば、さるが盗んだとされている餅
\index{もち@餅}
をめぐる考察から当時の農耕社会
\index{のうこうしゃかい@農耕社会}
システムが把握されるのである。
社会問題
\index{しゃかいもんだい@社会問題}
との関連性を説明する餅の領域に‘合戦’の影響が見られるという認識は
.........

\printindex
\end{document}
```

この例では、あえて索引項目を指定する index{..@..}を行頭に置いています。索引項目の確認や将来の文書処理の容易さのために、文書の実際の出力には無関係なこれらの記載はできるだけわかりやすく記述しておくためです。この文書ファイルに対して、以上の索引作成の手続きを経ると文書ファイルの最後に索引ページが出力されます。

参考文献

[1] Knuth, Donald E.,『改訂新版 TeX ブック—コンピュータによる組版システム』, アスキー出版局 (1992 年）.

[2] アスキー編集部監訳,『The LaTeX コンパニオン』, アスキー出版局 (1998 年).

[3] アスキー編集部監訳,『The LaTeX グラフィックスコンパニオン』, アスキー出版局 (2001 年）.

[4] 奥村晴彦,『[改訂版第 5 版]LaTeX2$_\varepsilon$–美文書作成入門』, 技術評論社 (2010 年）.

索引