好き好き LaTeX 2ε 初級編

渡辺徹

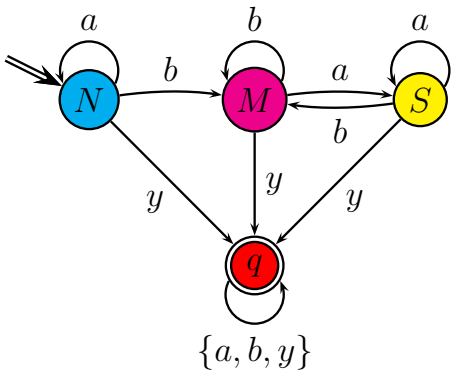
第 0.3d 版

2005 年 3 月 20 日

Thor Watanabe
Dept. of System Infromation Science
Future University-Hakodate
thor@tex.dante.jp (hakodate12@hotmail.com)
http://tex.dante.jp/

名前	表示	CMYK 値	名前	表示	CMYK 値
GreenYellow		0.15, 0, 0.69, 0	RoyalPurple		0.75, 0.90, 0, 0
Yellow		0, 0, 1, 0	BlueViolet		0.86, 0.91, 0, 0.04
Goldenrod		0, 0.10, 0.84, 0	Periwinkle		0.57, 0.55, 0, 0
Dandelion		0, 0.29, 0.84, 0	CadetBlue		0.62, 0.57, 0.23, 0
Apricot		0, 0.32, 0.52, 0	CornflowerBlue		0.65, 0.13, 0, 0
Peach		0, 0.50, 0.70, 0	MidnightBlue		0.98, 0.13, 0, 0.43
Melon		0, 0.46, 0.50, 0	NavyBlue		0.94, 0.54, 0, 0
YellowOrange		0, 0.42, 1, 0	RoyalBlue		1, 0.50, 0, 0
Orange		0, 0.61, 0.87, 0	Blue		1, 1, 0, 0
BurntOrange		0, 0.51, 1, 0	Cerulean		0.94, 0.11, 0, 0
Bittersweet		0, 0.75, 1, 0.24	Cyan		1, 0, 0, 0
RedOrange		0, 0.77, 0.87, 0	ProcessBlue		0.96, 0, 0, 0
Mahogany		0, 0.85, 0.87, 0.35	SkyBlue		0.62, 0, 0.12, 0
Maroon		0, 0.87, 0.68, 0.32	Turquoise		0.85, 0, 0.20, 0
BrickRed		0, 0.89, 0.94, 0.28	TealBlue		0.86, 0, 0.34, 0.02
Red		0, 1, 1, 0	Aquamarine		0.82, 0, 0.30, 0
OrangeRed		0, 1, 0.50, 0	BlueGreen		0.85, 0, 0.33, 0
RubineRed		0, 1, 0.13, 0	Emerald		1, 0, 0.50, 0
WildStrawberry		0, 0.96, 0.39, 0	JungleGreen		0.99, 0, 0.52, 0
Salmon		0, 0.53, 0.38, 0	SeaGreen		0.69, 0, 0.50, 0
CarnationPink		0, 0.63, 0, 0	Green		1, 0, 1, 0
Magenta		0, 1, 0, 0	ForestGreen		0.91, 0, 0.88, 0.12
VioletRed		0, 0.81, 0, 0	PineGreen		0.92, 0, 0.59, 0.25
Rhodamine		0, 0.82, 0, 0	LimeGreen		0.50, 0, 1, 0
Mulberry		0.34, 0.90, 0, 0.02	YellowGreen		0.44, 0, 0.74, 0
RedViolet		0.07, 0.90, 0, 0.34	SpringGreen		0.26, 0, 0.76, 0
Fuchsia		0.47, 0.91, 0, 0.08	OliveGreen		0.64, 0, 0.95, 0.40
Lavender		0, 0.48, 0, 0	RawSienna		0, 0.72, 1, 0.45
Thistle		0.12, 0.59, 0, 0	Sepia		0, 0.83, 1, 0.70
Orchid		0.32, 0.64, 0, 0	Brown		0, 0.81, 1, 0.60
DarkOrchid		0.40, 0.80, 0.20, 0	Tan		0.14, 0.42, 0.56, 0
Purple		0.45, 0.86, 0, 0	Gray		0, 0, 0, 0.50
Plum		0.50, 1, 0, 0	Black		0, 0, 0, 1
Violet		0.79, 0.88, 0, 0	White		0, 0, 0, 0

口絵 I color パッケージで標準的に使用できる色の名前



口絵 II PSTricks **による状態遷移図の描画**

この状態遷移図を描くのに以下の入力をした.

```
%\usepackage[dvips]{graphicx,color}
%\usepackage{pst-all,pstcol}
\begin{TeXtoEPS}
\begin{math}
 \rput(0,3.5){\rnode{root}\relax}
 \cnodeput[fillstyle=solid,fillcolor=cyan](1,3){N}{N}
 \cnodeput[fillstyle=solid,fillcolor=magenta](3,3){M}{M}
 \cnodeput[fillstyle=solid,fillcolor=yellow](5,3){S}{S}
 \cnodeput[doubleline=true,fillstyle=solid,fillcolor=red](3,1){Q}{q}
 \psset{arrows=->,labelsep=3pt}
 \ncarc{N}{M}   \Aput{b}
 \ncarc{M}{S}   \Aput{a}
 \ncarc{S}{M}   \Aput{b}
 \ncline{N}{Q}  \Bput{y}
 \ncline{M}{Q}  \Aput{y}
 \ncline{S}{Q}  \Aput{y}
 \ncline[doubleline=true]{root}{N}
 \nccircle{->}{N}{10pt} \Bput{a}
 \nccircle{->}{M}{10pt} \Bput{b}
 \nccircle{->}{S}{10pt} \Bput{a}
 \nccircle[angleA=180]{->}{Q}{10pt} \Bput{\{a,b,y\}}
\end{math}
\end{TeXtoEPS}
```

まえがき

フリーウェアとは何ぞや

LaTeX はフリーウェアですがその重要なマニュアルはフリーではありません．LaTeX プロジェクトメンバーの Michel Goossens 氏や Sebastian Rahtz 氏，Frank Mittelbach 氏，Leslie Lamport 氏らが出版しているマニュアルは日本語訳で 1 冊 5,000 円くらいのお値段です．そこで LaTeX ユーザーが必携といわれている書籍は

- 『文書処理システム LaTeX 2_ε』 [28] が 3,000 円.
- 『LaTeX コンパニオン』 [29] が 4,800 円.
- 『LaTeX グラフィックスコンパニオン』 [30] が 5,400 円.
- 『LaTeX Web コンパニオン』 [31] が 4,800 円.

と 4 冊程あります．「なっ！なんとっ！！必携の本を買ったら 18,000 円もかかるではないか！」と言われるでしょう．Lamport 氏に「人生そんなものだよ.」と言われればそれまでなのですが，これでは貧乏大学生やフリーウェアだから飛びついた方は手を出しづらいのではないかと思います．また，LaTeX の使い方である技術資料は全て公開されていますので，親切な方がウェブページなどで詳しく取り扱っている場合があります．そのようなページを見れば特に困ることはないと思いますが，情報が離れ離れで存在するので，どうも勝手が悪いようです．

Richard Stallman 氏の訴える通りこれがフリーウェアの抱える問題点ではないかと思います．そこで新たにフリーなマニュアルを作成することにしました．ただし LaTeX の既存のマクロ，クラスならびにプログラムの活用方法についての話に限定します．事務的な書類の作成ではなく主にレポートや論文を書くための情報を集めていますので，表に色を付けたいとか，フォントにこだわりたいという情報を含んでいません．さらにマクロ・クラスの作成方法は最小限にとどめていますので，既存の良書を付録 A から参照してください．

FUNNIST について少し

この冊子は主に私が在籍している公立はこだて未来大学内のために作成しているものですので，他大学の方などとは分野が異なることもしばしばあると思われます．その点を留意していただけると助かります．また，このような冊子の作成をしている組織に名前がありまして，これを FUNNIST と呼んでおります．公立はこだて未来大学を英語では *Future Univeristy-Hakodate* と省略して呼ばれますので *FUN* となります．私が発足しようと目論んでいる組織は未来大学の情報システムやネットワークシステム，果ては計算機の総合的な使い方を示す手引書（*Tutorial*）を作成することを目的としています．この

組織での重要な対象はネットワークと情報システムの二つです．この二つを英語にすると *Network* と *Information System* です．そのような理由もありこの組織の名称は *Future University-Hakodate Network and Information System Tutorial* に決定しました．しかし，毎回このように記述するのは骨が折れるので省略して *FUNNIST* と書くことにします．

凡例

本冊子では書体を変更することによって同じ語句でも違った意味を持つものが多数あります．‘dvips’ という語があったとしても ‘dvips’ や ‘`dvips`’，‘*dvips*’，‘*dvips*’ はすべて別の意味を持っています．これらの書体の種類については 3.19 節を参照してください．

書体	意味	例
ローマン体	通常の文章	dvips
サンセリフ体	パッケージやクラス（3.21 節参照）	dvips
タイプライタ体	キーボードからの入力など	`dvips`
イタリック体	変数や強調	*dvips*
スラント体	オプション（3.21.2 節参照）	*dvips*

謝辞

この冊子を作成するためには非常に多くの方々のご協力，ご助言がなければ実現することが難しかったことを容易に想像できます．LaTeX 全般に関しては秋田純一氏，奥村晴彦氏，野村昌孝氏より多くのことを学びました．出版，校正，デザインなどに関しては木村健一氏よりご助言をいただいたり，また書籍を貸して頂きました．

TeX の作者である Donald Knuth 氏，LaTeX の作者である Leslie Lamport 氏，LaTeX 2_ε の開発をされた Frank Mittelbach 氏，Johannes Braams 氏，David Carlisle 氏，Michael Downes 氏，Alan Jeffery 氏，Sebastian Rahtz 氏，Chris Rowley 氏，Rainer Schöpf 氏，TeX の日本語化をして下さった中野賢氏とアスキーの方々，Windows に pTeX を移植してくださった角藤亮氏，dviout を開発された大島利夫氏と乙部厳己（よしき）氏，BibTeX の開発をされた Oren Patashnik 氏，MakeIndex を開発・改良された Pehong Chen 氏と Nelson Beebe 氏，dvipdfm の作者である Mark Wicks 氏，Dvipdfm*x* の保守・管理をされておられる平田俊作氏と Cho Jin-Hwan 氏，PostScript や PDF などのページ記述言語を作成された Adobe 社の方々，さらに，フリーウェア，マクロパッケージなどの作成で，TeX の分野において貢献された方々にも感謝いたします．

秋田純一氏，大友康寛氏，田中健太氏，永田善久氏，野村昌孝氏，三上貞芳氏，和田志保美氏らはこの冊子の作成に貢献してくださった方々です．本当にありがとうございます．

履歴

2004 年 4 月 2 日に誤植を訂正していないバージョン 0.1 をウェブ上で公開しました．この版はバージョン 0.2 であり，誤植の訂正や若干の加筆が行われています．句読点も全角のピリオド・カンマに統一し，つめ掛けについては章見出しでも出力するようにしました．その他『数式の組版』の章に少し空白に関することを加筆しました．索引についても抜けていた人名や語句の補充をしました．相当な部分を修正したので履歴として残しておきます．

さらに 2004 年 8 月 5 日にこれ以上は修正しないという完成版に近いものを公開しました．

2004 年 8 月 19 日，0.3 でおかしな部分（改行が変な部分や索引の倍角ダーシ）を修正しました．参考資料の章を若干修正しました．

あれっおかしいな？と思ったら

この文書は私一人で執筆しておりますから，どこかに間違いや誤植がある確率が高くなっています．「あれっおかしいな？」と思う箇所がありましたら私のホームページ

http://tex.dante.jp/

の掲示板かメールアドレス

thor@tex.dante.jp

にご連絡ください．

フリーソフトウェアとフリーマニュアル

フリーなオペレーティングシステムにおける最大の欠陥は、ソフトウェアの問題ではありません–私たちがこれらのシステムに含めることが可能な、フリーの良質なマニュアルが不足していることこそが問題なのです。私たちの最も重要なプログラムの多くは、完全なマニュアルと共に提供されていません。ドキュメントはいかなるソフトウェアパッケージにおいても必要不可欠な一部分ですから、重要なフリーソフトウェアパッケージがフリーなマニュアルと共に提供されないならば、それは大きな欠陥です。私たちは今日、そのような欠陥を数多く抱えています。

昔々、何年も前のことになりますが、Perl を学ぼうと思ったことがあります。私はフリーなマニュアルを一部入手したのですが、それは極めて読みにくいものでした。Perl ユーザたちに代替品について聞いてみたところ、彼らはより良い入門用マニュアルがあると教えてくれたのですが、しかしそれらはフリーではありませんでした。

これはどうしてだったのでしょう? その良質なマニュアルの著者たちは、マニュアルをを O’Reilly Associates 社のために書き、O’Reilly はそれらを制限的な条件の下で出版したのです。禁複製で禁変更、ソースファイルは入手不可– こういった条件はマニュアルをフリーソフトウェアのコミュニティから締め出してしまいます。

これはこの種の出来事としては最初のものではありませんでした。そして (私たちのコミュニティにとっては大きな損失なのですが) 最後であるとも到底いえません。この出来事以降も、独占的なマニュアル出版者は非常に数多くの著者たちをそそのかし、彼らのマニュアルに制限を加えさせてきました。私は、 GNU ユーザの一人が彼の書いているマニュアルについて熱心に語るのを何度も聞きました。彼はそれによって GNU プロジェクトを援助できると考えていたのです–ところが、彼は続けて、私たちがそれを使うことができないように制限を課すであろう出版者との契約にサインしたと述べたので、私の希望は打ち砕かれるのが常でした。

きちんとした英語で書くということはプログラマの間ではまれなスキルですから、マニュアルをこういったことで失う余裕は全く無いのです。

フリーな文書で問題となるのは、フリーソフトウェアと同様、自由であり、価格ではありません。これらのマニュアルの問題点は O’Reilly Associates が印刷されたコピーに代価を要求するということではないのです–それ自体は別に構いません (フリーソフトウェア財団も、フリーな GNU マニュアルの印刷されたコピー を販売しています)。しかし、GNU マニュアルはソースコード形式で入手可能なのに対し、O’Reilly のマニュアルは紙媒体でしか入手できません。GNU マニュアルは複写および変更の許可と共に提供されています。Perl のマニュアルはそうではありません。こういった制限こそが問題です。

フリーなマニュアルであるための基準はフリーソフトウェアのそれとかなり良く似ています。その基準とは、すべてのユーザにある種の自由を与えるということです。マニュア

ルをオンラインまたは紙媒体でそのプログラムのすべてのコピーと一緒に提供できるよう、再配布 (商業的再配布を含む) が許可されていなければなりませんし、変更の許可も重要です。

普遍的なルールとして、人々にあらゆる種類の文章や書籍を変更する許可を与えることが必須だとは私も思いません。書かれたものに関する論点が、ソフトウェアのそれと同じである必要は無いのです。例えば、あなたや私に、この文章のような、自分の行動やものの考え方を説明する論説を変更する許可を与える責任があるとは考えられません。

しかし、フリーソフトウェアのための文書にとって変更の自由がなぜ重要であるかについては、特別の理由があるのです。人々がソフトウェアを変更する権利を行使して、ソフトウェアに機能を加えたり変更したりすると、彼らが良心的であればマニュアルも変更しようとするでしょう–そうすれば彼らは正確で有用なドキュメントを変更されたプログラムと共に提供できるわけです。そこで、プログラマが良心的になることや仕事を完遂することを禁止する、あるいはより正確に言えば、プログラムを変更するなら彼らがゼロから新しいマニュアルを書きなおすことを要求するマニュアルは、私たちのコミュニティのニーズを満たすものとは言えないのです。

全面的な変更の禁止は受け入れられませんが、ある種の制限を変更の手法に課しても何の問題も引き起こしません。例えば、原著者の著作権表示を保存することを要求したり、元の配布条件や著作者名のリストの保存を要求するのは OK でしょう。また、変更された版は、それらが変更されたという告知を含んでいなければならないという要求をするのも構いませんし、ある節全体の削除や変更を禁止することすらも、そういった節が技術的なトピックを扱っていない限り問題にはなりません (GNU マニュアルの一部はそういった節を含んでいます)。

この種の制限が問題にならないのは、現実的な問題としては、それらが良心的なプログラマがマニュアルを変更されたプログラムにあわせて手直しすることを止めさせないからです。言い換えれば、そういった制限はフリーソフトウェアのコミュニティがマニュアルを最大限に活用することを禁止しないのです。

しかしながら、マニュアルの技術的な内容はすべて変更可能でなければなりません。そして、変更の結果をあらゆる一般的なメディア、すべての通常チャンネルで配布できなければなりません。そうでなければ制限はコミュニティを妨げますので、マニュアルはフリーではなく、そこで私たちには他のマニュアルが必要となります。

不幸にも、独占的なマニュアルが存在する場合には、もう一つマニュアルを書いてくれる人を探すのは困難なことが多いのです。障害となるのは、多くのユーザが独占的なマニュアルで十分と考えていることです–そこで、彼らはフリーなマニュアルを書く必要を認めないのです。彼らはフリーなオペレーティングシステムが、満たすべき欠落を抱えていることが分からないのです。

どうしてユーザは独占的なマニュアルで十分だと思うのでしょう? 何人かは、この問題について考えたことがないのでしょう。私はこの論説が、そうした状況をいくらかでも変えることを期待しています。

他のユーザは、独占的なマニュアルは独占的なソフトウェアが受け入れられるのと同じ

理由から受け入れられるものだと考えます。彼らは全く実用的な見地からのみ物事を判断し、自由を基準として適用しないのです。こういった人々にも彼らなりの意見を持つ資格がありますが、しかしそういった意見は自由を含まない価値から飛び出してくるものなので、彼らは自由を評価する私たちの基準のよりどころとはなり得ません。

この問題についての話を広めてください。私たちはマニュアルを独占的な出版のために失い続けています。もし私たちが独占的なマニュアルは十分では無いという思想を広めるならば、おそらく GNU を文書を書くことで助けたいと思う次の人は手遅れになる前に、彼らがそれを結局のところフリーにしなければならないということを悟るでしょう。

私たちはまた、商業的出版者に対して、独占的なマニュアルに代わってフリーでコピーレフトの主張されるマニュアルを販売することを薦めています。あなたがこの動きを援助できる一つの方法は、マニュアルを買う前にその配布条件をチェックし、コピーレフトなマニュアルを非コピーレフトなものよりも好んで買うということです。

グニューの頭
(HP より)

Free Software Foundation について

前述の『フリーソフトウェアとフリーマニュアル』は Free Software Foundation の Richard Stallman 氏 によって書かれたウェブページを八田真行氏[*1]が翻訳したものです．このGNU（グニュー）に関する文章を公開している団体を Free Software Foundation （FSF[*2]）と言います．彼らのウェブページについては http://www.gnu.org/ にアクセスすると良いでしょう．私がこの冊子を作ったきっかけもこの FSF の活動によるものなので，是非ご覧ください．

[*1] mhatta@gnu.org
[*2] gnu@gnu.org

目次

第 1 章
ゲームを始める前に

この章では LaTeX の前提知識を説明します．LaTeX は組版ソフトですから組版に関する知識が必要なのは言うまでもありません．さらに LaTeX を始める前にその歴史背景と諸事情を知ることも実は結構役に立ったりします．

1.1 組版とはなんだろうか

組版（くみはん）とはある媒体，特に書籍などの紙のうえに読者が読みやすいように必要な情報を適切な位置に配置することです．

現代ではコンピュータ上で文書を組版できるようになりました．だれでも手軽に印刷用の美しいフォントを用いた組版が可能です．ここで文書がどのようにして組版されているのかを少し説明します．

世界中で出版されている書籍は一定のルールに沿って組版されているものです．たとえば 1 行を何文字にするか 1 ページを何行にするかなどの約束事があります．このような様式をどのようにするのかは各出版社や各種学会が各々で定めています．

なぜこのような決まり事があるかというと文字や図を含む本や雑誌は必ず誰かに見てもらう，読者を相手にしていることを前提としているからです．その本の内容に合わせて読者にとって読みやすい本とは何かを追求してこのような様々な書式が存在します．

LaTeX を用いるとユーザーがそのような高度な技術を持っていなくてもプログラムが半自動的に組版するようになっています．しかし最低限のルールを覚えなければとても出たら目な文書に仕上がってしまいます．

世の中にはパソコンで動くワープロソフトと呼ばれるソフトウェアが多数存在するようです．OpenOffice.org とか Microsoft Office などがその類です．これらのソフトと組版ソフトである LaTeX は何が違うのでしょう．それをこれからじっくり眺めていくことにしましょう．まずは LaTeX の周辺と予備知識を説明します．

1.2 TeX とは何か

TeX（てっく） [47] とは Donald Knuth 氏によって開発された組版プログラムです．特筆すべきことは数式の処理に優れていること，簡単なレポートの作成から論文の作成，果ては商業出版にも耐えうる機能を持っていることなどです．TeX の読み方ですがこれは英語ではな

くギリシャ語の τ と ϵ と χ の綴りですから日本でこれに該当する発音がないと思われます．一般的には「てっく」と発音するのが無難だと思います．

この TeX を用いれば書式が統一されてより美しい文書を作成することができます．オフィスソフトよりも論理構造のきちんとした文章が作成できます．無論 TeX がそれらの処理を自動でやってくれるわけではないので，ユーザーが必要な命令を明示的に記述します．覚えるまでに多少時間がかかるのが難点ですが，慣れれば文書の編集には便利なツールです．

1.3　WYSIWYG とは何か

WYSIWYG（ウィジィウィグ）とは “What You See Is What You Get” の略で「見たままのものが得られる」という意味合いでワープロソフトのように画面で見たイメージがそのまま紙などに出力されることを言います．TeX は WYSIWYG ではありませんから紙に出力されるイメージをどうにかして確認する作業が必要になります．毎回紙に印刷するのは大変時間を必要とし，なおかつ地球環境の悪化を促進するものです．そのためコンピュータの画面上で確認作業をします．これをプレビューと言います．なんだか面倒な手順を踏んでいるように思えますがこれは WYSIWYG に比べると応用が広いのです．WYSIWYG はときに “What You See Is all You’ve Get” と言われることもあり，「見たままのものしか得られない.」という意味合いで使われます．TeX とはまったく正反対のシステムのことを指します．

1.4　一括処理とは何か

TeX のもう一つの特徴として通常のプログラミング言語と同じように原稿を一括で処理する方式を採用しています．これは当然のことなのですがワープロソフトとは大違いです．一括処理を採用しているということは，仕上がりは**全てのページの組み版が終了するまで**分からないということです．マークアップ方式の言語ならば文書の全体をフォーマット（マークアップ付け）しなければならないのです．

1.5　LaTeX とは何か

組版プログラムとしての TeX は完成度が非常に高く，高性能です．そのためちょっとした記事を書こうと思っても手続きが非常に多いようです．そこであらかじめいくつかの命令を定義しておき，その定義を使って特定の書式を用意しておけば簡単に文書を作成することができます．このシステムを開発されたのが Leslie Lamport 氏で，彼の作成したシステムを LaTeX と言います．これも発音が微妙で，読み方には何種類かあります．作者の Lamport 氏が言う分には「らてふ」と発音するのが正しいと思いますが，日本では広く「らてっく」と発音されていますので「らてっく」でも良いと思います．

LaTeX も HTML と同様のマークアップ方式を採用しています．簡単な例を挙げると

```
<CENTER>人類普遍の原理である</CENTER>
```

などがあります．これは「人類普遍の原理である」という文字列を中央に寄せたいので，始まりと終わりをそれぞれ，‘<CENTER>’ と ‘</CENTER>’ という二つの規則で文字を囲んでいます．これがマークアップ方式の典型的な例です．マークアップ方式ではそれぞれの要素に属性を与えて文書を記述するということを行います．これを LaTeX で示せば

```
\begin{center}人類普遍の原理である\end{center}
```

となるので，先程の HTML の記述に良く似ているのがお分かりになるでしょう．

TeX も LaTeX も欧文言語圏のためのプログラムですから標準では日本語を処理することが出来ませんが，中野賢氏を始めとするアスキーの方々が TeX の日本語化をしてくださいましたので，今ではこの TeX/LaTeX を使って高品質な日本語組版ができるようになりました．アスキーによって日本語化された TeX や LaTeX をそれぞれ pTeX，pLaTeX と呼びます．

LaTeX が最初に登場したときのバージョンは 2.09 で，この頃のものを LaTeX2.09 と呼びます．それから煩雑であった LaTeX2.09 システムを整備して LaTeX 2ε が LaTeX プロジェクトチームによってリリースされました．次の LaTeX のバージョンは LaTeX3 と言われていますが，このバージョンが登場するのはもう少し先のようです．

1.6 情報の入手先

一般に Lamport 氏の『文書処理システム LaTeX 2ε』 [28] やコンパニオンシリーズ [29, 30, 31]，入門用として奥村晴彦氏の『LaTeX 2ε 美文書作成入門』 [27]，それに藤田眞作氏の書籍 [34] や，乙部厳己氏と江口庄英氏による *Another Manual* シリーズ [35, 36, 37] が参考になると思います．以上の書籍は入手が容易だと思います．

LaTeX やその周辺の情報を入手するにはインターネットを使って収集するのが良いでしょう．インターネット上で LaTeX に関する情報交換がされています．

付録 A に詳細な情報の入手先をまとめましたので，そちらからウェブのリンクなどを辿ってみてください．

1.7 より良く学ぶために

とにもかくにも Leslie Lamport 氏が書いた *LaTeX A Document Preparation System* [28] という本を読みましょう．この本を読めば LaTeX の基本はほとんど習得することができます．「うーん，でも LaTeX ってフリーウェアなんでしょ？フリーマニュアルじゃないじゃん，ほげほげぇ．」とどこからか聞こえてきました．まあしょうがないじゃないですか，そこは勘弁してくださいよ．きちんとした本として出すためには出版社と独占契約をしないといけないんですから．愚痴を言っても始まらないので次に進みましょう．

LaTeX の拡張的なことは 150 以上のマクロパッケージを解説した Michel Goossens 氏らが書いた『LaTeX コンパニオン』 [29] という本を読みましょう．これは原書で第 2 版があります．うーん，確かにマクロパッケージを紹介する本としては出来が良いのですが「これも買わないとあかんのですか？ぴよぴよぉ．」とまた空から声が聞こえてきます．

まぁ，買ってあげてください．この本の収入が LaTeX プロジェクトへの貢献にもつながるのですから．

さらに画像操作に関しては Michel Goossens 氏らが書いた『LaTeX グラフィックス コンパニオン』[30] という本を読みましょう．ほほぅ，これは PostScript や LaTeX における画像操作について詳しく書いてある本なのですが，またまた「わしはそんなに本を買ってられないぞ！」と怒りにも似た声が聞こえてきます．

極めつけに World Wide Web で LaTeX を扱うためには Michel Goossens 氏らが書いた『LaTeX Web コンパニオン』[31] という本を読みましょう．本当に詳しく解説しているのですが「わしはもう疲れたわぁ．」というとどめを刺されたような声が聞こえます．

ふむふむ，疲労困ぱいしているようですがまだまだこれからですよ，頑張ってついてきてください．

第 2 章

LaTeX の基本

まずは操作方法などの LaTeX の基本を習得しましょう．この冊子ではあなたが「何を書くべきか」ではなく「どうやって書くべきか」ということしか説明しないことを始めに断っておきます，当たり前ですが．

2.1 基本の基本

2.1.1 処理の流れ

コマンドを覚える前にまずは LaTeX での処理の流れを抑えておきましょう．テキストファイルに文章そのものと**コマンド**というものを書き，それを LaTeX 処理し，成形結果を確認するといったことを何度か繰り返して最終的な版を仕上げます（図 2.1）．ここで「成

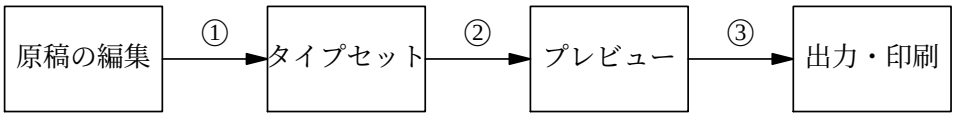


図 2.1　処理の流れ

形」とありますが，LaTeX では元のソースファイルそのものに変更を加えて「整形」するのではなく，そこから新規に DVI ファイルというものを成形するのです（最近は LaTeX ファイルから直接 PDF を作るプログラム pdfLaTeX なども存在します）．

1. 原稿（ソースファイル）の編集
   LaTeX を使うためには文章だけではなく，文章の構造や書式を決定するコマンドと呼ばれるものを記述します． この原稿をソースファイルと呼びます．原稿はメモ帳や Emacs などのテキストエディッタで編集します（まれに端末から直で行く人もいます）．Unix 系 OS では
   ```
   ▉ emacs filename.tex &
   ```
   とすると Emacs が立ち上がると思います．XEmacs でも何でも良いです．これが図 2.1 の①の矢印に対応します．
2. タイプセット （組版）
   ソースファイルができたらそれを**成形**します．そのときに使うプログラムは欧文のみの場合は latex，日本語を扱うときは latex を日本語化した platex です．シェルやコマンドプロンプトなどから

```
platex filename.tex
```

とすれば文書が成形されます．この作業のことを良く**タイプセット**とか**コンパイル**と言います．これが図 2.1 の②の矢印に対応します．

3. プレビュー（確認作業）
   今度はコンピュータの画面上で成形された結果を見ます．このとき filename.tex そのものが整形されるわけではなく新たに filename.dvi というファイルが作られます．これが LaTeX による組版後の文書になります．この組版後の結果をコンピュータ上で確認する作業のことを**プレビュー**すると言います．Unix 系 OS ならば

   ```
   xdvi filename.dvi &
   ```

   などとすると良いでしょう．最後のアンド '&' もあると便利です．dviout をインストールした Windows ならばダブルクリックするだけで見られるでしょう．これが図 2.1 の③の矢印に対応します．

このような流れがあることを確認して実際に動くかどうかを試してみましょう．

2.1.2　動かしてみる

インストールが済んでいれば LaTeX が動きます．インストールまで進んでいないという方は近くの詳しい方に聞いてみてください．「LaTeX をやりたいんですが，ほげほげぇ.」と言い出せば大抵の LaTeX ユーザは親切に対応してくれると思います（「ほげほげぇ」という部分が大切なキーワードとなります）．

とりあえず自分のいつも使っているテキストエディッタ（メモ帳とか Emacs とか）で以下のようなファイル hoge.tex

```
\documentclass{jarticle}
\begin{document}
こんにちは\LaTeX !!
\end{document}
```

を作成してください．Unix 系 OS ならば Emacs で良いでしょう．日本語を打ち込むためには Emacs の場合はまず Emacs のウィンドウ下部に注目してください．ウィンドウの下部の表示には

```
[-]J.:--Emacs:  hoge.tex (LaTeX)-[L1-All---|
```

となっていると思います．一番右側の [-] という部分が半角入力（英数入力）か全角入力（日本語入力）かの違いを表します．うえの状態は英数文字の入力ができます．ここで半角・全角の入力を切り替えるためには CTRL を押しながら ¥ を押します．すると [-] という表示から [あ] という表示になると思います．表示は使っている「かな漢字変換プログラム」によって若干違うかもしれません．[あ] の状態ですと日本語が入力できる状態です．最近のパソコンと呼ばれるコンピュータには **半角/全角** というキーがあり，Windows の場合はそれで半角と全角の切り替えを行いますが，Unix 系 OS は違いま

すので注意してください．詳しくはご自分のエディッタのマニュアルを見るなどの対応をしてください．

話を戻して次はタイプセット作業（LaTeX 処理, またはコンパイル）をします．Windows ならば[スタート]メニューから[ファイル名を指定して実行]というメニューがあるので, そこに「command」と入力して ‘OK’ ボタンを押せばコマンドプロンプトが起動するはずです．そしてシェル上やコマンドプロンプトでファイルが存在するディレクトリ（Windows の方はフォルダ）に移動して

```
▌platex hoge.tex
```

としてタイプセットしてください．すると画面には（端末）に

```
This is pTeX, Version 3.14159-p3.0.4 (sjis) (Web2C 7.3.9)
pLaTeX2e (based on LaTeX2e <2001/06/01> patch level 0)
(1) (./hoge.aux) )
Output written on hoge.dvi (1 page, 236 bytes).
Transcript written on hoge.log.
```

のように表示されると思います．始めにバージョン情報を表示して終わりには hoge.dvi に組版後のファイルを出力し，処理状況を hoge.log に書き出したことになっています．hoge.tex をタイプセットして出力された hoge.log にはエラーメッセージなどの重要な情報が書かれているときがあるので何か問題が発生したときは眺めてみると良いでしょう．

タイプセット後にはいくつかのファイルが生成されています．ls（Windows の方は dir）コマンドで

```
▌ls hoge.*
```

とすると

```
hoge.aux hoge.dvi hoge.log hoge.tex
```

の四つのファイルが存在することを確認してください．LaTeX の原稿であるソースファイル hoge.tex をタイプセットしただけで三つもファイルが生成されました，これは只事ではありません．これらのファイルが何であるかを説明します．

hoge.aux　次回のタイプセットに必要になる**中途ファイル**．目次の作成や相互参照をするために必要なファイル．

hoge.dvi　hoge.tex をタイプセットして出来上がった印刷できる**成形ファイル**．DVI ファイルと呼ばれる．

hoge.log　hoge.tex をタイプセットしたときの処理状況やどのような流れで処理をしたのかが書いてある**ログファイル**．

hoge.tex　先程作成した LaTeX の原稿である**ソースファイル**．

2.1.3　原稿作成時の注意点

これまでの作業ができていれば数式や図表を含まない簡単な文書を作成できることでしょう．そして実際に長い文章を打ち込んでみてください．ただし 10 個の半角記号

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

は特殊文字として LaTeX に別の仕事をさせるために使いますのでそのまま使うことができません．さらに 3 個の記号は出力が違う文字記号になります．

```
| < > はそれぞれ — ¡ ¿
```

となることでしょう．以上の 13 個の記号を文章中で出力するために面倒ですが，バックスラッシュ（円）記号を補ったり長い命令を打ち込みます．

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

~	\textasciitilde	^	\textasciicircum	\	\textbackslash
|	\textbar	<	\textless	>	\textgreater

▸ **問題 2.1** 半角文字と全角文字を混在させて長い文章の入力をしてください．特殊記号も含めるようにすると良いでしょう．

2.1.4　フォルダ・ファイルの基本的な操作

ディレクトリの移動方法を知らない，フォルダの作り方を知らないという方のためにコマンドプロンプトやシェルでの主要なコマンドを紹介します．まず Windows では表 2.1 などの基本的なコマンドが提供されています．それぞれのコマンドの使い方（ヘルプ）は

```
C:\> mkdir /?
```

とすることで表示できます．または[スタート]メニューの[ヘルプ]からコマンドプロンプトについて調べてみても同様のことができます．どちらかというと Windows ヘルプを利用したほうが良いでしょう．コマンドプロンプトなどの操作に慣れていないという方は 9.21 節を参照して LaTeX の入力支援環境を使うのも良い方法です．

表 2.1　Windows OS の基本コマンド

コマンド名	意味
mkdir	新規にフォルダを作成
cd	ディレクトリを移動
dir	ファイルの情報を表示
move	ファイル名を変更したり移動
copy	ファイルをコピー
del	ファイルを削除
help	コマンドのヘルプを表示

Unix 系 OS の方はコマンドを覚えなければ仕事にならないでしょう．シェルと言っても何種類かありますし，シェルに関しては 1 冊の本になるくらい奥の深いものなので詳細はそれらに譲ります．ここでは基本的なファイル操作のコマンドだけを表 2.2 に紹介します．それぞれのコマンドの簡単なヘルプが見たいときは

```
▉ mkdir -help | less
```

のようにすると less がページを整形します．

もう少し詳しいヘルプが見たいときは

```
▉ man mkdir
```

とします．もっと詳しいヘルプが見たいときは

```
▉ info mkdir
```

のようにすると info がページを整形します．less や info の操作方法は若干癖がありますので慣れるまで時間がかかるかもしれません．Unix 系 OS の基本的な操作方法，正規表現，プロセス，ファイル・ディレクトリの概念などについては付録 A の参考資料を参照してください．Unix 系 OS ならば

表 2.2 Unix 系 OS の基本コマンド

コマンド名	意味
mkdir	新規にフォルダを作成
cd	ディレクトリを移動
ls	ファイルの情報を表示
mv	ファイル名を変更したり移動
cp	ファイルをコピー
rm	ファイルを削除
help	内部コマンドのヘルプを表示
man	コマンドのヘルプを表示

```
▉ emacs hoge.tex &
▉ platex hoge.tex
▉ xdvi hoge.dvi &
```

の三つの操作ができればなんとかなります．この冊子では Unix 系 OS の基本ツールなどまで詳しく解説しないのでご自分で調べてみてください．

コマンドに対してシェル上で一緒に渡す文字列のことを**引数**と呼びます．そして多くのコマンドは**コマンドラインオプション**といってハイフン ‘-’ かハイフンが二つ ‘--’ で始まる引数を特別なスイッチとして扱います．このスイッチによってそのコマンドは挙動を変えます．それぞれのコマンドでどのようなコマンドラインオプションが使えるのかは各プログラムのヘルプを調べます．

▸ **問題 2.2** 以下の作業を端末上から行ってください．Windows の方は mv を move に，ls を dir に，スラッシュ ‘/’ を円 ‘¥’ と置き換えてください．

```
▉ echo hogehoge >> hoge.txt
▉ echo gehogeho >> hoge.txt
▉ mkdir geho
▉ cd geho
▉ mv ../hoge.txt ./
▉ ls
▉ more ./hoge.txt
▉ ls ../
```

上記の操作はどのような結果をもたらしたと考えられるでしょうか．新規にディレクトリ geho を作成し，現在のディレクトリに存在していたファイル hoge.txt を geho ディレクトリに移動したと考えられるでしょう．最後の操作でうえの階層のディレクトリに hoge.txt がないことでそれを確認できます．

2.1.5　エラーに遭遇する

LaTeX 処理をしているとエラーに悩まされるかもしれません．LaTeX は文章中にコマンドなどに関するエラーを発見するとそこで処理を中断します．処理を中断するとユーザにどうすれば良いかを促します．このとき端末には疑問符 '?' が表示されます．

▹ **例題 2.3** まずは以下のファイル error.tex を作成してください．

```
\documentclass{jarticle}
\begin{document}
Hello & Goodbye! Give me $100! Give me 100%!
Under_bar is stranger. Is sharp sing #?
No its' \#. Hello \& Goodbye!!
\end{document}
```

次にこのファイルをタイプセットしてください．すると端末には

```
! Misplaced alignment tab character &.
1.3 Hello &
          Goodbye! Give me $100! Give me 100%|
?
```

と表示されるでしょう．最後の行に疑問符 '?' が表示されています．この状態はユーザに何らかの操作を促している状態です．どうやら 3 行目でアンド '&' を不正に使っていると言われています．ここで Enter キーを押すとさらに

```
! You can't use 'macro parameter character #' in math mode.
l.4 Under_bar is stranger. Is sharp sing #
?
```

と表示されます．シャープ '#' も間違った使い方をしていると指摘されました．極め付けにもう 1 度 Enter キーを押すと

```
! Missing $ inserted.
<inserted text>
              $
l.6 \end{document}
?
```

という表示になります．今度はドル '$' を不正に使ったと言われました．以上のことから 3 行目から 6 行目にかけて半角記号の使い方が間違っていることが分かりました．ソースファイルをもう 1 度確認し，どこがどう違うのかを判別し修正してください．修正後のファイルは以下のようになるでしょう．

```
Hello \& Goodbye! Give me \$100! Give me 100%!
Under_bar is stranger. Is sharp sing \#?
No its' \#. Hello \& Goodbye!!
```

3 行目のアンド '&' とドル '$' と 4 行目のシャープ '#' にバックスラッシュ '\' を付けます．これを再びタイプセットしてみてください．今度は

```
! Missing $ inserted.
<inserted text> $
l.4 Under_
        bar is stranger. Is sharp sing \#?
?
```

ドル ‘$’ の書き忘れがあると言われています．4 行目のエラーメッセージで丁度アンダーバー ‘_’ の部分で表示が改行されていますから，この部分に間違いがあることが分かります．どうやらアンダーバーはドル ‘$’ などと同じようにバックスラッシュが必要なようです．ここでとりあえず Enter キーを押してタイプセットを終了してください．一つ目のエラーとして

```
! Missing $ inserted.
<inserted text> $
l.6 \end{document}
?
```

が表示されます．タイプセットは中断しませんが

```
Overfull \hbox (152.35132pt too wide) in paragraph at lines 3--6
[]\OT1/cmr/m/n/10 Hello & Good-bye! Give me $100! Give me
100Under$[]\OML/cmm/m/it/10 arisstranger:Issharpsing
\OT1/cmr/m/n/10 #?\OML/cmm/m/it/10 Noits[]\OT1/cmr/m/n/10 #
\OML/cmm/m/it/10 :Hello\OT1/cmr/m/n/10 &\OML/cmm/m/it/10
Goodbye\OT1/cmr/m/n/10 !!$
```

というごちゃごちゃした表示がでます．これは `Overfull \hbox` という**警告**であることが分かります．次に成形後の DVI ファイル `error.dvi` をプレビューしてください．すると行がページをはみ出しています．先程のアンダーバーに関するエラーにおいて

```
<inserted text> $
```

という表示がありました．どうやら LaTeX は自動的にドル ‘$’ を挿入したようです．‘b’ という文字が ‘r’ の下付きの添え字になっています．さらに ‘Give me 100Under$_b$’ となっており入力されたパーセント ‘%’ と感嘆符 ‘!’ が出力されておらず，次の行の ‘Under’ とくっついています．どうやら先程のごちゃごちゃした警告はこの行がページをはみ出していることを言っているようです．ですからファイル `error.tex` はさらに次のように修正されます．

```
Hello \& Goodbye! Give me \$100! Give me 100\%!
Under\_bar is stranger. Is sharp sing \#?
No its' \#. Hello \& Goodbye!!
```

これで望み通りうまくいきそうです．実際に上記のファイルをタイプセットし，その結果を吟味してください．

LaTeX の原稿をタイプセットしたときに端末に疑問符 ‘?’ が表示されて処理が中断しますが，この段階でこちらも疑問符 ‘?’ で返事を返すと

```
Type <return> to proceed, S to scroll future error messages,
R to run without stopping, Q to run quietly,
I to insert something, E to edit your file,
1 or ... or 9 to ignore the next 1 to 9 tokens of input,
H for help, X to quit.
?
```

と表示されます．疑問符 '?' が表示されている段階で上記に挙げるようなキーの入力をすると何らかの対処ができるようです．

Enter　エラーに対して LaTeX が適当な対処をした後にタイプセットを続行します．
S　Enter キーを押し続けたことと同じ動作をします．
R　エラーが検出されても停止せずにノンストップでタイプセットします．
Q　Q を押した場合は**バッチモード**に入り処理が続きます．
I 〈文字列〉　文字列を挿入してタイプセットを続けます．元の原稿に修正は加えられません．
H　そのエラーに対する英語のヘルプを端末に表示します．
X　ゲームを終了します．

X キーは余り押してはいけません．括弧が足りないというエラーの場合はとりあえず Enter キーを押せばそのまま処理を続行できます．

タイプセットをしてアスタリスク '*' が表示されて処理が中断するときがあります．Enter キーを押しても同じメッセージが表示されてどうにもならなくなります．

```
*
(Please type a command or say '\end')
```

この場合端末から

```
▉ \end{document}
```

と入力して処理が終了しなかった場合は強制的にプログラムを終了してください．ソース中で何かミスをしていると思われます．

2.1.6　プレビューアの操作

プレビューを行うプログラムのことを**プレビューア**と言います．OS によって使用可能なプレビューアが異なります．Windows ならば大島利雄氏の dviout, Unix 系ならば xdvi, Red Hat ならば pxdvi などを使い `hoge.dvi` を各アプリケーションで開きます．Windows の場合は dviout に関する豊富なヘルプやマニュアルが用意されているのでそちらを読んでみてください．ここでは Unix 系 OS で広く使われている xdvi を例に操作方法を説明します．まずシェル上で `hoge.dvi` の存在するディレクトリに移動し，xdvi に対してファイル名を指定し，

```
▉ xdvi hoge &
```

のようにします．Unix 系 OS ならばアンド ‘&’ をつけてバックグラウンドで起動します．こうするとタイプセットを再度したときに自動的に DVI ファイルを再表示します．dviout でも同様の再表示機能があります．

xdvi の基本的な操作方法を説明します．右側に枠で囲まれた文字がボタンになっています．ボタンのように見えませんが一応押せます．さらにボタンの右側にはページ番号があり，ページ番号をクリックすると該当するページを表示します．

xdvi でのマウスのクリックは拡大の機能を持っています．それぞれ

左クリック　少し拡大．
中央クリック　普通に拡大．
右クリック　かなり拡大．

となっています．また，右側にある ‘Quit’ とか ‘Abort’ などはボタンで，主なボタンの機能は以下のとおりとです．

Quit　xdvi を終了する．
Reread　一度読み込んだファイル ⟨*file*⟩.dvi を再描画する．
First　先頭ページに移動する．
Prev　前ページに移動する．
Next　次ページに移動する．
Last　最終ページに移動する．
View PS　PostScript ファイルを見る．
File　DVI ファイルを別に開く．

終了するには ‘Quit’ ボタンを押します．

2.1.7 ちょっと休憩

ここまで操作して少し疲れたでしょうから，ここで休憩をしましょう．DVI ファイルと言われてもなじみの薄いファイル形式かもしれません．そこでこの DVI ファイルを別のファイル形式の変換してしまいましょう．例えば PostScript に変換するには（Windows の方は dvipsk）

```
▌dvips -o hoge.ps hoge.dvi
```

とすると hoge.dvi から hoge.ps が生成されます．この hoge.ps を

```
▌ps2pdf hoge.ps hoge.pdf
```

として PDF ファイル hoge.pdf を生成することもできます．Postscript ファイル hoge.ps をプリンタに出力するには

```
▌lpr -P プリンタ名 hoge.ps
```

とすることで印刷要求を送信できます．〈プリンタ名〉についてはシステムの管理者に聞くなどしてください．

タイプセット時に拡張子.tex を省略して

```
▉ platex hoge
```

としてみてください．以上のような例でも LaTeX プログラムは hoge.tex を探し出して処理してくれるでしょう．これは LaTeX で使われているファイル検索の仕組みに関係しています．LaTeX では Kpathsearch というファイル検索機構を採用しています．まず上記の例では hoge.tex があるときに

```
▉ platex hoge
```

とすると Kpathsearch が自動的に拡張子をつけてファイルを探します．このときどのディレクトリ（フォルダ）から探し出すかという情報が必要になります．この情報が記載されたファイルは texmf.cnf というファイルです．ほとんどの環境でディレクトリ $texmf/web2c/以下にあります．ここで$texmf は LaTeX のファイルを格納すべき一番上のディレクトリを示します．Unix 系 OS ならば/usr/local/lib/texmf/などだったり Windows ならば C:¥usr¥local¥share¥texmf¥だったりするでしょう．

```
▉ kpsewhich texmf.cnf
```

とするとどこに texmf.cnf が存在するのかが分かります．

最近の LaTeX プログラムは Kpathsearch に対応しているので何も意識しなくても適切に設定ファイルやクラスファイルなどを検索してくれます．しかし手動で検索したいときもあると思います．まぁ，自分の使っているクラスファイルを少し編集したいとか，閲覧したいときは kpsewhich というプログラムを使って

```
▉ less ‘kpsewhich book.cls‘
```

とすると book.cls というファイルが$texmf 以下の特定のディレクトリから検索され，less がファイルを整形します．

2.1.8　コマンド

LaTeX では原稿を三つのパートに分割することができます．それに伴いいくつかのコマンドは，特定のパートでしか使用できません．

原稿先頭部分（イニシャルコマンド）
\documentclass[〈オプション,...〉]{〈クラス〉}[〈リリース〉]
前書き部分（プリアンブルコマンド）
\begin{document}
本文（ボディ）
\end{document}

この中で\documentclass，document 環境は必須であり，絶対に必要な記述です．

原稿先頭（イニシャル）部分には**イニシャルコマンド**と呼ばれるコマンドを記述することができ，同じように前書き（プリアンブル）部分には**プリアンブルコマンド**や定義などを記述することができます．そして，document 環境によって挟まれた本文部分にはコマンドの定義や組版用のコマンドを記述します．それぞれのコマンドは定められた場所で使うように決められています．ユーザがプリアンブルコマンドを本文で使うことができないように LaTeX の内部で細工が施されています．

ここで言葉の定義をしましょう．コマンド，命令，環境，引数，オプションなどの言葉を混同しがちですが，**この冊子では**以下のように取り決めます．

コマンド バックスラッシュ（Windows の方は円記号）と共に用いられる文字列．

命令 単独で使用するコマンド．引数を取ることができる．

例：\alpha，\maketitle

環境 ‘\begin{⟨なんとか⟩}’ と ‘\end{⟨なんとか⟩}’ で囲まれている領域，またはそれを囲むためのコマンド．引数を取ることができる．

例：\begin{center}**文字列**\end{center}

引数 コマンドに受け渡す文字列．

必須引数 波括弧 ‘{}’ で囲まれた要素．コマンドが必須引数をとるときは必ず受け渡す．

例：\section{**引数**}

任意引数 オプションとも言う．角括弧 ‘[]’ で囲まれた要素．コマンドが任意引数をとるときは任意に受け渡す．

例：\documentclass[**任意引数**]{jbook}

実はコマンドについての説明が不十分なのですが，今は命令と環境があると解釈してください．

2.1.9 括弧について

さて，LaTeX の基本を知ったところで**括弧**についての取決めをしたいと思います．括弧については色々な呼び方があるようですが，誤解を避けるために**この冊子では**以下のように定義します．

かぎ括弧 ‘「」’ 引用や会話文などに使う．

2 **重かぎ括弧** ‘『』’ 書名，引用の中の引用などに使う．

引用符 ‘‘ ’’ シングルクオートとも言う．左側にあるほうを左シングルクオート，右側にあるほうを右シングルクオートと言う．引用に使う．

2 **重引用符** ‘“ ”’ ダブルクオートとも言う．左側にあるほうを左ダブルクオート，右側にあるほうを右ダブルクオートという．長い引用に使う．

丸括弧 ‘()’ 小括弧，パーレンとも言う．語句の補足説明に使う．

波括弧 ‘{}’ 中括弧とも言う．コマンドに対して必須引数を渡すのに使われたり，要素を一つのグループにまとめるために使う．

角括弧‘ **[]** ’　大括弧とも言う．コマンドに対して任意引数を渡すときに使う．

山括弧‘**<>**’　この括弧に囲まれた文字列は何か別の文字列に書き換えられる．例えば，〈ファイル名〉などがあれば，これは任意の文字列 filename.tex，hoge.foo，**ほげ**などに置き換えられる．

ここで引用符と言うのが登場しましたが，欧文の引用符はシングルクオート（‘’）であり，和文の引用符はかぎ括弧（「」）となります．二つを区別するために欧文用のものを**シングルクオート**，和文のものを**かぎ括弧**と言うことにします．文中に出てくる引用符という言葉はそのどちらも示すことになります．

2.2　LaTeX に関わるファイル形式

タイプセット時に作成される中途ファイル以外にも LaTeX では多くのファイル形式が存在することを経験するでしょう．一般にファイル形式は拡張子によって種類を識別します．

〈ファイル名〉**. 拡張子**

のようにピリオドの後の文字で区別されます．

パッケージをインストールするときに見かけるものは以下の通りです．

.dtx　パッケージ化されたマクロ．複数のクラス 〈クラス *1*〉.cls，〈クラス *2*〉.cls,... 〈クラス *n*〉.cls が 〈クラス〉.dtx 中にまとまっていることも多い．または 〈マクロ〉.sty が複数まとまっているときもある．

.ins　パッケージ化されたマクロを取り出すためのファイル．〈*classes*〉.dtx とともに配布されている．

.sty　便利な機能をうまくまとめたもの．**マクロ**，**マクロパッケージ**，**パッケージ**，**スタイルファイル**（ちと古い言い方）とも言う．

.cls　原稿の書式を決定するファイル．**クラス**，**クラスファイル**，**文書クラスファイル**，**ドキュメントクラスファイル**とも言う．

.clo　クラスのオプションに応じた設定を記述したファイル．

.fd　フォントの属性を定義したファイル．ユーザが意識して使うことはない．

原稿を作成するときに見かけるものは以下の通りです．

.tex　LaTeX が処理を受け付ける原稿．**ソース**，**ソースファイル**とも言う．

.sty　上記参照．

.cls　上記参照．

.clo　上記参照．

.bib　文献成形プログラム BibTeX が処理できる参考文献ファイル．**参考文献データベース**と言う．

.ist　索引の書式を決めるファイル．**索引スタイル**と言う．

.bst　参考文献の表示形式を決めるもの．**参考文献スタイル**と言う．

`.eps` Adobe 社が開発したページ記述言語 PostScript で書かれたファイル．主にベクトル画像などに使われる．

原稿をタイプセットした後に見かけるものは以下の通りです．これらは全て中途ファイルであり，LaTeX が原稿を完成させるためにに必要なものです．

`.log` LaTeX の組版結果の詳細情報．**ログファイル**と言う．
`.aux` 相互参照などの情報が書かれたファイル．1 度目以降の処理に必要とされる．
`.dvi` 原稿を LaTeX でタイプセットした後に作成される印刷結果に限りなく近いファイル．このファイルをプレビューしたり，または他のデバイスドライバによって別の形式に変換できる．
`.toc` 「目次」を出力するための目次情報が書き出されたファイル．
`.lof` 「図目次」を出力するための図目次情報が書き出されたファイル．
`.lot` 「表目次」を出力するための表目次情報が書き出されたファイル．
`.bbl` BibTeX によって並べ替えをした後の参考文献リスト．`thebibliography` 環境を用いて記述されている．
`.blg` BibTeX の実行結果が出力されるログファイル．
`.idx` 並べ替えられる前の索引の語句が書き出されたファイル．MakeIndex，mendex などのプログラムで並べ替えをする．
`.ind` makeindex などによって並べ替えられた索引ファイル．`theindex` 環境を用いて記述されている．
`.ilg` makeindex などを実行したときの処理結果が出力されるログファイル．

2.3 コマンドの基本

LaTeX では便利なコマンドがあらかじめ用意されています．それらをどのように用いるか，また必要な機能がないときはどうすれば良いのかを説明します．

2.3.1 原稿の先頭でのコマンド

少し前置きが長くなりましたが LaTeX の原稿の構造をもう一度見ておきましょう．

```
\documentclass[⟨オプション,...⟩]{⟨クラス⟩}[⟨リリース⟩]
前書き部分（プリアンブルコマンド）
\begin{document}
本文（ボディ）
\end{document}
```

さて，原稿の先頭部分，`\documentclass` が始まる前のイニシャルコマンドには `filecontents` 環境が使えます．

```
\begin{filecontents}{⟨ファイル名⟩}
⟨内容⟩
\end{filecontents}
```

この filecontents 環境が持つ機能ですが，指定した ⟨ファイル名⟩ に ⟨内容⟩ を書き出してくれます．例えば原稿を一つのファイルとしてしか配布できない場合に EPS 画像などを同時に含めるならば，この部分に EPS 画像のソースを記述します．ただしこの環境は書き出すファイルの先頭にコメントを挿入しますので，アスタリスク '*' を付けて

```
\begin{filecontents*}{<ファイル名>}
%!PS-Adobe-2.0 なんとかかんとか...
\end{filecontents*}
```

とすると自動的に付加される余分なコメントが入りません．

2.3.2　プリアンブルでのコマンド

原稿の先頭には filecontents 環境が使えることは分かりました．次に書くべきコマンドは何でしょうか．それは，おそらく人生で何度も目にするだろう\documentclass 命令です．

```
\documentclass[⟨オプション,...⟩]{⟨クラス名⟩}[⟨リリース⟩]
```

この命令は「これから文書で使う命令の定義や前書きを書きますよぉ，ほげほげ．」という意味合いを持っており，この命令を書いた後は原稿の前書き部分（プリアンブル）として解釈されます．

⟨クラス名⟩ には 3.21.1 節で紹介するものが使えます．⟨オプション⟩ にはそれぞれのクラスが用意している任意引数を渡すことが出来ます．このオプションのことを特に**文書クラスオプション**とか**ドキュメントクラスオプション**と言います．⟨リリース⟩ には自分の使っているクラスファイルがいつ配布されたのかを書きます．

⟨リリース⟩ にはクラスの配布された日付を ⟨YYYY/MM/DD⟩ という書式で記述できます．例えば，2003 年 12 月 31 日に公開された日本語のクラス jarticle ならばおおむね以下のようになります．

```
\documentclass[11pt,a4j]{jarticle}[2003/12/31]
```

もしも，クラスファイルが 2003 年 12 月 31 日以前のもので要求されているバージョンよりも古ければ，LaTeX はタイプセット時に警告（Warning）を出します．

他にも 3.21.3 節で紹介しているようなパッケージを使う場合はプリアンブル部分に\usepackage を使います．

```
\usepackage[⟨オプション,...⟩]{⟨パッケージ名⟩}[⟨リリース⟩]
```

これはプリアンブルのみでしか使えません．\usepackage 命令は\documentclass 命令と同じように，そのパッケージが提供するオプションを指定したり，リリースにはそのパッケージのバージョンを指定できます．例えば，画像ファイルなどを LaTeX で扱いたいと思い，デバイスドライバとして Dvipdfm*x* を使う場合は

```
\usepackage[dvipdfm]{graphicx}[2001/01/01]
```

のように graphicx パッケージを使うことを**プリアンブルで**宣言します.

同じパッケージを 2 度や 3 度以上読み込もうとしても，1 度読み込まれているなら再度読み込もうとしません．パッケージに渡すオプション（リリースを除く）を特に**パッケージオプション**と呼びます.

文書クラスオプションやパッケージオプションのいずれにしても，たいてい「命令」と「必須引数」のあいだの ⟨オプション⟩（任意引数）は**複数個**渡すことができます．例えば

```
\documentclass[10pt,a4paper,twocolumn]{article}
```

のように *10pt*，*a4paper*，*twocolumn* という三つのオプションはコンマ ‘,’ を区切りとして書けば良いのです.

同時に複数のパッケージを使うことも宣言できます．graphicx，amsmath，makeidx などを

```
\usepackage{graphicx,amsmath,makeidx}
```

のように宣言できますがパッケージオプションをそれぞれのパッケージに対して渡すことはできません.

基本的なソースファイルは次のようになります.

```
\documentclass[10pt,a4paper,oneside]{jarticle}
\usepackage[dvipdfm]{graphicx}
\usepackage[dvipdfm,usenames]{color}
\begin{document}
なんとかかんとか.
\end{document}
```

LaTeX 処理を実行した原稿のプリアンブルに

```
\listfiles
```

という`\listfiles` 命令を記述すれば，自分が処理している原稿に何のファイルが使用されているのかを端末と ⟨*filename*⟩`.log` に書き出します．例えば

```
\documentclass{jbook}
\listfiles
\begin{document} hoge \end{document}
```

のようなファイル ⟨*filename*⟩`.tex` が存在し，platex などで LaTeX 処理をしたならば

```
This is pTeX Version 3.141592-p3.1.2 (sjis)(Web2C 7.5.2)
 *File List*
  pldefs.ltx 2000/07/13 v1.2 pLaTeX Kernel
   jy1mc.fd 1997/01/24 v1.3 KANJI font defines
   jy1gt.fd 1997/01/24 v1.3 KANJI font defines
 kinsoku.tex
 plpatch.ltx
   jbook.cls 2001/10/04 v1.3 Standard pLaTeX class
```

```
  jbk10.clo 2001/10/04 v1.3 Standard pLaTeX file
 ************
Output written on filename.dvi (1 page, 216 bytes).
Transcript written on filename.log.
```

のような表示が出るでしょう．ここで少し疑問に思っていただきたいことは，「原稿には jbook を使うことしか宣言していないのに何か別のファイルも一緒に読み込まれている．」ということです．

第 3 章
文章の組版

LaTeX で文書を作成するためには文章の組版に関する約束事を知る必要があります．論理的な文章を書きたいと思ったら，その理論を知る必要があります．この章ではそれらを LaTeX で実現するための基本的な部分を説明します．

3.1 文章の論理構造

簡単な文書を作成するうえで覚えたほうが良い項目を示します．

表　題　文書には必ず表題をつけて**誰**が**いつ**，**何**を作成したのかを示します．
目　次　ページが多い場合には目次をつけて読者に参照しやすいようにします．
見出し　見出しを付けてこれから何について話をするのかを明確にします．
字下げ　段落始めは 1 文字ほど開けます．
段　落　一つの話題について一区切り付いたら段落を分けます．
句読点　文章の中で文の区切り，文の終わりには句読点などの区切り記号を付けます．
注　釈　難解と思われる用語，補足すべき情報があれば注釈として添えます．

これは日本語でもほかの言語でもほぼ共通だと思います．誰かに何かを文書で伝えるときにこれらの要素は必要でしょう．文書の最小構成単位は「単語」です．「文字」から「文」ができ，「段落」ができ「節」ができ，「章」，「部」へとつながっていきます．日本語の場合は表意文字なので最小単位は「1 文字」に相当します．

LaTeX はユーザが約束通りにコマンドを打ち込み文章を練り上げていれば，字下げ，相互参照，図表の配置，目次の作成など様々なことを半自動的に行ってくれます．ここではその基本的な約束を紹介します．

3.2 表題

表題はその文書が何について書かれたものなのかを示すために必要な要素です．通常は「題名」，「作者」，「日付」を書くのが一般的ですからプリアンブルに

```
\title{〈題名〉}
\author{〈作者〉}
\date{〈日付〉}
```

の三つを書き込みます．LaTeX ではプリアンブルに表題の情報を書き込んでも出力まではしませんので\begin{document} の後に

\maketitle

とします．

例を示すと入力が

```
\documentclass{jarticle}
\title{はじめての\LaTeX}
\author{未来 太郎}
\date{2004 年 3 月 30 日}
\begin{document}
\maketitle
{\LaTeX}を使うのはこれが初めてです．
\end{document}
```

であったならば，大体の出力は以下のようになります．

はじめての LaTeX

未来 太郎

2004 年 3 月 30 日

LaTeX を使うのは . . .

▹ **例題 3.1** 著者が複数人いるときは\and を使って区切ります．所属などがあるときは\thanks で欄外に出力します．以下のようなソースを実際に自分で出力してみてください．またその結果も吟味してください．

```
\author{夏目漱石\thanks{ほげほげ研究所 ほげほげ事業部}  \and
        福澤諭吉\thanks{ほげほげ株式会社 ほげほげ研究所}\and
        芥川龍之介\thanks{ほげほげ大学 ほげ学部 ほげ学科}}
```

▹ **例題 3.2** もう少し派手に著者名を紹介するとき，例えば「氏名」，「所属」，「連絡先」の三つを記述したいときは

```
\author{ほげ太郎\\ ほげ大学 ほげ学科\\ name@server.ac.jp}
```

のようにします．このソースを実際に出力しその結果を吟味してください．例題 3.1 の \thanks 命令と\and 命令を使って

```
\author{夏目漱石\thanks{ほげ大学 ほげ学科 name@server.ac.jp}}
```

とするか改行で区切る方法のどちらも試してください．

▸ **問題 3.3** 以下のソースを出力してその結果を吟味してください．

```
\author{夏目漱石 \\ ほげほげ研究所 \\ ほげほげ事業部 \and
        福澤諭吉 \\ ほげほげ株式会社 \\ ほげほげ研究所\and
        芥川龍之介\\ ほげほげ大学 ほげ学部 \\ ほげ学科}
```

この場合は横に著者が並びますが ‘\and’ を取り除いて ‘\\’ に書き換えると出力はどう変わるでしょうか.

3.3 見出し

文書に**見出し**と**目次**がなければ，記事の検索に時間がかかるのは容易に想像できるでしょう．そこで，文書の中には**階層的な見出し**を作成します．またその文書の概略が存在すればその文書に何が書かれているのかがすぐに分かるので，**概要**を付け足すのも効果的です.

3.3.1 見出しの出力

文書の中の一連の段落に何が書かれているのかを分かりやすくするために見出しを記述します．また見出しは同一ページに同じ名前のものが存在しても良いように通し番号をつけて一元的に管理します.

LaTeX での見出しの定義は表 3.1 の通りです．\section などの見出し命令を使って見

表 3.1 LaTeX での見出しの定義

\part[⟨目次用の見出し⟩]{⟨見出し⟩}	部
\chapter[⟨目次用の見出し⟩]{⟨見出し⟩}	章 *
\section[⟨目次用の見出し⟩]{⟨見出し⟩}	節
\subsection[⟨目次用の見出し⟩]{⟨見出し⟩}	項（小節）
\subsubsection[⟨目次用の見出し⟩]{⟨見出し⟩}	目（少々節）
\paragraph[⟨目次用の見出し⟩]{⟨見出し⟩}	段落
\subparagraph[⟨目次用の見出し⟩]{⟨見出し⟩}	小段落

*article や jarticle では定義されていません.

出しを作成します．前後の空白の調節や改ページ，改行，書体の変更などはほぼ自動的に行われ，**通し番号**が付加されます．‘[⟨短縮した見出し⟩]’ という任意引数がありますが，これは見出しが非常に長いときに，それを短縮した文字列を目次に書き出すようにします．別に長いときだけではなく，見出しと目次の文字列を別にしたいときなどにも使えるでしょう．使い方は簡単です．見出しを階層構造的に書き記せば，LaTeX は自動で階層ごとに番号付けをします．例としては次のような通し番号が振られます（正確には出力は異なります）.

```
\chapter{特殊相対性理論}
  \section{歴史的背景}
\chapter{一般相対性理論}
  \section{電気学との関連}
    \subsection{電気の次元数}
```

第 1 章 特殊相対性理論
1.1 歴史的背景
第 2 章 一般相対性理論
2.1 電気学との関連
2.1.1 電気の次元数

3.3.2　見出しの深さ

表 3.2　見出しの階層

見出し	命令	深さ *
部	\part	-1 （0）
章	\chapter	0 （なし）
節	\section	1
小節	\subsection	2
少少節	\subsubsection	3
段落	\paragraph	4
小段落	\subparagraph	4

* 括弧内は (j)article での深さ

文章の論理構造を整理するとき，一つの文書を**項目ごと**に分けることができます．さらにその項目を小項目で分けることもできるわけです．小項目があると文書の構造は**階層的に**なります．項目が分かれていることを区別するために見出しを付けます．見出しを目次としてひとまとめに出力すると，読者は目的の項目を探しやすくなります．

LaTeX ではあらかじめ部，章，節，小節，少少節，段落，小段落という七つの見出し用のコマンドを用意しています．ただし (j)article などで章は用意されていませんし，クラスによって深さが若干違います．

3.4　目次の出力

目次は見出しから読みたい箇所に移動するための見出し一覧です．これは 20 ページ以上の文書には必須でしょう．目次といっても LaTeX では

\tableofcontents （**目次**を出力するための命令）
\listoffigures（**図目次**を出力するための命令）
\listoftables（**表目次**を出力するための命令）

の三つの命令が用意されており，それぞれ出力したい場所に命令を書きます．注意すべきこととして目次を作成するためには最低 2 回のタイプセットをします．その理由を考えるために，以下のようなファイル mokuji.tex を作ります．

```
\documentclass{jarticle}
\begin{document}
\tableofcontents% 目次を出力する命令
\section{ほげ}
\subsection{ほげほげ？}
\section{どれどれ}
\subsection{どれどれどれ}
\end{document}
```

次にこのファイル mokuji.tex をタイプセットすると出力ファイル mokuji.dvi にはただ「目次」と出力されるだけで，実際の目次が出力されていないことに注目してください．端末には

```
No file mokuji.aux.
No file mokuji.toc.
[1] (./mokuji.aux) )
```

というメッセージが表示されます．No File なんとか というのは〈なんとか〉というファイルが存在しないよ，足りないよというメッセージです．ですが，同じディレクトリ（フォルダ）には mokuji.aux も mokuji.toc も存在するようです．どうやらタイプセットする前には存在せず，タイプセット後にこの二つのファイルは作成されたようです．この二つのファイルを覗いてみましょう．まずは中途ファイル mokuji.aux には

```
\relax
\@writefile{toc}{\contentsline{section}{\numberline{1}ほげ}{1}}
\@writefile{toc}{\contentsline{subsection}{\numberline{1.1}ほげほげ？}{1}}
\@writefile{toc}{\contentsline{section}{\numberline{2}どれどれ}{1}}
\@writefile{toc}{\contentsline{subsection}{\numberline{2.1}
   どれどれどれ}{1}}
```

のように，ちょっと分かりづらい記述があります．要するに mokuji.toc に見出し関係の記述をどうすべきかを書いているようです．さらに目次用の中途ファイル mokuji.toc には

```
\contentsline{section}{\numberline{1}ほげ}{1}
\contentsline{subsection}{\numberline{1.1}ほげほげ？}{1}
\contentsline{section}{\numberline{2}どれどれ}{1}
\contentsline{subsection}{\numberline{2.1}どれどれどれ}{1}
```

という先ほどの mokuji.aux とほとんど同じ記述があります．これを踏まえてもう 1 度 mokuji.tex をタイプセットします．すると端末には

```
(d:/usr/local/share/texmf/ptex/platex/js/jsarticle.cls
Document Class: jsarticle 2004/02/25 okumura
) (./mokuji.aux) (./mokuji.toc) [1] (./mokuji.aux) )
```

のようなメッセージが表示されます．どうやら mokuji.aux も mokuji.toc も存在し，それらを適切に処理してくれたようです．出力ファイル mokuji.dvi の先頭には始め出力されなかった「目次」があることでしょう．

目次

3.4.1　目次を出力する深さ

目次をどの階層まで出力するかはカウンタ tocdepth の値を表 3.2 に従って変更します．jsbook などで章（\chapter）まで出力したいならば

```
\secounter{tocdepth}{0}
```

のようにします．(j)book と (j)report の標準は 2，(j)article ならば 3 です．jsbook は 1 になっています．

3.4.2　見出しの番号付けの深さ

見出しの通し番号はカウンタ secnumdepth によってどの階層まで出力するかを決められます．secnumdepth の値は表 3.2 に従って変更します．少節（\subsection）までに番号を付けるようにするには

```
\setcounter{secnumdepth}{2}
```

のようにします．これは目次側にも影響します．

3.5　概要の出力

文書の概略が存在すればその文書に何が書かれているのかが大まかに分かるので概要を書くのが良いでしょう．「概要」は「はしがき」とも呼ばれ，文書クラスによって出力方法が違います．(j)article 系ならば abstract 環境を使います．この abstract 環境は\maketitle 命令と関わりがあるので概要を出力するためには\maketitle 命令の後に書きます．

```
\maketitle
\begin{abstract}
⟨文書の概要⟩
\end{abstract}
```

次に (j)report の場合ですが概要専用の環境は用意されていません．そこで概要を章立てすると良いので\chapter*命令を使います．このとき\chapter 命令にアスタリスク ‘*’ をつけると目次に見出しを書き出さず，章番号を付け足しません．使用例は

```
\chapter*{概要}\addcontentsline{toc}{chapter}{概要}
```

と記述してから概要の文章を書きます．標準の文書クラスでは概要専用のコマンドは定義されていません．用途は異なりますが奥村晴彦氏の jsbook には abstract 環境が定義されています．これは各章の始めに「この章について」のようなまえがきを書くときに使われます．

最後に (j)book 場合ですが，これは\frontmatter が宣言されているときに\chapter 命令を使うと良いと思います．具体的には

```
\begin{document}
\frontmatter% 前付け
\chapter{まえがき}
ここに概要やまえがきを書きます．
\mainmatter% 本文
\chapter{序論}
```

とすると目次に概要を書き出しますが番号を付け足しません．

3.6 段落と字下げ

文章で段落をはじめようと思えばまずは字下げ（インデント）をします．小学校の作文でも必ず作文用紙では次の段落に移るときは 1 文字開けたと思います．この字下げの作業を LaTeX は半自動で行います．使い方は

```
ほげはほげほげでほげですから，ほげほげしてからほげます．
それからほげはほげほげですからほげました．ほげほげは大変
ほげでしたからほげまでほげます．

ほげですから，ほげはほげほげしました．
```

のように 1 行空けて入力すれば

> ほげはほげほげでほげですから，ほげほげしてからほげます．それからほげはほげほげですからほげました．ほげほげは大変ほげでしたからほげまでほげます．
> ほげですから，ほげはほげほげしました．

として自動的に段落を作成してくれます．明示的に\par 命令で段落の終了を知らせることができ，以下のようにも書けます．

```
ほげはほげほげでほげですから，ほげほげしてからほげます．\par
それからほげはほげほげですからほげました．ほげほげは大変
ほげでしたからほげまでほげます．
ほげですから，ほげはほげほげしました．\par
```

以上のように LaTeX はワープロソフトとは違い，原稿中の一つの改行が出力と対応していないのがお分かりになるでしょう．LaTeX では改行すべき位置を自動で計算しているのです．また，このときの字下げの量は\parindent という長さ変数で指定されているので

```
\parindent=3em
```

とすると約 3 文字分の字下げを段落の始めで行うことができます．

3.6.1 行頭の字下げの有無

段落の開始には字下げをすべきなのですが，何らかの理由により字下げを抑制したいときがあります．字下げの有無に関しては\indent と\noindent 命令が使えます．

\indent 可能ならば字下げをします.
\noindent 可能ならば字下げをしません.

jreport などのクラスファイルではこのような命令を使っても行頭の字下げができないときがあります．その場合は indentfirst パッケージを読み込みます．

```
%\section{字下げ}
\indent ほげは\indent ほげですから，ほげ．\par
\noindent ほげほげでした．
```

ほげはほげですから，ほげ.
ほげほげでした.

どうも最近のビジネス文書などでは字下げをせずに段落と段落に空きを入れて段落の終わりと段落の始まりを示す傾向があります．これを LaTeX で行うためには段落と段落の空きを調節する\parskip という可変の長さ変数を調節して

```
\parindent=0pt% 字下げ
\parskip=10pt plus 0pt minus 0pt
```

とします．実際にこのような設定にすれば分かりますが，ありとあらゆる部分に空きを入れてくれるので，別の方法を考えなければなりません．

3.6.2　段落の字下げ

文を引用している場合はそれが引用であることを明確にするために，段落全体を字下げする習慣があります．これには quote 環境や quotation 環境が使えます．ただしこの場合は自分で字下げ幅を設定できません．quote 環境の中で使われている list 環境を自分でカスタマイズするとうまく行きます．もう少し簡単に行うには indent パッケージの indentation 環境を使います．indentation 環境の**一つ目**の必須引数には**左側の**字下げ，**二つ目**には**右側の**字下げを指定します．

```
ここは普通の文章領域です．不必要に字下げを
調整するのは好ましいことではありません．
\begin{indentation}{3zw}{3zw}
左側の字下げは全角 3 文字分，右側の字下げ
も全角 3 文字分ありますか？
\end{indentation}
\begin{indentation}{0zw}{5zw}
左側の字下げはなし，右側の字下げ
は全角５文字分ありますか？
\end{indentation}
ここも普通の文章領域です．
```

ここは普通の文章領域です．不必要に字下げを調整するのは好ましいことではありません.

　　左側の字下げは全角 3 文字分，右側の字下げも全角 3 文字分ありますか？

左側の字下げはなし，右側の字下げは全角５文字分ありますか？
ここも普通の文章領域です.

3.6.3　ダブルスペース

またダブルスペースといって**行送り**を倍にするということも迫られる場合があるようです．これは行送りの量を決める長さ変数\baselinestretch を

```
\renewcommand{\baselinestretch}{2}
```

とする方法があります．他にも doublespace パッケージを

```
\usepackage{doublespace}
```

としてプリアンブルで読み込むと上記の方法よりもうまくダブルスペースを実現できます．

3.7 長さの単位

先ほどは何らかのパラメータに数値を代入する

```
\parindent=0pt
```

のような記述がありました．これにはポイント ‘pt’ という単位が使われています．LaTeX において使用できる長さの単位（表 3.3）は色々あります．ポイントは絶対的な長さではないのでクラスファイルによって変わったりプログラムによっても若干の違いがあります．奥村晴彦氏の jsclasses ではクラスオプションに *10pt* 以外のフォントサイズ指定がされている場合は版面の拡大縮小を使っていますので単位がずれます．これには各単位に ‘true’ を付けて長さを指定します．例えば ‘cm’ ならば ‘truecm’ のようにします．

表 3.3 LaTeX で使用できる主な単位

単位	読み	補足（数値は概算）
in	インチ	1 in=25.4 mm=72.27 pt
cm	センチメートル	1 cm=10 mm=28.3 pt
mm	ミリメートル	1 mm=2.83 pt
pt	ポイント	1 pt=0.35 mm
em	M の字の幅と同じ．	使用中のフォントに依存
ex	x の字の高さと同じ．	使用中のフォントに依存
zw	日本語の一文字の幅．	使用中のフォントに依存

3.8 句読点

句読点についてですが，縦書きにするか横書きにするかで違います．縦書きのときは何も迷わずに全角の読点「,」と句点「.」を用います．横書きのときは句読点は**半角**のカンマとピリオドを使うのがベストだと思います．その場合は句読点の後には半角のスペースを入れてください．横書きで全角の句読点を使うときは

> 今日のお弁当のふたには “I wish I could.” という文句が書かれていた。

のように二つの文書に異なる句読点が混在することになります．なるべく句読点は統一したほうが気持ちが良いでしょう．ただし，

> 今日のお弁当のふたには「I wish I could。」という文句が書かれていた。

などとするのはとっても気持ち悪いのでやめたほうが無難です．横書きのときは文章の内容と相談して句読点を決めてください．

句読点が少し気持ち悪く見える例です．

地球は，青かった． 10\,m，高かった．　　地球は，青かった． 10 m，高かった．

どのようにするのが良いかはご自分で判断していただければ良いのですが，この冊子でも一応の見解を出しておきます．まずは欧文だけの場合を考えます．欧文のみの文書では以下の規則に従うだけで良いでしょう．

- 句読点は半角のコンマ「,」とピリオド「.」を使う．
- 単語の引用はシングルクオート「‘’」，1 文の引用はダブルクオート「‘‘’’」を使う．

和文のみの場合は

- 句読点は読点‘，’と句点‘．’を使う．
- 単語の引用はかぎ括弧‘「」’，文の引用はダブルクオート‘『』’を使う．

となるでしょう．ですが和文と欧文が混在する文書ではどのように取り決めたら良いのでしょうか．本来ならばまず句読点のつけ方も引用の仕方も一つの文書の中で統一するが規則です．そのため和文と欧文が混在する場合は欧文の規則に従うということです．

ですが，そうすると別の問題が出てきます．日本語組版で「句読点は 1 文字分の幅を割り当てる」という規則に反することがあります．これには全角のコンマ「，」とピリオド「．」を使うようにすると我慢できます．

3.9　注釈

注釈とは文章の中で出てきた注意すべき語句を説明するために付けるものです．注釈は読者が読まなくても良い，本文とは関係のない情報を示すために使われます．LaTeX では 2 種類の注釈を出力できます．一つはページ下部に出力する**脚注**（\footnote），もう一つは注釈語の横に出力する**傍注**（\marginpar）です．紙面の下端に表示される脚注には\footnote 命令を使います．

このように**傍注**が出力されます

```
注釈語\footnote{⟨注釈内容⟩}
```

この命令を使用すると LaTeX は組版時に自動的に\footnote で通し番号を付けます[*1]．脚注の出力は使用しているクラスファイルによって違うので確認してみると良いでしょう．

[*1] このように注釈が文章の頁の下端に出力されます

ラプラス変換やフーリエ変換\footnote{Fourier Translation}は通常理工系の大学ならば必修で... と思われる.

ラプラス変換やフーリエ変換[a]は通常理工系の大学ならば必修で... と思われる.

[a] Fourier Translation

3.10 文字の強調

最近のワープロ文書では重要な文字列に下線を引いて強調を表現しているようです．論文や書籍では**欧文**を**イタリック体**，**和文**の場合は**ゴシック体**にするほうが良いと思われます．LaTeX では文字列の強調のために\emph が使えます．もしかしたら使用しているクラスファイルによっては和文がゴシック体にならないかもしれません．

欧文の強調には\emph{English Emphasize}として，和文の強調は\emph{文字列の強調}のようにします.

欧文の強調には *English Emphasize* として，和文の強調は**文字列の強調**のようにします.

場合によっては\em コマンドが使えます．これは**宣言型コマンド**（6.2.2 節）と呼ばれるもので，広範囲な論理強調に使うことが出来ます．ただしこのコマンドを使うならば**イタリック補正**を挿入する場合があります．イタリック補正とはイタリック体の文字の直後にローマン体などの別のフォントが来た場合に挿入すべき空白のことです．LaTeX が用意している\emph や\textit などはこのイタリック補正が適切に挿入されるようになっています．宣言型の強調コマンド\em を使う場合は自分で挿入しますから\/命令を使うことになります．

```
\emph{Future University-Hakodate} is
interesting.\par
{\em FUN\/} is funky?\par
{\em FUN} is not funky?
```

Future University-Hakodate is interesting.
FUN is funky?
FUN is not funky?

3.11 そのまま出力できない記号

LaTeX ではいくつかの半角の記号を直接出力することができません．これは予約文字と呼ばれるもので，‘\’ に似た命令の一種だと思ってください．出力できない記号は

```
\ { } $ & # ^ _ ~ % < > |
```

の 13 個であり，それらの記号を出力するには 2.1.3 節 を参照してください.

3.11.1 特殊記号

北欧文字などを出力するための特殊文字も用意されており，それらを出力するには表 3.4 の命令を用います．表中のアスタリスク ‘*’ 付きの記号は fontenc パッケージを ‘*T1*’ というオプション付きで読み込むと出力できます．具体的には次のようにします.

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

表 3.4　特殊記号

å	\aa	ø	\o	†	\dag	Đ	\DJ *	«	\guillemotleft *
Å	\AA	Ø	\O	‡	\ddag	ŋ	\ng *	»	\guillemotright *
æ	\ae	ı	\i	£	\pounds	Ŋ	\NG *	‹	\guilsinglleft *
Æ	\AE	ȷ	\j	¡	!‘	þ	\th *	›	\guilsinglright *
œ	\oe	ß	\ss	¿	?‘	Þ	\TH *	„	\quotedblbase *
Œ	\OE	SS	\SS	ð	\dh *			,	\quotesinglbase *
ł	\l	§	\S	Đ	\DH *			"	\textquotedbl *
Ł	\L	¶	\P	đ	\dj *				

表 3.5　アクセント記号

ü	\"u	ā	\={a}	à	\‘{a}	ạ	\d{a}	ǎ	\v{a}	ů	\r{o}
é	\’{e}	ő	\H{a}	a͡	\b{a}	ç	\k{c}	ñ	\~{n}		
ȧ	\.{a}	ô	\^{o}	o̧	\c{o}	ĭ	\u{ı}	o͡o	\t{oo}		

アクセント類を出力するには表 3.5 の命令を使います．‘i’ や ‘j’ は表 3.4 中の ‘\i’ と ‘\j’ を使います．

```
J\"org {mu\ss} ein Gel\"ande f\"ur
   seine Fabrik erwerben.
```

Jörg muß ein Gelände für seine Fabrik erwerben.

3.11.2　日本の通貨記号

円記号 ‘¥’ を標準では出力することができません．何らかのパッケージを読み込みます．okumacro や ascmac，textcomp などの既存のマクロではたいてい\yen か\textyen などの命令で定義されています．そもそもなぜ円記号が LaTeX では出力できないのかをここではっきりとさせておきましょう．

Windows ユーザで LaTeX を使い始めた人がまず疑問に思うことの一つに円記号があります．普段自分の使っているエディッタやワードプロセッサでは円記号は問題なく出力されているのに，LaTeX でタイプセットするとなぜか「バックスラッシュ」といわれる記号に置き換わってしまいます．「これが円記号の本当の姿なんだよ」とやさしく教えてあげても，本人の慣れのほうが優先されるのか納得してくれません．けっこう昔の話なのですが ASCII コードの特定の領域にはそれぞれの国が独自の記号を割り当てても良いという流れがあったので，日本は ‘0x80’ 以降のコードに半角カタカナなどを割り当てたり，‘0x7E’ の**バックスラッシュ**を**円記号**に割り当てました．キーボードに円記号があるけれど実際に入力して LaTeX で出力するとバックスラッシュになるという問題が起きています．そうすると円記号が打てないという問題になるわけです．半角カタカナが割り当てられている証拠を見るためには Unix 系 OS の方は ascii というプログラムを端末から

```
▌ascii
```

とすると 8 ビット中の領域に何の記号が割り当てられているのかが分かります．LaTeX でもこの問題に悩まされており，キーボードから円記号 ‘¥’ を入力しても思いっきりバックスラッシュ ‘\’ が出力されます．

> 「あらまぁ，大変よ！子供がぐれちゃって円からバックスラッシュになっちゃったわ！！」

といった具合で，子供の不良行為になれていない親のように取り乱してしまう人も続出してしまいます．そのような子供は

> 「俺は Unix 使いだから気にならねぇや，ほげほげ.」

と言うのですが，いざ LaTeX でタイプセットしていると

> 「なんだこのスラッシュのへんてこりんなやつは，こんなんじゃ書類は受け取らん！出直して来い！！」

と相手の都合でバックスラッシュではどうにもならないときが，しばしばあります．

そのため多くの方々が LaTeX で円記号を出力する方法を模索しておられました．いくつも方法があるのですが簡単な例として次のように定義します．

```
\newcommand{\yen}{Y\llap=}
```

この方法では ‘¥’ のような出力になります．ただし，この方法だとずれるときがありますので

```
\newcommand{\yen}{\ooalign{Y\crcr\hss=\hss}}
```

ということも考えられます．以上の二つの例からも分かる通り半角英字の ‘Y’ と数学のイコール ‘=’ の二つを重ね合わせてなんとなく円記号を作っています．そのためちょっとイコールのほうが下に行き過ぎているので，少しうえに移動します．

```
\newcommand{\yen}{\ooalign{Y\crcr\raise.1ex\hbox{\hss=\hss}}}
```

以上のような苦労をしなくとも textcomp と fontenc を使う手もあります．

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

という設定ではビットマップフォントが埋め込まれるかもしれませんのでお好みで

```
\usepackage{txfonts}% mathpazo, pxfonts などでも良い
```

の 1 行を追加しておき原稿中で ‘\textyen’ と使うと良いでしょう．

3.12　原稿中での空白の扱い

LaTeX では半角スペースとタブはどちらもスペースとして扱われます．二つ以上のスペースが並んでいるときは一つのスペースとして扱われます．また，一つだけの改行もスペースとして扱われます．改行が二つ連続しているとそれを段落の区切りと判断します．

```
半角の空白 はこのように　2　つ以上
あっても　　一つとみなされます．
```

半角の空白 はこのように 2 つ以上あっても 一つとみなされます．

3.13　コメントの挿入

ファイルのある行のどこからでも % があるとそれ以降を**コメント**して扱います．行頭に ‘\%’ を置けばそこから行末まですべてが**コメントアウト**されます．複数行のコメントを挿入したいときは comment 環境を使います．これを使用するためには Victor Eijkhout 氏による comment パッケージを読み込みます．

```
%\usepackage{comment}
ここは出力されますが % ここはされない．
\begin{comment}
この環境の中もコメントになるので
\end{comment}
出力されませんか？
```

ここは出力されますが出力されませんか？

3.14　べた書き

テキストをそのまま出力するときがあると思います．たとえばプログラムリストを載せたいときは特殊記号などが入り，そのままでは張り込めないでしょう．そのようなときはべた書きが可能です．短い文字列の場合は

```
\verb+文字列+
```

を使います．複数行になるときは verbatim 環境を使います．

```
\begin{verbatim}
ここにべた書きしたい複数行の文字列を挿入します．
\end{verbatim}
```

```
\verb|#include<stdio.h>|はプリプロセッ
サとよばれ...
\begin{verbatim}
int main( void ){
   int hoge = 0x7E;
   printf("%c\n",hoge);
}
\end{verbatim}
```

#include<stdio.h>はプリプロセッサとよばれ...

```
int main( void ){
   int hoge = 0x7E;
   printf("%c\n",hoge);
}
```

\verb 命令や verbatim 環境にはアスタリスクを付けることができます．さらに\verb 命令の場合は 〈文字列〉 を括る区切り記号はアスタリスク以外ならば何でも良いことになっています．

```
\verb|134|, \verb+456+, \verb9|()|9.\par
\verb*|1 3 5|, \verb*9ok? ok?9.\par
\begin{verbatim*}
int main( void ){
\end{verbatim*}
```

```
134, 456, |()|.
1␣3␣5, ok?␣ok?.

int␣main(␣void␣){
```

3.15　引用や文の区切り

文献から一文を引用する，段落を引用するという場面があると思います．まず**単語の引用**にはシングルクオート ‘’ を使います．シングルクオートも 2 種類あり左シングルクオート（‘）はキーボードの Shift を押しながら @ を押し，右シングルクオート（’）は Shift を押しながら 7 を押すと入力できると思います．LaTeX ではこれらを区別して記述します．

文の引用ではダブルクオートを使います．Word などでダブルクオートを挿入すれば自動的に “一文” のように変換されますが LaTeX は不親切で前述のシングルクオートをうまく組み合わせて引用します．これは左シングルクオートを二つと右シングルクオートを二つで括ることになります．他に 1 文用の quote 環境や段落ごと引用するための quotation 環境があります．

```
‘単語はシングルクオートで囲む’
‘‘文はダブルシングルクオートで囲む’’
```

さらに段落ごと引用する場合は段落の左側を字下げして出力します．場合によっては文字を小さくします．行頭の字下げをしないときは quote 環境を，字下げするときは quotation 環境を使います．

```
\begin{quote} 段落引用は quote 環境で囲む\end{quote}
\begin{quotation} 段落引用は quotation 環境で囲む\end{quotation}
```

一般的に以下のような使い方になります．

```
‘単語’ の引用はシングルクオートで‘‘文章の一文’’
の引用は左シングルクオート二つと右シングルクオ
ート二つです．
```

‘単語’ の引用はシングルクオートで “文章の一文” の引用は左シングルクオート二つと右シングルクオート二つです．

```
段落を引用する quote 環境の他にも
  \begin{quote}  行頭の字下げをする
    段落引用の quotation 環境がある．
  \end{quote}
といわれている．
```

段落を引用する quote 環境の他にも

> 行頭の字下げをする段落引用の quotation 環境がある．

といわれている．

和文の引用における**引用符**は全角のかぎ括弧**「」**を使ったほうが美しいのですが，欧文の場合の引用符には半角のクオート ‘’ を使うのが良いと思います．一つの文書に両者が混在するのは余り好ましい状態とは言えません．

‘FUN’ は‘‘Future University-Hakodate’’ の略で，‘FUN’ はほげと密接な関わりがあり渡辺によると「渡辺らによると『ほげはほげである．』という説がある．」と考察している．

‘FUN’ は “Future University-Hakodate” の略で，‘FUN’ はほげと密接な関わりがあり渡辺によると「渡辺らによると『ほげはほげである．』という説がある．」と考察している．

どちらか一方に統一するのが作成者側にも読者にも混乱は少ないでしょう．LaTeX で引用というものをもう少し効率良く行うには

```
\newcommand{\qu}[1]{‘#1’}%    単語の欧文引用
\newcommand{\qq}[1]{‘‘#1’’}% 1 文の欧文引用
\newcommand{\yo}[1]{「#1」}% 単語の和文引用
\newcommand{\yy}[1]{『#1』}% 1 文の和文引用
```

のような命令を定義しておけば，後から引用符を統一できます．上記の\yo と\yy を

```
\newcommand{\yo}[1]{‘#1’}% 単語の和文引用
\newcommand{\yy}[1]{‘‘#1’’}% 1 文の和文引用
```

と変更するだけで文章中の引用符を一括して変更できるわけです．

それは渡辺らによれば\yo{デカルトの名言に \qu{I think, therefore I am.}がある．}という調査結果が存在する．

それは渡辺らによれば「デカルトの名言に ‘I think, therefore I am.’ がある．」という調査結果が存在する．

3.15.1　書籍名や雑誌名の引用

書籍名や雑誌名を引用する場合はその名前を**強調**することになっています．強調といっても欧文の場合は\emph 命令を使ってイタリック体（*Italic shape*）にします．和文の書籍名を引用する場合は 2 重かぎ括弧（『』）が使われています．

\emph{⟨article’s Name⟩}（欧文の場合）
『雑誌名』（和文の場合）

以上のような方法を使って何か別の文書を示す場合はその文書名を強調表示します．

渡辺が 2004 年に\emph{Natural}に投稿した論文『それが私の生きる道』には「人生そんなものだよ．」という名言が書かれている．

渡辺が 2004 年に *Natural* に投稿した論文『それが私の生きる道』には「人生そんなものだよ．」という名言が書かれている．

このような他の文書の引用には新たに欧文用の引用命令\yousyo や，和文用の\wasyo などを作るとあとで統一したいときには便利でしょう．

```
\newcommand{\yousyo}[1]{\emph{#1}}% 欧文
\newcommand{\wasyo}[1]{『#1』}% 和文
```

```
渡辺が 2004 年に\yousyo{Natural}に投稿した論文
\wasyo{それが私の生きる道}には「人生そんなもの
だよ.」という名言が書かれている.
```

渡辺が 2004 年に *Natural* に投稿した論文『それが私の生きる道』には「人生そんなものだよ.」という名言が書かれている.

3.15.2 ダッシュ

ダッシュには和文と欧文のものを併せると 4 種類ほどあります．ひとまとめにしたい単語の区切りや，文の中断などに使います.

en-dash '–' 数値の範囲などを表す.
em-dash '—' 文の中断を表す.
全角ダーシ '—' 欧文の全角ダッシュに近い意味を表しますが，若干高さが違います.
倍角ダーシ '——' 和文での文の中断などを表す.

さらにダッシュに似たものにハイフンとマイナスがあります.

ハイフン '-' 欧文で単語の途中にハイフネーションとして挿入される.
マイナス '−' 数学記号で負の数値を表す.

以上の記号を混同することなく正しく使うのが好ましいです．倍角ダーシを出力するためには okumacro パッケージを読み込みます．出力方法は表 3.6 の通りです.

表 3.6 ダッシュなど

記号の種類	出力	入力・命令	
en-dash	–	`--`	ハイフンを二つ
em-dash	—	`---`	ハイフンを三つ
全角ダーシ	―	―	全角のダッシュ
倍角ダーシ	——	`\――`	'\' と全角ダーシ二つ
ハイフン	-	-	そのまま
マイナス	−	`$-$`	数式中でハイフン一つ

```
``When I was a young---a little dog---I
could read about 100--200 books in a day.
This is a just fairy-tale.''
```

“When I was a young—a little dog—I could read about 100–200 books in a day. This is a just fairy-tale.”

通常ハイフンやダッシュの両隣には空白を入れません．ハイフンによって単語を一塊にしている語句は，ハイフンの途中で改行してはいけません．これは通常の 1 単語のハイフネーションと重複する可能性があるからです.

```
{\TeX}の\mbox{for-each}文は Perl における
\mbox{foreach}文とは性質が異なるた
め，\mbox{X-ray}の影響を受けた
Future \mbox{University-Hakodate}は
\mbox{if-then}文を使う傾向にある.
```

TEX の for-each 文は Perl における foreach 文とは性質が異なるため，X-ray の影響を受けた Future University-Hakodate は if-then 文を使う傾向にある.

3.15.3　ハイフネーション

行末に長い単語がある場合にその単語のハイフネーションにあった改行位置を見つけて LaTeX は改行を行います．ユーザーが明示的に改行位置を指定することもできます．

\hyphenation{〈複数の語句〉}
\-（位置指定）

\-は文章中で直接 ‘Ghost\-script’ のように使います．\hyphenation の場合はプリアンブルなどに

```
\hyphenation{Ghost-script Post-Script}
```

のような記述をします．

```
George Johnson and George Brahms always say
supercalifragilisticexpialidocious, they
always say super\-cali\-fragilistic\-expi%
ali\-docious, and Robert sometimes say
\mbox{supercalifragilisticexpialidocious}.
```

George Johnson and George Brahms always say supercalifragilisticexpialidocious, they always say supercalifragilisticexpialidocious, and Robert sometimes say supercalifragilisticexpialidocious.

‘super なんちゃら’ の一つ目，二つ目では改行の位置が違います．\mbox 命令を使うと改行を許しません．ハイフネーションを

```
\hyphenation{su-per-cali-frag-ilis-tic-ex-pi-ali-doci-ous}
```

とするとか

```
\hyphenation{super-cali-fragilistic-expiali-docious}
```

とするかは，言語によってもハイフネーションのパターンが違うので注意が必要です．

3.15.4　改行

改行はバックスラッシュ ‘\’（Windows などでは円 ‘¥’）を二つ並べて ‘\\’ のようにすれば入れることが可能ですが，文章の中に改行を入れるときは慎重に挿入しなければいけません．できることならばユーザ側の強制的な改行は挿入しないほうが良いでしょう．同じ段落とある文字列を区別したいときは改行ではなく引用（3.15 節参照）を使うのが良いでしょう．

*[〈長さ〉]
\newline
\par

任意引数に改行を行うときの縦の長さを指定できます．ページの先頭での改行を行うことはできません．アスタリスクを付けると改行直後にページを改めることを禁止します．\newline は ‘\bs\’ とほぼ同時の命令です．\par は改行ではなく**改段落**，すなわち段の終わりを示します．その直後の文字列は字下げされます．

```
改行は\\ のように
バックスラッシュを二つ続けて書くと
```

改行は\verb|\\| **のように\\バックスラッシュ**
を二つ続けて書くと\\[1cm] **ユーザによる強制的**
な改行が挿入されます．\par
この文章は新しい段落から組まれ\newline
字下げされる場合があります．

改行は\\ のように
バックスラッシュを二つ続けて書くと

ユーザによる強制的な改行が挿入されます．
この文章は新しい段落から組まれ
字下げされる場合があります．

3.16 空白について

空白は要素と要素を区切るために使われます．**空き**の広さによって意味が違います．正しい量の空白を挿入しなければ意味が変わってしまいます．

3.16.1 文章の中の空白

まず一つの段落内のおける空白の種類を考えてみましょう．日本語の場合はある文字とそれに隣接する文字のあいだに挿入される**文字間空白**というものが存在します．漢字と漢字がぎゅうぎゅうに詰められていては，非常に読みづらいでしょう．この処理は通常日本語 TeX が自動的に行います．欧文でもこれは知らないあいだに変更されています．例えば**合字**や**字詰め**などと呼ばれるものがあります．以下の入出力を見比べてください．

```
Fifth files were found in a folder and
were shuffled by anyone.\par
Fifth f{}iles were found in a folder and
were shuf{}f{}led by anyone.
```

Fifth files were found in a folder and were shuffled by anyone.
Fifth files were found in a folder and were shuffled by anyone.

ここでは ‘fl’ や ‘ffl’ などがその例です．

欧文の場合，単語と単語のあいだに空白を挿入します．これを**単語間空白**とか**単語間スペース**と呼びます．これは人間が意図的に単語の区切りとして ‘My␣name␣is␣Thor.’ のように挿入します．

さらに文と文とを区切るための**文間空白**があります．これは文の終わりを示すもので，単語間空白や文字間空白よりも広い空白になります．LaTeX では

- ピリオドの前の文字が**大文字**ならば**単語間**空白を挿入する．
- ピリオドの前の文字が**小文字**ならば**文間**空白を挿入する．

という二つのルールしか持っていません．

そこで問題になるのが大文字で終わる単語や小文字を含む文です．

```
I want to be a Mr. Right and go to
 N.Y. I wish I could.
```

I want to be a Mr. Right and go to N.Y. I wish I could.

分かりづらいですが ‘Mr.’ と ‘Right’ のあいだの空白のほうが ‘N.Y.’ と ‘I’ のあいだの空白よりも若干広いことが分かるでしょう．これらを正しい空白にするためには人間が明示

的に二つの命令を使います．単語間空白を挿入するためには\␣命令を，文間空白には\@命令を使います．

```
I want to be a Mr. Right and go to
N.Y. Let me do.\par
I want to be a Mr.\ Right and go to
N.Y\@. Let me do.
```

I want to be a Mr. Right and go to N.Y. Let me do.
I want to be a Mr. Right and go to N.Y. Let me do.

そして行と行のあいだの**行間空白**がありますし，段落と段落のあいだの**段落間空白**もあります．これらは LaTeX が最適な空白の量を調節してくれているので，普段は気にすることはないでしょう．

最後に文章における空白をまとめると

文字間空白　文字間に挿入される空白．
単語間空白　単語間に挿入される空白．\␣命令で明示的に挿入できる．
文間空白　文間に挿入される空白．\@命令で明示的に挿入できる．
行間空白　行間に挿入される空白．
段落間空白　段落間に挿入される空白．\par 命令で明示的に段落の終了を告げることができる．

の五つがあるということです．

3.16.2　その他注意すること

それらが並んでいることで一つの意味を持つ単語間には改行を入れないようにします．例えば人名やページ番号，略語などはひとまとめにします．これにはチルダ ‘~’ を使います．

```
Mr.~Sato read page~10 and looked at
figure~3 and table~2 in the book.
```

Mr. Sato read page 10 and looked at figure 3 and table 2 in the book.

引用符が隣り合うときには，引用符と引用符のあいだに小さな空白を挿入します．

```
‘‘‘Hello’ is a fine greeting and I always
say ‘Hello.’’’ \par ‘‘\,‘Hello’ is a fine
greeting and I always say ‘Hello.’\,’’
```

“‘Hello’ is a fine greeting and I always say ‘Hello.’”
“ ‘Hello’ is a fine greeting and I always say ‘Hello.’ ”

3.16.3　和文と欧文のあいだの空白

日本語 TeX では和文と欧文のあいだには空白が挿入されています．これを和文と欧文の**四分空き**と呼びます．四分空きとは全角空白の 4 分 1 の空白のことです．これは和文組版の規則で挿入すべき空白であって，挿入したほうが美しく見えると言われています．以下の例を見ると良く分かるでしょう．例では\mbox で四分空きを無効にしています．

```
日本語と hoge のあいだには四分空きが\par
あると\mbox{}hoge\mbox{}です．
```

日本語と hoge のあいだには四分空きが
あるとhogeです．

普段は何も意識せずに空白が挿入されているので問題ないのですが，原稿の記述の仕方によってその空白が四分空きよりも広くなります．意図的に全角文字と半角文字のあいだに半角空白を挿入するとその部分は四分空きよりも広い**単語間空白**になるときがあります．組版の規則に従うとこの空白は統一すべきですので入力の段階でそれらに気を付けます．例として LaTeX という記号と全角文字の書き方を示します．日本語 TeX は自動的に隣り合う文字が半角文字か全角文字かを判別してくれます．始めは日本語 TeX にその処理を任せて，慣れてきたら自分でその空白を調節すればよいでしょう．実際に入力して試してください．

```
\LaTeX と日本語\TeX      \\
\LaTeX\ と日本語 \TeX    \\
{\LaTeX}と日本語{\TeX}   \\
{\LaTeX} と日本語 {\TeX}\\
```

LaTeX と日本語 TeX
LaTeX と日本語 TeX
LaTeX と日本語 TeX
LaTeX と日本語 TeX

使っている欧文書体の種類によっても違いますし，好みの問題もあるのでこれだと断言できませんが，入力するうえでの作業を考えると三つ目が一番手軽だと思います．ただし，この方法をとるときは欧文同士の空白に注意します．

```
\TeX and  {\LaTeX}are very famous.\\
{\TeX} and {\LaTeX} are very pupular.
```

TeXand LaTeXare very famous.
TeX and LaTeX are very pupular.

入力ファイルでは‘\TeX’の後に空白を挿入しているつもりでも，出力において空白は\TeX に吸収されてしまいます．

3.17 行揃え

日本人は行揃えを良く使いたがるそうです．行揃えには三つの環境と三つの宣言型のコマンドを使うことが出来ます（表 3.7）．環境型のコマンドは広い範囲に使い，宣言型のコマンドは一つの要素や別の環境の中で使うことができます．中央揃えには center 環境です．1 行もしくはそれ以上の文字列，表，図などを中央に寄せることが可能です．行頭や最終行に改行は入れません．右揃えには flushright 環境です．文字列を右寄せにします．左揃えには flushleft 環境です．字下げを行わずに左に寄せます．

表 3.7 揃えの命令と宣言

種類	環境	宣言
左揃え	flushleft	\raggedright
中央揃え	center	\centering
右揃え	flushright	\raggedleft

```
ビジネス文書で大活躍するでしょう．
\begin{flushleft}
段落の字下げを行わずに \\ 文字列を左に揃えます．
\end{flushleft}
```

ビジネス文書で大活躍するでしょう．

段落の字下げを行わずに
文字列を左に揃えます．

```
ビジネス文書で大活躍するでしょう．
\begin{center}
文章を\\ 中央揃えに \\ します．
\end{center}
```

ビジネス文書で大活躍するでしょう．

文章を
中央揃えに
します．

```
ビジネス文書で大活躍するでしょう．
\begin{flushright}
ビジネス文書で活躍中の\\ flushright 環境です．
\end{flushright}
```

ビジネス文書で大活躍するでしょう．

ビジネス文書で活躍中の
flushright 環境です．

さて，この三つの行揃えのコマンドを使ってありがちなビジネス文書の作成をしましょう．

```
\begin{flushright}
        緊急連絡 \\ 2005 年 3 月 31 日
\end{flushright}
\begin{flushleft} 渡辺　徹殿      \end{flushleft}
\begin{flushright}
              ほげほげ会社\\ 人事課
\end{flushright}
\begin{center}  人事異動のお知らせ\end{center}
あなたは 2004 年度から檜山方面に配属されます．
\begin{flushright}          以上  \end{flushright}
```

このようにすると，以下のような出力となります．

緊急連絡
2004 年 3 月 31 日

渡辺 徹殿

ほげほげ会社
人事課

人事異動のお知らせ

あなたは 2004 年度から檜山方面に配属されます．

以上

3.18 箇条書き

箇条書きには

itemize 環境 項目の先頭に記号（ラベル）が付く**記号付き**箇条書き環境．環境の深さによって記号が ‘•，–，∗，·’ のように自動的に変わる．

enumerate 環境 項目の先頭に通し番号が付く**番号付き**箇条書き環境．深さによって通し番号が ‘1，(a)，i，A’ のように自動的に変わる．

description 環境 項目の前に説明を\item の任意引数で指定する**説明付き**箇条書き環境．

という三つの環境を使うことが出来ます．レポートや論文の場合はなるべく箇条書きは避けて，文章による記述が望ましいようです．理解のしやすさを考えれば箇条書きを使うべきでしょう．これらの環境は入れ子にすることが可能です．入れ子に出来る項目の深さは通常四つまでです．itemize 環境の先頭の記号は入れ子にした場合自動的に変更されます．各環境においての項目は\item 命令を使います．itemize においては\item[\#] とすることで先頭のラベルの記号を指定することが可能です．

```
\begin{itemize}
 \item 入れ子にしたい
 \item[*]  入れ子になるはず
  \begin{itemize}
   \item 入れ子です
  \end{itemize}
\end{itemize}
```

- 入れ子にしたい
* 入れ子になるはず
  - 入れ子です

```
\begin{enumerate}
 \item はじめの項目
 \item 次の項目
 \begin{description}
  \item[項目名] 説明
  \item[ほげ] ほげはほげですから
 \end{description}
\end{enumerate}
```

1. はじめの項目
2. 次の項目

   項目名 説明

   ほげ ほげはほげですから

3.19 書体について

文字は意思伝達手段であって長いあいだに洗練された媒体です．怒りの意思を強く込めたいならば人は荒々しく文字を書くでしょうし，優しさを込めたいならば丸みを帯びた書き方になるでしょう．以上のような文字の形を書体と言います．

世の中にはこれらを書体というひとつの枠組みで整理しています．書体は読者に対して何らかのメッセージを分かりやすく伝えるために変更される場合があります．ですから書体を変更するということには必ず意味があるべきなのです．むやみやたらに書体を変更し

ても逆に読者を混乱させます．また自分だけのルールで書体を変更しても読者には何の意味なのかが分かりませんので，一般的に使われている書体に関するルールを守るのもマナーです．

LaTeX はマークアップ型のシステムなのでユーザーが直接書体変更用の命令を使うことは本来ならば必要のないことだと思われます．以下のコマンドは直接使うのではなく新規に環境を定義して用いるのが望ましいでしょう．

3.19.1　文字の大きさの変更

LaTeX においては比較的簡単に文字の大きさを変えることが可能ですが，文字は文書クラスオプションで指定した基準の文字の大きさに応じて変更されます．文字の大きさを変更したいときは表 3.8 の宣言型のコマンドを

{\命令 文字の大きさを変えたい文字列}

のように使用します．

表 3.8　文字の大きさの変更

大きさ	命令	出力例
とても小さい	\tiny	野鳥
かなり小さい	\scriptsize	花鳥
小さい	\footnotesize	雷鳥
やや小さい	\small	白鳥
普通	\normalsize	飛鳥
やや大きい	\large	やちょう
大きい	\Large	かちょう
かなり大きい	\LARGE	らいちょう
とても大きい	\huge	はくちょう
特大	\Huge	ひちょう

```
そういえば，{\scriptsize これ}は小さい文字だけど，{\Large こっち}は大きい文字になってるね．
```

そういえば，これは小さい文字だけど，こっちは大きい文字になってるね．

このような書体の大きさを変更するコマンドを直接使うのは好ましくなく，きちんとマークアップ付けをするべきです．例えば強調のために文字を大きくしたいのであれば新規に\kyocho 命令を作ります．

```
\newcommand{\kyocho}[1]{{\Large#1}}
\newcommand{\Kyocho}[1]{{\LARGE#1}}
ああそういえば\kyocho{ここは大事だからね}．
それに\Kyocho{ここはもっと大事}だよ．
```

ああそういえばここは大事だからね．それにここはもっと大事だよ．

表 3.9　基準の文字の大きさによるコマンドの挙動の違い

コマンド\基準の大きさ	10 pt	11 pt	12 pt	使用すべき要素 *
\tiny	5 pt	6 pt	6 pt	振り仮名
\scriptsize	7 pt	8 pt	8 pt	
\footnotesize	8 pt	9 pt	10 pt	索引・脚注
\small	9 pt	10 pt	11 pt	図表見出し
\normalsize	10 pt	11 pt	12 pt	少少節見出し・本文
\large	12 pt	12 pt	14 pt	小節見出し
\Large	14 pt	14 pt	17 pt	節見出し
\LARGE	17 pt	17 pt	20 pt	
\huge	20 pt	20 pt	25 pt	部・章見出し番号
\Huge	25 pt	25 pt	25 pt	部・章見出し

* 使用すべき要素は 1 段組での場合です.

3.19.2　書体の変更

LaTeX において書体の種類は

ファミリー　デザイン上の系統の種類.
シリーズ　線の太さと文字幅の違いによる種類.
シェイプ　形状の変化の違いによる種類.
サイズ　フォントの大きさ.

の四つに分けられます．サイズに関しては前述の通りです.

文字の大きさや書体の種類を不必要に変更することは，逆に読者を混乱させるだけです．自分はきちんと使い分けることが出来る，という方は表 3.10 の一覧から適切な書体を選んで，美しい文書を作成してください． ファミリーとシリーズとシェイプはそれぞ

表 3.10　書体を変更するコマンド

種類	命令	宣言	出力
ローマンファミリー	\textrm	\rmfamily	ABCabc
サンセリフファミリー	\textsf	\sffamily	ABCabc
タイプライタファミリー	\texttt	\ttfamily	ABCabc
ミディアムシリーズ	\textmd	\mdseries	ABCabc
ボールドシリーズ	\textbf	\bfseries	**ABCabc**
イタリックシェイプ	\textit	\itshape	*ABCabc*
スラントシェイプ	\textsl	\slshape	*ABCabc*
スモールキャピタルシェイプ	\textsc	\scshape	ABCABC

れ組み合わせて使うことが出来ます．例えば「セリフがなくて太くて傾いたフォント」という文字を出力したければ次のようにします．

```
\textsf{\textbf{\textit{I like sushi.}}}
{\sffamily\bfseries\itshape I like sushi.}
```

I like sushi. I like sushi.

使用している基本書体によっては出力できないタイプもあります．

```
\texttt{\textbf{Type writer and bold
 extended?}} I like \textsc{small caps} and
\textit{\textbf{bold italic}}. {\ttfamily
\bfseries type writer and bold extended}
```

Type writer and bold extended? I like SMALL CAPS and ***bold italic***. **type writer and bold extended**

書体のファミリーやシェイプなどを先に指定してから大きさを変更します．

```
{\Large\textbf Large Bold?} 成功．\\
{\textbf\Large Bold Large?} 失敗．
```

Large Bold? 成功.
Bold Large? 失敗.

和文の書体は基本的には明朝体とゴシック体の二つしか用意されていません（表 3.11）．これは従来の和文組版で二つの書体しか使われなかった名残です．現在の pLaTeX で和文の多書体を図ることはそれ程難しくありません．ただ不用意に和文を多書体にしても読者がそれに慣れていないと思われますので，悪戯に行わないほうが良いかもしれません．

表 3.11　和文書体のファミリー

種類	命令	宣言	出力
明朝ファミリー	`\textmc`	`\mcfamily`	永字八法って何ですか？
ゴシックファミリー	`\textgt`	`\gtfamily`	**永字八法って何ですか？**

```
和文組版において明朝体は通常の文章の組版，
ゴシック体は\textgt{文章の強調に}使われます．{\gtfamily 見出しも強調すべき要素なのでゴシック体にするのが普通です．}
```

和文組版において明朝体は通常の文章の組版，ゴシック体は**文章の強調に**使われます．**見出しも強調すべき要素なのでゴシック体にするのが普通です．**

3.19.3　基本書体の変更

フォントの大きさやファミリーなどを指定する命令は分かりました．しかし，文書に使われている書体そのものを変更するにはどうすれば良いのでしょうか．実は普段何気なく LaTeX で文書処理をしているときに使われている欧文のフォントは Donald Knuth 氏がデザインした Computer Modern と呼ばれる**基本書体**が使われています．この基本書体そのものを変更するには基本書体を変更する定義がされたマクロを読み込むか，自分で指定する必要があります．数式に使われる数式書体も基本的に Computer Modern が使われます．フォントついては奥村晴彦氏の文献 [27] などを参照してください．本書では取り扱いません．あえて言うならば Young Ryu 氏が作成した txfonts を使うのが手軽ではないかと思

います．標準配布の Times 系のフォントを使うにする mathptmx や，Palatino 系のフォントを使うようにする mathpazo よりも pxfonts や txfonts の方が良いと思います．

3.20 文章の修正

このようにして基本的な文章構造を組み上げて，結果的に紙の上などに出力するわけですが，一発で完璧な文書になることはほとんどありません．何度も修正と加筆を繰り返し，最終的な論文に仕上がるものと思います．

そのときに必要なのは文章の校正に関わる約束事です．LaTeX ではほとんどの多くの処理を半自動的に行うので，普段は気にならない部分です．例えば半角英数字と全角の日本語とのあいだには 4 分空きといって，全角空白の 4 分の 1 のスペースを挿入したり，行の先頭に句読点があってはいけないという，行頭禁則処理の問題も LaTeX（pTeX において）は半自動で行います．

このような自動的な処理以外にもユーザー側の入力ミスにより修正が必要になる場合があります．その場合は 1 度作成した文章を校正記号 [51] などを使って修正するのが良いでしょう．

現在では文章はコンピュータ上ですべて組むことが出来るので，間違いを見つけたらその場ですぐに修正可能です． 紙に印刷してチェックするという作業は非効率的かもしれません．コンピュータのモニター上と印刷した紙上の両者の特性を活かして文章を修正してください．文章作成上で注意すべき点として

- 1 文を長くしすぎていないか．
- である調で統一されているか．
- 修飾語の関係をはっきりしているか．
- 同音異義語などの間違いはないか．
- 段落の区切り，章の区切りは明確か．

などが挙げられます．

3.20.1 構文チェック

LaTeX の原稿を書いていると括弧が足りないとか，大文字と小文字を書き間違えているなどのコマンドの構文の記述ミスをしてしまいます．自分の記述が間違っていないかを Frank Mittelbach 氏と Rainer Schöpf 氏が作成した syntonly を使うと，実際の出力をせずに構文のチェックだけを行うようにします．`\syntaxonly` 命令をプリアンブルに記述します．syntonly パッケージを読み込んだだけでは何も起こりません．

```
\documentclass{jsarticle}
\usepackage{syntonly}
\syntaxonly
\begin{document}
I like {\LaTex}.% これはエラーになります．
\end{document}
```

構文だけをチェックするので通常の DVI 出力を伴うタイプセットよりも処理が速いと思います.

3.21　クラスとパッケージ

LaTeX はマークアップ言語なので書式と内容は分離されるのが普通です．そこでクラス（class）とパッケージ（package）という二つのファイルを使うようになっています.

さて，クラスやパッケージという用語が出てきましたが，簡単に説明しましょう．昔の話は置いといて（LaTeX2.09），LaTeX では文書の書式を決定するためにクラスというものを宣言します．クラスはドキュメントクラスとか文書クラスなどと呼ばれています．また，便利な機能を集めたものを**パッケージ**と呼びます．パッケージは**マクロパッケージ**とか，ただ単に**マクロ**などと呼ばれます.

そうして，LaTeX の原稿（ソースファイル）では必ず文書の先頭に

```
\documentclass[⟨オプション⟩]{⟨クラス⟩}
\usepackage[⟨オプション⟩]{⟨パッケージ⟩}
```

のような記述をして，文書の書式を大雑把に決定します.

例えば，日本語の小規模の画像を含み，書体の大きさが 11 ポイントで 2 段組の記事を書こうと思えば

```
\documentclass[twocolumn,11pt]{jarticle}
\usepackage[dvips]{color}
\usepackage[dvips]{graphicx}
```

のように原稿中で宣言します．使用するクラスの中にはオプションが存在し，上記のように 2 段組のために *twocolumn* やフォントの大きさを指定するために *11pt* というオプションを指定します．また衝突の起きない限り，複数のパッケージを使うことを同時に宣言することもできます.

```
\documentclass[twocolumn,11pt]{jarticle}
\usepackage[dvips]{graphicx,color}
```

クラスとパッケージを明確に区別するためにクラスの拡張子には.cls を，パッケージの拡張子には.sty を付けるようにします.

3.21.1　標準的なクラス

LaTeX や pLaTeX の範囲内で提供されている標準的なクラスを紹介します．クラスファイルは ⟨*classes*⟩.dtx と ⟨*classes*⟩.ins という二つのファイルで配布されることが多いようです．あるクラスが何からインストールできるのかは，すでにでき上がっているクラスファイル ⟨*classes*⟩.cls が存在すればそのファイルの先頭部分に

```
%% This is file ‘class.cls’,
%% generated with the docstrip utility.
```

```
%% The original source files were:
%% classes.dtx  (with options: 'option')
```

のような記述があります．オリジナルのソースファイルはオプションに *'option'* を指定してファイル名が ⟨*classes*⟩.dtx です，と説明してくれていますので Unix 系 OS ならば

```
▉ find /usr/local/share/texmf/ -name classes.dtx
```

などでそのファイルを検索できるでしょう．Windows や Macintosh はファイルの検索の機能があると思いますので，そちらを使えば良いでしょう．そのようにして検索した場所に ⟨*classes*⟩.dtx や ⟨*classes*⟩.ins が存在します．そのファイルがある場所に移動して，

```
▉ platex classes.dtx
```

とすればそのクラスの仕様書を見ることができます．さらに

```
▉ platex classes.ins
```

とするとそのクラスを導入することができます．このような操作をすると DocStrip というマクロが適切にファイルを処理します．例えばファイル classes.ins をこのように処理すると，article.cls，report.cls，book.cls，bk10.clo，bk11.clo，bk12.clo，size10.clo，size11.clo，size12.clo という九つのファイルが生成されます．

日本語を含まないような文書には欧文専用のクラスが使用できます．それぞれどのような文書を作成したいかによって何を用いるかが分かれます．標準では以下の欧文用のクラスが使えます．

article 小規模の記事を作成するためのクラス．classes.dtx や LaTeX コンパニオンに詳しい仕様が書かれている．

report 報告書を作成するためのクラス．同じく classes.dtx や LaTeX コンパニオンに詳しい仕様が書かれている．

book 書籍を作成するためのクラス．同じく classes.dtx や LaTeX コンパニオンに詳しい仕様が書かれている．

slides スライドを作成するためのクラス．slides.dtx に詳しい仕様が書かれている．

proc article をベースに議事録などを作成するためのクラス．proc.dtx に詳しい仕様が書かれている．

以上の article，report，book の三つをまとめて classes と呼ぶことがあります．

日本語の文書では，標準で以下のクラスが使えます．

jarticle 小規模の日本語の記事を作成するためのクラス．

jreport 日本語の報告書を作成するためのクラス．

jbook 日本語の書籍を作成するためのクラス．

tarticle 縦書きの小規模の日本語の記事を作成するためのクラス．

treport 縦書きの日本語の報告書を作成するためのクラス．

tbook 縦書きの日本語の書籍を作成するためのクラス．

以上の jarticle，jreport，jbook の三つをまとめて jclasses と呼ぶことがあります．

3.21.2　クラスオプション

ドキュメントクラス（文書クラス），単にクラスにはもう少し詳細な設定を行うことができます．\documentclass の任意引数として記述します．多くのクラスファイルでは次のクラスオプションが使えると思います．

文字サイズ〈*10pt*，*11pt*，*12pt*〉　原稿で基本となる文字の大きさを決めます．この文字サイズを基準としてさまざまなパラメータが設定されます．標準は *10pt*．

用紙サイズ〈*a4paper*，*a5paper*，*b5paper*，*letterpaper*〉　原稿の用紙の大きさを指定します．和文の場合はこの他に *b4paper*，*a4j*，*a5j*，*b4j*，*b5j* などです．geometry パッケージや jsclasses を使うと選択の幅が広がります．

用紙方向〈*landscape*〉　用紙を横置きにします．標準は縦置きです．

印刷面〈*oneside*，*twoside*〉　用紙の片面（*oneside*）だけに印刷するかそれとも両面（*twoside*）に印刷するかを指定します．

段組〈*onecolumn*，*twocolumn*〉　1 段組（*onecolumn*）にするか 2 段組（*twocolumn*）にするかを指定します．

表題〈*titlepage*，*notitlepage*〉　表題を独立して出力する（*titlepage*）か，同じページに出力する（*notitlepage*）かという表題のレイアウトを指定します．

数式の位置〈*fleqn*〉　別行数式の位置を左揃えに指定します．標準は中央揃えです．

数式番号の位置〈*leqno*〉　数式番号の位置を左側に指定します．標準は右側です．

ドラフト〈*draft*，*final*〉　文書の領域をはみ出してしまった箇所に印をつけるかどうか．執筆途中で印刷するときにはドラフトモードにする．ドラフトモードの *draft*，原稿が完成したら *final* に変更する．標準は *final*．

左右起し〈*openright*，*openany*〉　(j)report や (j)book において章などの開始ページの指定をする．常に奇数ページで起こす（*openright*）か，どちらからでも起こす（*openany*）かを設定する．(j)report の標準は *openany*．(j)book の標準は *openright*．

最近では，奥村晴彦氏（http://oku.edu.mie-u.ac.jp/~okumura/）が管理している jsclasses というクラスファイル群が定評です．このクラス群を導入すると

jsarticle	小規模の日本語の記事を作成するためのクラス．
jsbook	日本語の書籍や報告書を作成するためのクラス．
jspf	某学会誌用のクラス．

の三つが使用できます．これらのクラスで指定できるクラスオプションが jclasses に追加されています[*2]．以上の jsarticle，jsbook，jspf の三つをまとめて jsclasses と呼びます．

文字サイズ　〈*9pt*，*10pt*，*11pt*，*12pt*，*14pt*，*17pt*，*20pt*，*21pt*，*25pt*，*30pt*，*36pt*，*43pt*，*12Q*，*14Q*〉

[*2] (j)classes で定義されていたいくつかのクラスオプションが実装されていません．

用紙サイズ　〈*a4paper*，*a5paper*，*a6paper*，*b5paper*，*b4paper*，*a4j*，*a5j*，*b4j*，*b5j*，*a4var*，*b5var*〉

言語の指定　〈*english*〉

3.21.3　標準で使用できるパッケージ

LaTeX を導入すると一緒に添付される標準的なパッケージがあります．これらはプリアンブル部分に

```
\usepackage[〈オプション〉]{〈パッケージ〉}
```

として使用可能になります．各パッケージの詳細な説明書が読みたいときは

```
▌platex filename.dtx
```

とすれば〈*filename*〉`.dvi` が作成されます．索引の作成や目次の作成，相互参照の解決などをすれば完全な DVI ファイルが完成します．各ソースファイルへの検索パスがなければ該当する〈*filename*〉`.dtx` を検索することはできません．Windows ならばファイルの検索，Unix 系 OS ならば find コマンドなどで探してください．大抵は LaTeX をインストールしたディレクトリ（フォルダ）の下 '`$texmf/tex/latex/base`' にあります．

LaTeX がコンピュータに導入されているならば以下の応用的なマクロやソフトウェアが同封されていることでしょう．これらのファイルは欧文の文書を作成するうえでは必須のものとされています．日本語の文書のみを作成するならば，いくつかのマクロやソフトウェアは必要ないでしょう．

$\mathcal{AMS}$-LaTeX　米国数学会（American Mathematical Society）が提供しているソフトウェア並びにパッケージ．$\mathcal{AMS}$-TeX という TeX 用を LaTeX でも使えるようにしたもの．マクロ，フォントなどを総称した呼び名が $\mathcal{AMS}$-LaTeX で，パッケージの名前は amsmath と言う．数学系の文書を書くときには必須のマクロ．

babel　多言語を LaTeX で扱うためのマクロ．このマクロを日本語と共存させるためには少々工夫が必要．詳しい情報は付録の A.4 節を参照してください．

graphicx　画像の挿入や加工などを担うマクロ．同時に color というマクロも含まれる．これはデバイス依存の機能で環境により出力が異なることがある．

tools　LaTeX3 プロジェクトチームによって提供される標準からはこぼれたマクロ．

これらのマクロについては少なくとも LaTeX コンパニオンか文書処理システム LaTeX 2ε に記述されていることが保証されています．

LaTeX3 プロジェクトチームによって提供される tools は '`$texmf/tex/latex/tools`' に置かれており，その内訳は以下のとおりです．

array	`array` や `tabular`，`tabular*` のような表や行列を拡張した環境を使うことができるマクロ．
calc	LaTeX での計算を楽にするマクロ．

dcolumn	表や行列の環境で小数点などを揃えるためのマクロ.
delarray	行列で括弧付けを容易にするためのマクロ.
hhline	表や行列で複雑な罫線を簡単に引くことができるマクロ.
longtable	ページをまたぐような長いなが〜い表を作るときに使うマクロ.
tabularx	通常の tabular 環境よりも幅に関して柔軟な表を作るためのマクロ.
afterpage	\clearpage の拡張版のような\afterpage が使えるマクロ.
bm	数式中で太字を簡単に使うようにするためのマクロ.
enumerate	enumerate 環境を拡張するためのマクロ.
fontsmpl	任意のフォントの一覧を表示するためのマクロ.
ftnright	2 段組で全ての脚注を右側に表示するマクロ.
indentfirst	jarticle や jreport などの標準的なクラスで, 章（\chapter）や節（\section）の直後の段落でも字下げを行うようにするマクロ．通常は字下げしないように設定されているので，和文文書を作成しているときはいつでも読み込むようにすれば良い.
layout	現在の文書のページレイアウトを表示するマクロ.
multicol	他段組を実現するためのマクロ.
showkeys	\label，\ref，\cite などの相互参照のラベル名（keys）を表示するためのマクロ.
theorem	定義型環境を簡単に宣言するためのマクロ.
varioref	相互参照をしやすくするためのマクロ.
verbatim	verbatim 環境を拡張するためのマクロ.
xr	別の文書とでも相互参照できるようにするためのマクロ.
xspace	文中で使われるようなマクロに適切な空白の挿入などを行うマクロ.

第 4 章

参考文献の出力

論文などの文書で重要なのが参考文献です．参考文献の扱いがきちんとできればより良い論文になります．参考文献を明記することはその文献の著者に対する礼儀です．さらに読者がその論文に興味を持ったとき，その事項を深く知るための道しるべにもなります．そもそも他人の著作物を引用したり転載するには著作権法という法律の範囲内で行う必要があります．この章では LaTeX での参考文献の取り扱い方を紹介します．

4.1 参考文献の慣わし

参考文献を明記することはその文献の著者に対する礼儀です．さらに読者がその論文に興味を持ったとき，その事項を深く知るための道しるべにもなります．そもそも他人の著作物を引用したり転載するには著作権法という法律の範囲内で行う必要があるのは前述の通りです．参考文献は文書の終わりにまとめて記載するのが慣わしです．本文中では括弧書きで「著者名と年号」だけの表示にしたり，参考文献の通し番号だけの表示にする場合もあります．

参考文献は文書の終わりにまとめて表示するのは分かりましたが，さらにそれらの文献はあるスタイルに合わせて**並べ替える**ことになります．例えば参考文献を引用した順番で並べ替えるスタイルや，文献の著者名順に並べ替えるスタイルもあります．いずれにしても読者に対しての明確な道しるべとして存在する必要がありますので，その点を考慮した並べ替え方を行います．例えば参考文献が非常に多い場合，これらを手動で並べ替える作業だけで一晩かかりそうです．これを自動化するために Oren Patashnik 氏が作成した BibTeX というプログラムを使うと便利です．通常は日本語化された jBibTeX を使うことになるでしょう．

4.2 参考文献を手動で並べる場合

まずは文献を手動で並べ替えそれを出力する方法を先に紹介します．参考文献がそれほど多くない場合は文献を手動で並べ替えることが考えられます．そのときは thebibliography 環境を使います．文献を

```
\bibitem[〈表示形式〉]{〈ラベル〉}〈項目〉
```

のように文書の末尾にまとめます．これらの文献を thebibliography 環境を使って囲みます．

```
\begin{thebibliography}{⟨ラベルの幅⟩}
\bibitem[⟨表示形式⟩]{⟨ラベル⟩} ⟨項目⟩
\end{thebibliography}
```

参照するときは

```
\cite{⟨ラベル⟩}
```

とします．例を示すと以下のようになります．

```
Donald Knutu 氏による\emph{\TeX ブッ
ク}~\cite{TeX book}は手元に置きたい
名著である．
\begin{thebibliography}{9}
\bibitem{TeX book} Donald~E. Knuth.
   改訂新版{\TeX}ブック．
   アスキー，1989.
\end{thebibliography}
```

ここでの thebibliography 環境の引数は ‘9’ となっていますがこれは参考文献の表示形式に割り当てるラベルの最大の幅を指定します．参照している文献が一桁のときは

```
\begin{thebibliography}{9}
```

のようにしますが，2 桁に突入したときは

```
\begin{thebibliography}{99}
```

と書きます．

4.2.1　文献の並べ方

thebibliography 環境では文献は自動的に並べ替えられません．そのときは手動で文献を並び替えます．文献の並べ替えの仕方は様々あるのですが書籍の場合

```
\bibitem[⟨表示形式⟩]{⟨ラベル⟩} ⟨著者⟩. ⟨書名⟩. ⟨発行年⟩, ⟨出版社⟩.
```

とするのが一般的です．読者には**誰のなんという文献**ということが伝わりやすいスタイルです．このように文献を追加し，複数の文献を並べるときは

- 最初の著者の姓をアルファベット順で並べる．
- 同じ著者から複数の文献を参考にしているときは発表年が早い方を先に並べる．

という規則に従います．何か複雑な表記に直面したときは項目の最後に「補足」を付け足すようにすれば簡単に処理できます．

日本人の名前，「未来太郎」の場合は ‘Taro Mirai’ という読みになるので

```
\bibitem[Hokkai 1997]{HM1997a} Michiko Hokkai.\
   \emph{Going My Way}.\ 1997,\ Future.\
\bibitem[Hokkai 1999]{HM1999a} 北海道子.\
   それが私の生きる道.\ 1999,\ 未来出版.\
\bibitem[Watanabe 2000]{NN2000a} 渡辺徹.\
   ほげほげはほげ.\ 2000,\ NNN 出版.\
```


のような文献リストがあった場合は，‘[Hokkai 1999]’ と ‘[Watanabe 2000]’ のあいだに入ることになります．

例では北海道子さんの場合は「北海道子」と ‘Michiko Hokkai’ の 2 通りあります．これは正しくなく同じ著者名の表示は統一します．さらに日本人の名前は姓名のあいだに半角の空白は挿入しません．表示形式は特に指定しなかった場合は昇順に番号付けされます．この表示形式の規則としては「[番号]」とか「[名前 年号]」など作成者と読者に分かりやすいような表示方法にすれば良いでしょう．

しかし，これは自分で文献を並べ替えなどする必要がありますので文献を沢山参照している論文などを作成するときには実用的とは言えません．

4.3　参考文献をプログラムで並べ替えるとき

参考文献が非常に多い場合は手動で並べ替えるのが困難です．参考文献の番号付け，並び替えを行うときに著者順とか発表年順などの規則が存在します．LaTeX にはこのような手間を省いてくれるプログラムがきちんとあります．日本語化された jBibTeX [79, 78] というのがこれにあたります．原理は簡単で決められたスタイルに合わせて複数の文献を並び替えるだけです．

4.3.1　jBibTeX の使い方

参考（引用）文献は LaTeX のソースとは別のファイルに保存します．これを**文献データベース**と呼びます．ファイル名は任意でよいのですが拡張子は `.bib` となるようにしてください．

4.3.2　文献データベースの作成

プログラムによって半自動的に文献を並べ替える方法を紹介します．まずは**文献データベース**と呼ばれるファイルを作ります．名前は `hoge.bib` ということにしておきます．使い方は一つの文献に対して

```
@⟨文献の形式⟩{⟨ラベル⟩,
⟨属性 1⟩={⟨値 1⟩},
⟨属性 2⟩={⟨値 2⟩}, }
```

という記述をします．このような記述を文献の数だけ作成します．参考文献といっても色々ありますので，まずは具体例を見てください．

```
@book{TW2004a,
author    = {渡辺 徹},
yomi      = {Toru Watanabe},
title     = {ほげを深く考える},
publisher = {ほげほげ出版},
year      = {2004},
note      = {実在しません}}
```

これを見て，勘の良い人ならば様々なルールがあることに気づくでしょう．

- 一つの文献はアットマーク ‘@’ からはじめます．
- ‘@’ の後に ‘book’ とありますがこれは「文献の形式」を表します．この場合は一般に本屋さんで売っている ‘book’ であることが分かります．
- 次にその文献の情報を波括弧で括ります．括るときはまずその文献に ⟨ラベル⟩ をつけます．要は目印です．これがないと参照できません．ここでは覚えやすいように ‘TW2004a’ と著者名の頭文字と発行年にしています．
- ‘author’，‘yomi’，‘title’，‘publisher’，‘year’，‘note’ などは想像がつくでしょう．
- 注意しなければならないのは**行末にコンマを入れる**ということです．これがないと処理の段階でエラーになります．
- **値は波括弧で囲む**ことも忘れてはいけません．
- 日本人の著者名は姓名のあいだに半角の空白を入れます．出力されるときは自動的に除かれます．
- 著者名の ‘yomi’ には「名」の次に「姓」を書きます．

このような文献データベース hoge.bib を作成したならば，今度は論文の本体で，この文献を参照します．参照のコマンドは\cite です．方法は 4.2 節の場合と同様です．

4.3.3　参考文献の出力

一通り参照したら今度は LaTeX 文書の一番最後に参考文献を出力する記述を追加します．プリアンブルですることはありません．文書の最後のほうで\bibliography 命令を使って次のようにします．

```
\bibliographystyle{⟨スタイル⟩}
\bibliography{⟨ファイル名⟩}
```

⟨スタイル⟩ には文献を並べかえるスタイルを指定し，⟨ファイル名⟩ には文献データベースの ⟨ファイル⟩.bib から拡張子.bib を除いた名前を書きます．

これでソースファイルの編集は終わりました．しかしこのままでは参考文献の一覧は出力されません．ここで jBibTeX というプログラムを使用します．端末などからファイルのある場所に移動して次のコマンドを実行します．

```
▉ platex file .tex
▉ jbibtex file
```

```
platex file.tex
platex file.tex
```

とすると参考文献が出力されます．ここで参考文献データベースに文献を追加していても本文中で参考していない（\cite 命令で参照していない）場合はその文献は一覧には出力されませんので注意してください．本文中で明示的に参考しなくても文献リストには出力したいときには

```
\nocite{⟨ラベル⟩}
```

とすることで参考文献リストにも出力できます．

さて，どうしてこのようになっているのかを少し考えてみましょう．まずは以下のようなファイル hoge.tex を作成してください．

```
\documentclass{jsarticle}
\begin{document}
この冊子~\cite{TW2004a}は稚拙だ．
\bibliographystyle{jplain}
\bibliography{ref}
\end{document}
```

次に以下のような参考文献データベース ref.bib を作成してください．

```
@book{TW2004a,
author    = {渡辺 徹},
yomi      = {Toru Watanabe},
title     = {ほげを深く考える},
publisher = {ほげほげ出版},
year      = {2004},
note      = {実在しません}}
```

このようなファイルが出来上がったら端末から hoge.tex を 1 回だけタイプセットします．すると端末には

```
LaTeX Warning: Citation 'TW2004a' on page 1 undefined on
input line 3.
No file hoge.bbl.
LaTeX Warning: There were undefined references.
```

のような警告が表示されます．一つ目の警告では参照する対象が見つからないと言われています．次に `No file hoge.bbl` と言われて hoge.bbl というファイルが足りないことになっています．最後に正しく相互参照の処理を出来なかったと報告されます．

ここで hoge.aux の中身をのぞいてみましょう．ファイルの中には

```
\relax
\citation{TW2004a}
\bibstyle{jplain}
\bibdata{ref}
```

の 4 行が書き出されているでしょう．実は JBibTeX はこの情報を使って参考文献の並び替えをします．例えば文書で引用された順に文献を並び替えるときにはその引用された順番が分からなければなりませんから，このように ⟨*file*⟩.aux から何らかの情報を得ることになるのです．

次に JBibTeX を使って文献の一覧を作成します．端末などから

```
▉ jbibtex hoge
```

とすると

```
This is JBibTeX, Version 0.99c-j0.33 (Web2C 7.5.2)
The top-level auxiliary file: hoge.aux
The style file: jplain.bst
Database file #1: ref.bib
```

のようなメッセージが表示されます．1 行目には JBibTeX のバージョン情報，2 行目には使用した中途ファイル（hoge.aux），3 行目には文献を出力するスタイル（jplain.bst），最後に文献データベース（ref.bib）には何を使ったのかが出力されています．

このようにして並べ替えなどを終えた文献一覧はこの場合 hoge.bbl に書き出されています．実際 hoge.bbl の中身を見てみると

```
\begin{thebibliography}{1}
\bibitem{TW2004a} 渡辺徹.
\newblock ほげを深く考える.
\newblock ほげほげ出版, 2004.
\newblock 実在しません.
\end{thebibliography}
```

という出力となっています．これは 4.2 節で手動で記述した場合と類似しています．この段階で hoge.bbl が作成されていない場合は何らかの記述ミスが考えられますので JBibTeX の処理結果を hoge.blg から読み取ってください．

うまく hoge.bbl が作成されているならば，その文献一覧を hoge.tex に取り込むために

```
▉ platex hoge.tex
```

として再度タイプセットします．すると端末には

```
LaTeX Warning: Citation `TW2004a' on page 1 undefined on input line 3.
(./hoge.bbl) [1] (./hoge.aux)
LaTeX Warning: There were undefined references.
LaTeX Warning: Label(s) may have changed.
Rerun to get cross-references right.)
```

という警告などが表示されます．ラベルに関して何か変更があったから再度タイプセットしなさいと言われていますので，

```
▉ platex hoge.tex
```

として 3 度目のタイプセットをすることになります．3 度もタイプセットしなければならないのは面倒かもしれませんが，1 度 jBibTeX によって文献一覧 ⟨*hoge*⟩.bbl を作成しておけば再度文献一覧を作成するのは新しく文献を参照したときだけだと思います．原稿執筆中は特に正式な文献一覧が必要なわけではありませんので，最終的な原稿のタイプセットのときだけ

```
platex hoge.tex
jbibtex hoge
platex hoge.tex
platex hoge.tex
```

とすれば良いことになります．もう少し手間のかからないものとして Make や latexmk を使う方法もあります．

4.3.4　文献の種類及び項目

```
\bibliographystyle{⟨スタイル⟩}
\bibliography{⟨ファイル名⟩}
```

\bibliographystyle 命令は参考文献の出力形式を指定します．‘jplain’ というのは「日本で標準」の形式という意味です．\bibliography 命令で文献データベースを読み込んでいます．これは複数ファイルをカンマで区切って読み込んでもできます．

参考文献としてその文献がどのような形式なのかを指定する必要があります．雑誌の 1 部なのか，論文の 1 部なのかを明示します．

```
@book{label,
```

となっている一行で ‘book’ となっている部分に対応する形式を表 4.1 から選んでください．

表 4.1　文献の形式

文献の形式	説　明
article	論文誌など発表された論文
book	出版社の明示された本
booklet	印刷，製本されているが出版主体が不明なもの
inbook	書物の一部（章，節，文など何でも）
incollection	それ自身の表題を持つ，本の一部分
inproceedings	会議録中の論文
manual	マニュアル
masterthesis	修士論文
phdthesis	博士論文
misc	他のどれにも当てはまらないときに使う

‘author’，‘title’，‘publisher’，‘year’ 以外にも指定することの出来る項目があります．文献リストの各文献に表 4.2 の項目（フィールド）を追加します．文献の ⟨形式⟩ に

表 4.2　フィールド名

項目	内容
address	出版社の住所
annote	注釈付きのスタイルで使われる
author	著者名
booktitle	本の名前
chapter	章，節などの番号
crossref	相互参照する文献のデータベースキー
edition	本の版
editor	編集者
howpublished	どのようにして発行されたか
journal	論文誌名
key	著者名がないときに相互引用，ラベル作成などに使われる
month	発行月か書かれた月
note	読者に役立つ付加情報
number	論文誌などの番号
organization	会議を主催した機関名あるいはマニュアルの出版主体
pages	ページ（範囲）
publisher	出版社（者）名
school	論文が書かれた大学
series	シリーズの本の名前
title	表題
volume	論文誌などの巻
year	発行年か書かれた年

より必須となる項目が違います．各文献における必須項目と任意項目は表 4.3 の通りです．必須項目は必ず記述しなければならない項目で任意項目は必要に応じて書き足せば良いでしょう．項目のあるなしで文献の並べ替えに若干の影響が出ますが，それ程神経質になる必要はありません．どこかの学会などに提出時などは jBibTeX によって並び替えられた ⟨*file*⟩.bbl を手動で修正・置換したほうが早いかもしれません．

著者 ‘author’ が複数人数のときはカンマで区切るのではなく

```
author={夏目 漱石 and 福沢 諭吉 and 芥川 龍之介}
```

のように ‘and’ を使用します．また著者の苗字と名前のあいだには半角の空白を挿入するようにしてください．‘author’ や ‘editor’ の名前が非常に多いときには名前を

```
author={代表著者 and others}
```

表 4.3　文献の種類における必須・任意項目

文献の種類	項目
article	author, title, journal, year
任意	volume, number, pages, month, note
book	author, title, publisher, year
任意	volume, series, address, edition, month, note
booklet	title
任意	author, howpublished, adddress, month, year, note
inbook	author, title, chapter, pages, publisher, year
任意	volume, series, type, note , address, edition, month
incollection	booktitle, author, title, year publisher,
任意	editor, volume, series, type, month, note, address, edition
inproceedings	author, title, booktitle, year
任意	editor, volume, series, pages, address, month, organization, publisher, note
manual	title
任意	author, address, edition, month, year, note, organization
masterthesis	author, title, school, year
任意	type, address, month, note
misc	
任意	author, title, howpublished, month, year, note
phdthesis	author, title, school, year
任意	type, address, month, note

とします．こうすると標準スタイルの jplain では自動的に適切な名前に変換されます．

4.3.5　各文献スタイルの出力例

BibTeX にはどのような文献スタイルが用意されているのかをここで出力例による紹介をします．通常は jplain で問題ないのですが学会によっては参考文献の出力形式を指定される場合があります．使用できるものは欧文の場合，plain，alpha，abbrv，unsrt の 4 つほどで和文の場合は，jplain，jalpha，jabbrv，junsrt となります．他にも WWW 上には個人や学会で文献スタイルを公開していることがありますので，それらを使用することも可能です．

jplain　昇順に通し番号つけるだけの単純なもの.

[1] 野比太郎, 剛太タケル. 2000. 四次元ポケットの考察. NNN 出版.

jalpha　著者が一人の場合は著者は「頭文字 3 文字 年号」で表示し，共著のときは「各著者のイニシャル 年号」で表示する．‘key’ 項目を追加することにより表示するイニシャルなどを変更できる.

[NG 2000] 野比太郎, 剛太タケル. 2000. 四次元ポケットの考察. NNN 出版.

jabbrv　著者名，月，誌名を簡略表記にする.

[1] 野比, 剛太. 2000. 四次元ポケットの考察. NNN 出版.

junsrt　文献を本文中で参照している順番で並べ替える.

[1] 野比太郎, 剛太タケル, 2000. 四次元ポケットの考察. NNN 出版.

4.3.6　文献を同時に複数参照しているとき

複数の文献を同時に参照しているときは ‘[3,2,5,1]’ となってしまい文献リストの表示が並べ替えられず，‘[1-3,5]’ となりません．その場合 Donald Arseneau 氏による cite パッケージを使います．ただし hyperref との併用はできません．このパッケージを利用すれば参考文献が複数ある場合 ‘[1-3,5]’ のように連番をハイフンでつなげ昇順に並べ替えます．プリアンブルで読み込むだけで使用可能です.

4.3.7　参照の形式を変更する

ある分野では丸括弧を使い著者名と年号を使うスタイル「(渡辺 1999)」が推奨される（丸括弧は全角で）場合は

```
\makeatletter
\renewcommand{\@cite}[2]{（{#1\if@tempswa , #2\fi}）}
\renewcommand{\@biblabel}[1]{（#1）}
\makeatother
```

という記述をプリアンブルに入れると良いでしょう．JBibTeX を使っている場合は文献スタイルに依存します．親切な方がウェブで提供していると思いますので文献スタイルファイル探してみてください.

もう少し細かい設定をしたい場合は cite パッケージを使います．このパッケージのオプションとして

nospace　項目のあいだの区切りで単語間空白を挿入しません.

space　項目のあいだの区切りで単語間空白を挿入します.

nosort　並び替えを行いません.

などが用意されています.

```
\usepackage[space]{cite}
```

表 4.4　cite パッケージで変更できる命令

命令	意味	標準のスタイル
\citeform	個々の項目の修飾	なし
\citepunct	項目の区切り	コンマと小さい空白
\citeleft	リストの左括弧	[
\citeright	リストの右括弧	]
\citemid	\cite の任意引数の前に付ける記号	コンマと文字間空白

のように使用してください．設定できるコマンドとして表 4.4 の五つがあります．まずは使用例を見てください．例えば以下のようなファイル mycite.tex を作成します．

```
\documentclass[12pt]{jsarticle}
\usepackage{cite}
\begin{document}
そうです~\cite[どれよ]{First,Second,Third,Sixth,Fifth}.
\begin{thebibliography}{9}
 \bibitem{First}  First Name. \emph{はじめ}. 1991, ほげ出版.
 \bibitem{Second} Second Name. \emph{つぎ}. 1992, ある出版.
 \bibitem{Third}  Third Name. \emph{つぎのつぎ}. 1993, ある社.
 \bibitem{Forth}  Forth Name. \emph{そのつぎ}. 1995, ほげ社.
 \bibitem{Fifth}  Fifth Name. \emph{さらにつぎ}. 1994, ほげ出版.
 \bibitem{Sixth}  Sixth Name. \emph{さいご}. 1990, ほげ堂.
\end{thebibliography}
\end{document}
```

このまま何も設定しなければ，

> そうです [1–3,5,6, どれよ].

のように並べ替えられ，\cite の任意引数の「**どれよ**」の前にコンマと小さい空白が挿入されております．さらに項目はコンマで区切られています．次にこのファイルのプリアンブルに（‘\usepackage’ の後に）

```
\renewcommand\citemid{; }
\renewcommand\citeleft{ (}
\renewcommand\citeright{) }
\renewcommand\citepunct{,}
```

という記述をしておけば

> そうです（1–3,5,6; どれよ）.

という出力になります．個々の項目を修飾するためには\citeform 命令の再定義をします．ローマ数字で番号を表示するときは

```
\renewcommand\citeform[1]{\romannumeral 0#1}
```

とすると

> そうです（i–iii,v,vi; どれよ）.

のようになります．個々の項目を丸括弧で囲むときは

```
\renewcommand\citeform[1]{（#1）}
```

としたり，章番号を前に付けたいときは

```
\renewcommand\citeform{\thechapter.}  % [1.3-1.4]
```

とします.

4.4　文献の管理

文献の数があまりに多くなると管理するのが大変になります.

インターフェースは英語ですが Nizar Batada 氏が作成した JBibTeX Maneger というプログラムが

http://csb.stanford.edu/nbatada/JBibtexManager/

にあります．Java で動くので OS に依存しないと思います．これを使うと GUI で文献を管理できます．項目による文献の並び替え表示などもあります．日本語がうまく通るかどうか検証していません.

Morten Alver 氏と Nizar Batada 氏が作成した JabRef が

http://jabref.sourceforge.net/

にあります．こちらも Java で動作するプログラムです．日本語がうまく通るか分かりませんが，ソースが公開されているのでどうにかなるでしょう.

シェアウェアで Ref for Windows が

http://homepage3.nifty.com/refwin/

にあります．こちらは Windows 上で動作する文献管理プログラムです．Ref for Windows で文献のリストを作成し，それを ⟨*file*⟩.bib に出力するという手法のプログラムです.

第 5 章

原稿の出力形式

LaTeX の原稿の執筆が終わったらそれを組版（タイプセット）しなければならないのは自明のことですが，どのようなファイル形式にするかは用途により分かれるところです．この章ではどのようなファイル形式があるのか，どうやって変換するのかを説明します．

5.1 出力形式の種類

LaTeX の原稿の執筆が終わったらそれを組版（タイプセット）しなければならないのは自明のことですが，どのようなファイル形式にするかは用途により分かれるところです．目的と気分によってその形式を変えます．それぞれの形式がどのような特徴を持っているのかを知っておかなければ，どの形式に変換すれば良いのかが分かりません．ですからまずはどのような形式が存在し，どのような特徴があるのかを紹介します．

DVI　DVI は *Device Independent* の略で装置に依存しない汎用のページ記述言語です．画像を含んだり特殊な描画を行っていない原稿の場合はこの DVI ファイルから印刷を行うことができます．装置に依存する命令もこの DVI ファイルの中に記述されており，それを適切に解釈してくれるデバイスドライバがあります．通常はプレビュー作業用に使われています．DVI ファイルは ⟨*file*⟩`.dvi` のように拡張子が `.dvi` となります．

PostScript　Adobe 社が昔に開発したページ記述言語です．現在のバージョンは 1.3 で Unix 系 OS ではこの PostScript 形式のファイルがプレビュー及び印刷に広く使われています．良く PostScript を省略して PS と書くことがありますし，拡張子は `.ps` になっています．標準では ファイルが圧縮されないので ⟨*file*⟩`.ps.gz` の形で配布されているかもしれません．印刷業界でもこの PostScript 形式が良く使われています．PostScript の仲間に EPS（Encapsulated PostScript） というファイル形式もあります．こちらはベクトル画像などに良く使われています．

PDF　PDF は Portable Document Format の略で Adobe 社の開発している PostScript の後継のページ記述言語です．現在のバージョンは 1.5 でプレビューと印刷結果が同程度の品質を得ることができる形式です．世界中で広く使われています．日本語は通りませんが LaTeX 形式の原稿を直接 PDF に変換する pdfLaTeX というプログラムも存在します．

HTML　HTML HyperText Markup Language の略でウェブ上で情報を公開するためのハイパーリンク（Hyper Link）という機能を備えたページ記述言語です．普段ウェブブラウザから見ているページも HTML で記述されています．現在は HTML の後継の XHTML が主流になろうとしています．LaTeX と同じようにマークアップ言語です．

以上の形式のほかにもあるのですが，有名な形式はこの四つです．この章ではどのように LaTeX の原稿を各形式に変換するかを解説します．

5.1.1　DVI

DVI とは *DeVice Independent* の略でデバイスに依存しないファイル形式です．通常 LaTeX が成形後の結果をまとめるのもこの DVI 形式です．platex などのプログラムで LaTeX の原稿を端末などから

```
platex filename.tex
```

とすると LaTeX ファイル ⟨*filename*⟩`.tex` から運が良ければ DVI ファイル ⟨*filename*⟩`.dvi` が生成されます．DVI ファイルにはグラフや画像などの図は挿入されていませんが，それらの情報は DVI ファイルに記載されています．図などの特別な情報を解釈できるかはその**プレビューアやデバイスドライバに依存しています**．DVI ファイルはプレビューなどで一時的に組版後の結果を確認するに便利です．Windows では大島利雄氏らが開発している dviout，Unix 系 OS ならば xdvi や xdvik などが使えます．

5.1.2　PostScript

机上出版（DTP）が始まった頃から Adobe 社の PostScript というのが出版業界におけるページ記述言語の標準です．プログラミング言語としての完成度も高く非常に洗練されたページ記述言語です．今でも多くの出版社，印刷所がこの PostScript を採用しています．PostScript は印刷を目的としたファイル形式なのできちんと手順を踏めば高品質な印刷結果を得ることができます．LaTeX もこの PostScript 形式への出力が可能となっています．この PostScript 形式のファイルは Ghostscript と呼ばれるプログラムを使うことにより，コンピュータ上で閲覧したり，プリンターで印刷することができます．

5.1.3　PDF

Adobe 社が開発した電子文書形式で PDF という形式があります．PDF は *Portable Document Format* の略で，パソコンの画面からでも印刷したのと寸分違わぬ表示を得ることができます．マニュアルの配布や資料の配布ではこの PDF 形式が広く用いられています．

5.2　DVI を PS に——dvips(k)

Tomas Rokicki 氏が開発し，Karl Berry 氏が Kpathsearch に対応させた dvips を使うと DVI ファイルを PS ファイルに変換できます．dvips というプログラムは Windows の方は dvipsk，Unix 系 OS の方は dvips という名前が付いているかも知れません．Red Hat の場合は pdvips という名前になっています．使い方は端末などから

```
▌dvips filename.dvi
```

とするだけです．拡張子.dvi は省略しても構いません．この dvips を実行するときのコマンドラインオプションが多数あります．主なオプションを載せておきます．

-D 〈解像度〉　出力する解像度を dpi 単位で指定します．
-o 〈ファイル名〉　出力するファイル名を指定します．
-t 〈サイズ〉　*a0* から *a8*，*b0* から *b8* の範囲で用紙の大きさを指定します．
-T 〈横幅〉,〈高さ〉　用紙の大きさを単位付き直接指定します．‘21cm,27cm’ のように使います．このようにしなくとも原稿のプリアンブルで

```
\AtBeginDvi{\special{papersize=210mm,270mm}}
```

としても同じことになります．
-A　奇数ページだけ出力します．
-B　偶数ページだけ出力します．
-p 〈ページ番号〉　出力する最初のページを指定します．ただし LaTeX の原稿中のページ番号を参照します．
-l 〈ページ番号〉　出力する最終のページを指定します．ただし LaTeX の原稿中のページ番号を参照します．
-pp 〈ページリスト〉　出力するページ範囲を指定します．これも LaTeX のページ番号に依存します．11,21-35 のようにコンマで複数ページ指定することもできます．
-r　印刷するページの順序を逆順にします．
-P 〈設定〉　設定ファイルを読み込みます．標準では config.ps というファイルを読み込みます．Windows の方は常に config.dl を読み込むために

```
▌dvips -P dl filename.dvi
```

などとするのが良いでしょう．

使い方は

```
▌dvips 〈オプション〉〈引数〉 filename.dvi
```

とするだけです．Windows の方で角藤版の pTeX を使っているならば

```
▌dvipsk -D 2400 -Pdl -o output.ps input
```

とすると良いでしょう．さらに複数ページからなる DVI ファイルから特定のページだけを EPS 形式にしたいというならば

```
▉ dvipsk -E -Pdl -pp14 -o outp14.eps input
```

とします．このようにして抽出した EPS 形式のファイル `outp14.eps` は EPS 画像として再利用できます．

5.2.1　PostScript ファイルの加工

LaTeX ファイルをタイプセットした ⟨*file*⟩`.dvi` を dvips で ⟨*file*⟩`.ps` まで変換することはできますが，この PostScript ファイルを編集することができれば便利です．これには Angus Duggan 氏の psutils というツール群が役立ちます．ページの再配置や面付け作業などもこの psutils で行うと良いでしょう．

5.3　DVI を PDF にその１

Mark Wicks 氏が作成した Dvipdfm [76] を使うと DVI ファイルから PDF を作成できます．平田俊作氏の日本語化パッチを当てたバージョンがそれぞれの環境で入手できます．それから現在 dvipdfm は平田俊作氏と Cho Jin Hwan 氏が中心となって活動している dvipdfmx project team によってさらに改良が加えられ Dvipdfm*x* へと進化しています．dvipdfm は少々古くなっていますので後継の Dvipdfm*x* を使うことをお勧めします．Dvipdfm*x* は dvipdfm の上位互換のようなものですので，まずは dvipdfm の基本的な使い方を知っておきましょう．dvipdfm で可能な操作がそのまま Dvipdfm*x* に当てはまります．

dvipdfm の主な機能は PDF ブックマーク，HyperTeX，Tpic スペシャルなどをサポートしています．画像ファイルは JPEG，PNG，EPS，EPDF ファイルの**バウンディングボックス**という情報されあれば，そのまま PDF に取り込むことができるようになります．

dvipdfm にはコマンドラインオプションによってある程度の出力結果の設定を行います．主要なオプションは以下の通りです．

`-c`　カラースペシャルを全て無効にします．白黒印刷のときなどに使います．

`-e`　サブセットフォント化をやめます．最近の Dvipdfm*x* ではこの`-e` オプションが削除されています．

`-f` ⟨ファイル名⟩　フォントマップファイルを指定します．

`-m` ⟨数字⟩　ページの拡大率を指定します．`-p` オプションとと併用すると良いでしょう．

`-o` ⟨ファイル⟩　出力するファイル名を指定します．標準では ⟨*file*⟩`.dvi` を指定すれば ⟨*file*⟩`.pdf` が作成されます．

`-p` ⟨**サイズ**⟩．　出力する用紙のサイズを指定します．標準では *a4*．指定できるサイズは *letter*，*a6*，*a5*，*a4*，*a3*，*b5*，*b5*，*b4*，*b3*，*b5var* などです．このようにしなくとも原稿のプリアンブルで

```
\AtBeginDvi{\special{pdf:papersize width 210mm height 270mm}}
```

としても同じことになります．

`-l`　用紙を横置きにします．ソースファイル中でドキュメントクラスオプションの

landscape が有効でなければ意味がないでしょう．

`-s` ⟨範囲⟩　出力するページの範囲を指定します．ハイフンを使うと範囲を指定，コンマを使うと複数の範囲を指定できます．例えば ‘`-s 3-5,10-20`’ とすると 3–5 ページと 10–20 が一つの PDF に出力されます．ハイフンの片方に何もないとそれ以前か，それ以降のページを全て含みます．‘`-s 15-`’ とすると 15 ページ以降全てを出力します．他にもページを逆順にすることもできます．また悪ふざけで ‘`-s -,-`’ とするとどのような出力なるか試してみると良いでしょう．

`-r` ⟨解像度⟩　PDF ファイルの解像度を指定します．標準は 600dpi になっています．

`-V` ⟨バージョン⟩　PDF のバージョンを指定できます．2 から 5 までのバージョンを指定できますが，古いバージョンを指定すると意図しない結果になることがあります．互換性を優先しなければならないときなどに使います．

`-x` ⟨長さ⟩　水平方向のオフセットを指定します．標準は `1.0in` です．単位には mm，cm，in，pt が使えます．

`-y` ⟨長さ⟩　垂直方向のオフセットを指定します．標準は `1.0in` です．単位については `-x` と同様です．

`-z` ⟨数字⟩　圧縮率を指定します．圧縮率は 0–9 まで指定でき 9 が最高です．標準は *9* ですのでビットマップ画像などの画質を落としたくない場合は 0 などにすると良いでしょう．

`-v`　処理内容を標準出力に詳しく表示します．

`-vv`　さらに処理内容を詳しく表示します．

例えば白黒印刷用の DVI ファイルの 15 ページから 20 ページを PDF に変換したいときは

```
▌dvipdfm -c -s 15-20 -o output.pdf input.dvi
```

のようにします．入力ファイルの拡張子 `.dvi` は省略しても構いません．

PDF ファイルを Adobe Reader や Acrobat Reader などで閲覧しているときに dvipdfm による DVI ファイルの変換を行うと `** ERROR ** Unable to open output.pdf` というメッセージを表示してエラーになります．1 度開いている PDF ファイルを閉じてから再度変換すると良いでしょう．

5.3.1　画像ファイルの扱い

dvipdfm において JPEG，PNG，PDF，EPS などの画像ファイルは**バウンディングボックス**という情報があれば張り込むことが可能です．dvipdfm 付属の ebb というプログラムで画像のバウンディングボックスの作成をすれば，JPEG，PNG，PDF，EPS を直接 PDF に張り込めます．具体的な手順としてはファイルのあるディレクトリで

```
▌ebb filename.jpg
```

とすれば拡張子が `.bb` の ⟨*filename*⟩`.bb` というファイルが作成されます．作成された ⟨*filename*⟩`.bb` を見てみると

```
%%Title: ./filename.jpg
%%Creator: ebb Version 0.5.2
%%BoundingBox: 0 0 595 842
%%CreationDate: Tue Dec 30 13:04:10 2003
```

のように〈ファイル名〉,〈作成プログラム〉,〈バウンディングボックス〉,〈作成日時〉の情報が出力されます．沢山〈*filename*〉.bb のファイルを保存しておくのが好ましくない場合は，該当する画像ファイルを読み込んでいる箇所で,

```
\includegraphics[bb={0 0 595 842}]{filename.jpg}
```

とすれば〈*filename*〉.bb がなくても良いことになります．使用する画像のファイル名の〈ファイル名〉. **拡張子**は ‘filename.png’ のように〈*8* 文字〉.3 **文字**したほうが無難なようです．

5.4　DVI を PDF にその 2

平田俊作氏と Cho Jin Hwan 氏が中心となって活動している Dvipdfm*x* Project によって開発されている中国語，日本語，韓国語などにも対応した dvipdfm の拡張版 Dvipdfm*x* を使うことができます．Dvipdfm*x* Project ウェブページは

http://project.ktug.or.kr/dvipdfmx/

にあります．Dvipdfm*x* は中国語（Chinese），日本語（Japanese），韓国語（Korean），16 ビットエンコーディングの文字コード（Unicode など）にも対応しています．CID フォントの埋め込みによって日本語フォントなどを持っていない人でも日本語 PDF を表示できるようにもなっています．PDF のセキュリティ機能も使うことができます．基本的に dvipdfm の上位互換なので dvipdfm で可能なことは Dvipdfm*x* でも可能でしょう．ちなみに graphicx や pict2e パッケージでのオプションには

```
\usepackage[dvipdfmx]{graphicx,color}
```

ではなく

```
\usepackage[dvipdfm]{graphicx,color}
```

とするようにしてください．

Dvipdfm*x* で指定できる主なコマンドラインオプションは以下の通りです．

-S　PDF のセキュリティを有効にします．

-K〈数字〉　PDF のセキュリティのキービットを指定します．40 か 128 です． 標準で 40 です．

-P　PDF のセキュリティのレベルを設定します．

-vh　用紙サイズを指定する-p で使用できる用紙一覧を表示します．実にさまざまな用紙がすでに定義されています．

-p ⟨幅⟩,⟨高さ⟩　定義済みの ‘a4’ 以外にも，用紙のサイズを単位付きで ‘20cm,20cm’ のように指定することもできます．詳しくは

```
dvipdfmx -vh
```

として表示される情報を見てください．

Dvipdfm*x* の-P オプションによる PDF のセキュリティの設定はについては表 5.1 を見てください．0x04 から 0x20 までのビットにそれぞれ許可・不許可が割り当てられていま

表 5.1　Dvipdfm*x* でのセキュリティレベルの指定

ビット	印刷	改変	文字列などのコピー	注釈の追加
0x04	許可			
0x08		許可		
0x10			許可	
0x20				許可
0x28		許可		許可
0x3C	許可	許可	許可	許可

す．要は表中の 16 進数の値を 10 進数に直し，それを自分の設定したいレベルに合わせて，それぞれのビットを足したものを再び 16 進数に直せば良いのです．印刷（0x04）と文書の改変（0x08）だけを許可したいならばこのビットを 10 進に直して二つのビットを足します．すると 12 になるのでこれを 16 進に直してあげます．電卓などで計算すると ‘0x0C’ になりますから

```
dvipdfmx -S -P 0x0C input.dvi
```

とすれば良いことになります．さらに

```
dvipdfmx -S -P 0x28 input.dvi
```

とすると改変と注釈の追加だけを許可するようにできますし，特に制限を課さないならば

```
dvipdfmx -S -P 0x3C input.dvi
```

とするとパスワードによる保護と暗号化のみになるものと思われます．

5.4.1　もうちょっぴり使いこなす

例えば論文投稿や印刷所に渡すような PDF のデータを作成するときには Dvipdfm*x* を使うときにはちょっと注意が必要です．自分の環境で正常に印刷できても印刷所の環境によってはフォントがないなどでうまくできない場合があります．また低解像度のビットマップフォントが含まれている場合も受け付けれてくれないかもしれません．PDF に対して全てのフォントを埋め込むのも一つの手です．日本語などではなく欧文の場合は pdfLaTeX か pdfε-LaTeX を使いますと LaTeX から直接 PDF に変換できます．日本語など

のフォントを含むような原稿ですと pLaTeX で処理した DVI ファイルを Dvipdfm*x* で PDF に変換という形が手軽な方法だと思われます．Dvipdfm*x* は EPS などの PostScript ファイルを画像として LaTeX に張り込んでいる場合は，それらを Ghostscript の力を借りて PDF に取り込みますので Ghostscript の性能が結果に依存します．現段階では Dvipdfm*x* が PostScript を解釈するのは難しいそうです．日本語の処理もある程度できる Ghostscript のバージョン 7.07 を使うのが良いそうです．

Dvipdfm*x* の場合は基本的に PDF，JPEG，PNG，METAPOST 形式の画像しかサポートしておりませんので，EPS 形式の画像は何らかの形で PDF に変換してから取り込むことになります．この EPS ファイルは Ghostscript の ‘pdfwrite’ というデバイスを使って変換することがほとんどです．そのときに epstopdf か ps2pdf14 などを使います．epstopdf は PDF に EPS の BoundingBox を反映してくれます．ps2pdf 系を使う場合は PDF に BoundingBox がうまく反映されないので以下のようなシェルスクリプト eps2pdfs

```
#!/bin/bash
EPS=‘ls *.eps‘;
for fig in $EPS; do
   epstopdf $fig
   $f=‘basename $fig .eps‘
   grep "^%%BoundingBox:" $fig > $f.bb
done
```

を作成し PATH の通っている場所に複製したならば

```
▉ eps2pdfs
```

とすると同ディレクトリの EPS ファイルが全て PDF に変換されます．⟨*file*⟩.eps があったとすればこれは ⟨*file*⟩.pdf と ⟨*file*⟩.bb が作成されます．このようにして EPS から PDF に変換したファイルは LaTeX の原稿で

```
\documentclass{jarticle}
\usepackage[dvipdfm]{graphicx}
\begin{document}
\includegraphics{filename.pdf}
\end{document}
```

のようにして取り込むことができます．ちなみに ⟨*file*⟩.bb は

```
%%BoundingBox: 142 160 443 665
```

のような情報が出力されているでしょう．以上のようなシェルスクリプトを動作させることができない Windows ユーザーの方はフリーの Cygwin を導入されるのが良いと思うのですが^^;．この ⟨*file*⟩.bb にある四つの数値を原稿中で

```
\includegraphics[bb={142 160 443 665}]{filename.pdf}
```

と記述すると ⟨*file*⟩.bb はいらなくなります．

フォントの設定として$TEXMF/fonts/map/dvipdfm/base/ 近辺に cid-x.map というファイルがあります．cid-x.map は *CMap* **ファイル**と呼ばれ，端末などから

```
▉ kpsewhich -progname=platex -expand-path='$CMAPINPUTS'
```

とすると

```
.;/usr/local/share/texmf/fonts/cmap
```

のように Cmap ファイルの所在が分かります．ファイル cid-x.map の中に

```
rml  H Ryumin-Light
gbm  H GothicBBB-Medium
rmlv V Ryumin-Light
gbmv V GothicBBB-Medium
```

のように 'rml' や 'gbm' で始まる行があると思います．この記述をフォント名などに変更すると日本語のフォントに何を使うのかが指定できます．標準では日本語などのフォントを埋め込まないようになっていると思います．例えばみかちゃん氏によるフォント mikachanAll.ttc を日本語の明朝体に使い，MS ゴシックをゴシック体に使う場合の設定は次のようになります．

```
rml  H  :0:mikachanAll
rmlv V  :0:mikachanAll
gbm  H  :0:msgothic
gbmv V  :0:msgothic
```

5.4.2　PDF ファイルの操作

PDF ファイルは商用のプログラムを使わないと自由度の高い操作は難しいと思われます．簡単な操作ならば xpdf に付属するツールを使うと良いでしょう．xpdf に付属するツールを使うには xpdfrc という設定ファイルに以下のような設定をすると良いでしょう．

```
cidToUnicode    Adobe-Japan1    /Resource/Adobe-Japan1.cidToUnicode
unicodeMap      ISO-2022-JP     /Resource/ISO-2022-JP.unicodeMap
unicodeMap      EUC-JP          /Resource/EUC-JP.unicodeMap
unicodeMap      Shift-JIS       /Resource/Shift-JIS.unicodeMap
cMapDir         Adobe-Japan1    /Resource/CMap
toUnicodeDir                    /Resource/CMap
```

下記のプログラムは PDF ファイルにセキュリティ設定がなされている場合はパスワードを必要としたり，または全く機能しない場合があります．以下のプログラムは全て端末から操作します．

pdftops　PDF ファイルを PostScript ファイルに変換します．

pdfimages　PDF ファイルに含まれるビットマップ画像を指定したディレクトリに抽出します．あらかじめ出力するディレクトリを作成しておきます．

```
▉ pdfiamges filename.pdf dir/
```

とするとディレクトリ ‘dir’ に ppm 形式か pbm 形式の画像として抽出されますので適宜お望みの変換してください．

pdftotext　PDF ファイルの文章をテキストファイルに抽出します．フォントマップファイルを必要とします．ASCII コード中の標準的な文字でなければうまくいかないかもしれません．

pdfinfo　PDF ファイルの「文書情報」を表示します．

pdffonts　PDF ファイルに使われているフォント情報を表示します．フォント名やフォントの種類，フォントが埋め込まれているかなどが分かります．

例えば

```
▉ pdffonts file.pdf
```

とすると次のような情報が表示されます．

```
name                                 type            emb sub uni object ID
------------------------------------ --------------- --- --- --- ---------
Times-Roman                          Type 1          no  no  no       7  0
GothicBBB-Medium-Identity-H          CID Type 0      no  no  no       9  0
Helvetica                            Type 1          no  no  no      10  0
Ryumin-Light-Identity-H              CID Type 0      no  no  no      12  0
Times-Italic                         Type 1          no  no  no      13  0
FRZWWS+txsy                          Type 1C         yes yes yes     14  0
EPSMLX+t1xtt                         Type 1C         yes yes yes     15  0
Times-Bold                           Type 1          no  no  no      16  0
LEPUME+rtxmi                         Type 1C         yes yes yes     23  0
CACNFM+rtxsc                         Type 1C         yes yes yes     32  0
Helvetica-Oblique                    Type 1          no  no  no      65  0
UQXVYG+rtxr                          Type 1C         yes yes yes     66  0
```

‘name’ というのがフォント名であり ‘type’ というのが使用されているフォントの種類を示します．‘emb’ はそのフォントが埋め込まれているかどうか，‘sub’ はサブセット化されているかどうか，‘uni’ というのは Unicode マッピングされているかどうかを示します．詳しいことは PDF 関連の資料 [21] をご覧ください．

5.5　HTML への変換

LaTeX の原稿ファイルを HTML に変換するソフトウェアがあります．近年では自分が作成した文書を WWW 上で公開することが頻繁にあります．論文なども例外ではなく HTML で出力する必要に迫られる場合があります．Unix 系 OS であれば LaTeX2HTML, TtH などが有名です．Unix 系 OS であれば幾つかのライブラリを追加すれば使える状態にあるのではないかと思います．

5.5.1 TeX4ht

Eitan Gurari 氏が開発している TeX4ht は Unix 系 OS ならばパッケージとして用意されていると思います．Windows では角藤亮氏が必要なファイルを用意して下さっております．HTML への変換に必要なプログラムは NTT が開発した日本語 TeX である jTeX，画像編集プログラムの ImageMagick，TeX4ht 本体です．ImageMagick は Windows でもバイナリが用意されていますし，Unix 系 OS ならばパッケージに含まれていることが多いようです．使用方法は tex4ht パッケージを原稿のプリアンブルに

```
\usepackage[html,charset=Shift_JIS,png]{tex4ht}
```

のように記述します．次に端末などから

```
▉ ht jlatex filename
```

とすれば ⟨*file*⟩.html と数式や画像などの PNG ファイルが出来上がります．ht を実行するときのコマンドラインオプションで知っておくと便利なものに次のようなものがあります．

-dry-run　ht プログラムが呼び出しているプログラムがどのように実行されるかが分かります．

-cleanup　HTML ファイルを生成後に中途ファイルを削除します．

-output-name=⟨名前⟩　出力ファイル名を ⟨名前⟩ に指定します．

-output-dir=⟨ディレクトリ⟩　出力するディレクトリを ⟨ディレクトリに⟩ に指定します．すでに存在するディレクトリでないとできないかもしれません．

別の変換方法としてソースファイルで tex4ht を**読み込まずに**，そのまま端末から

```
▉ htlatex filename "html,charset=Shift_JIS,png"
```

としても変換できます．

日本語の文書クラスを使うときは j-article j-report，j-book を使うようにしてください．そのための下準備として$TEXMF/tex/generic/tex4ht/ などのディレクトリに移動し，

```
▉ cp article.cls j-article.cls
▉ cp report.cls j-report.cls
▉ cp book.cls j-book.cls
```

として j-classes 用に設定ファイルを複製します．そうすると

```
\documentclass[11pt]{j-report}
```

と使うことができます．

tex4ht を読み込むときのパッケージオプションとして以下のものを追加すると良いでしょう．

html　tex4ht のオプションで一番最初に指定するのは出力するファイルの形式です．HTML（*html*）や XHTML（*xhtml*） などの形式を指定します．

charset=⟨エンコーディング⟩　文字コードを ⟨エンコーディング⟩ で指定します．‘Shift_JIS’ と指定しても一部の半角英字が正しく表示されません．

fonts+　標準のフォント設定では少し寂しいものがあるときは直接フォントをウェブブラウザに指定します．該当フォントがない場合は代替フォントに置き換わります．

fn-in　標準では脚注や傍注がおそらく別ページに出力されますが，このオプションを使うとページ最下部に出力されるようになります．

png　標準での画像出力形式は GIF（*gif*）になっていると思いますが，GIF の場合はバグなのかどうか分からないが不正な GIF が生成されることもあるので PNG にしたほうが良いでしょう．ただし閲覧者のウェブブラウザが PNG 形式の画像を表示できるかどうかは分からないので注意が必要です．ここ最近のブラウザならば PNG は表示できると思われます．他に JPEG（*jpg*）も指定できます．

imgdir:⟨ディレクトリ⟩*/*　標準では画像は HTML ファイルと同じディレクトリに出力されるのでこのオプションを指定して ⟨ディレクトリ⟩ を指定します．最後のスラッシュは必須と思われます．

pic-m　数式を画像化するオプション．tex4ht は画像化しなくても良いと思われる部分は画像にしません．われわれ日本語圏の人間にはそれでは都合が悪いことがあるので，苦渋の選択で全ての数式を画像化します．もっと強力に画像化するときは *pic-m+* を使います．`equation` などの数式環境全体を画像化するならば *pic-equation*，*pic-eqnarray*，*pic-matrix*，*pic-array*，*pic-align* などを使います．

pic-eqnarray　併用するパッケージによっては `eqnarray` 環境が正しく認識されないためかタイプセットできないので `eqnarray` 環境だけは画像化するように設定したほうが無難かもしれません．

⟨数字⟩　1 から 4 までの数字を指定して，出力 HTML ファイルのページを階層ごとに区切ります．LaTeX での見出しの階層に従って区切られます．

section+　通常は目次からリンクを辿りますがこのオプションが指定されている場合は見出しから目次に戻ることができます．

next　DVI 形式や PDF 形式のファイルは連続的にページが続いています．しかし HTML 形式でファイルを出力すると不連続になりますので，連続的に次のページへ進むためのリンクを作成します．

htm　出力 HTML 形式のファイル名を ⟨*8* 文字⟩`.htm` とします．他の OS との互換性を考慮するならば必要かもしれません．滅多にないと思いますが，例えば出力された HTML ファイルを ISO9660 フォーマットの CD-R に書き込むときなどに使えます．

以上のような設定をプリアンブルに

```
\usepackage[html,charset=Shift_JIS,fonts+,fn-in,png,imgdir:images/,
pic-m,pic-eqnarray,info]{tex4ht}
```

とすると良いでしょう．tex4ht は一番最後に読み込むようにするのが基本です．

他にも使用するパッケージがあるならば tex4ht を読み込む前に

```
\usepackage[dvips]{graphicx,color}
\usepackage{url}
```

などとしてください．hyperref はちょっと喧嘩するみたいですので

```
\usepackage{その他のパッケージ}
\usepackage[オプション]{tex4ht}
\usepackage[tex4ht]{hyperref}
```

とすると良いでしょう．

標準では以下の文字が日本語環境だと化けます．

```
\S \P \pounds \OE \ae \AE \aa \AA \ss \l \L
\o \O \i \j ?‘ !‘ \textvisiblespace \textless
\textgreater
```

文字コードの iso-8859-1 で画像にしなくても良い文字なので，文字化けというか日本語環境では表示可能ではありません．uhtlatex を使って Unicode フォントに置き換え，欧文フォントも Unicode にすると表示できると思われます．

日本語処理でひとつ問題となるのは余計な部分に入る半角空白です．この半角空白は tex4ht が文字列の処理を基本的に行単位で行い，その行を段落タグ ‘<P>’ で閉じてしまうからです．欧文の場合は適切な区切りで文字が改行されるのでこの方法でも良いのでしょうが，和文の場合はこれではいけません．が，適当な解決策は私には分かりません．

既存の画像を挿入したいときは\Picture 命令を使います．

\Picture[⟨代替文字⟩]{⟨画像ファイル名⟩}

ある範囲を画像化するときは\Picture+ 命令と\EndPicture 命令で囲みます．

\Picture+{⟨出力ファイル名⟩}{⟨要素⟩}\EndPicture

例えば hoge.jpg というファイルが存在し，これを HTML ファイル中に貼り付けるときは

```
\Picture[hoge の画像です]{hoge.jpg}
```

とします．tabular 環境などの表全体を画像化するならば

```
%\usepackage[html,png]{tex4ht}
\Picture+[画像にした表です]{mytable.png}
\begin{tabular}{|c|c|c|}
 \LaTeX2.09& \LaTeXe& \LaTex3\\
\end{tabular}
\EndPicture
```

となります．

HTML タグを直接出力ファイルに埋め込むには\HCode 命令を使います．HTML タグの強制改行や水平線を入れるならば

```
\HCode{<HR><BR><BR><BR><BR>}
```

という使い方もできます．

hoge 環境を使用しておりその環境を丸ごと画像にしたいならば document 環境の中で

```
\ConfigureEnv{hoge}
   {\IgnorePar\EndP\Tg<div class="pic-hoge">\Picture*{}}
   {\EndPicture\Tg</div>}{}{}
\Css{div.pic-hoge {スタイルシート}}
%\ConfigureEnv{hoge*}% これは適宜記述してください．
%   {\IgnorePar\EndP\Tg<div class="pic-hoge-star">\Picture*{}}
%   {\EndPicture\Tg</div>}{}{}
%\Css{div.pic-hoge-star {スタイルシート}}
```

のような設定をすると hoge 環境が画像化されます．picture 環境などの LaTeX で標準の描画用の環境は自動的に画像として出力されます．

tex4html の二つ目の引数に t4ht に渡す引数を書くこともできます．これは

```
▉ tex4html test "-p"
```

のような指定をして画像を生成しないように挙動を変えることもできます．

ハイパーリンクを作成するには\Link 命令も使えますが，個人的には hyperref か url パッケージを用いたほうが気持ちが良いでしょう．

```
%\usepackage[html,charset=Shift_JIS]{tex4ht}
%\usepackage[tex4th]{hyperref}
\href{http://www.google.co.jp}{Google}は検索エンジンです．
Google を参照するにはウェブブラウザのアドレス欄に
 \begin{quote}
  \url{http://www.google.co.jp}
 \end{quote}
と打ち込んで移動してください．
```

他にも長島順清氏のページ

http://osksn2.hep.sci.osaka-u.ac.jp/~naga/

や玉木広氏さんのページ

http://pantodon.shinshu-u.ac.jp/

も参考になることでしょう．

pTeX や pLaTeX に依存するパッケージなどは使用できません．パッケージ ⟨*file*⟩.sty の先頭に

```
\NeedsTeXFormats{pLaTeX2e}
```

と書かれている場合は pTeX 依存ですから処理するのが難しくなります．

第 6 章
コマンドとマークアップ

LaTeX はページ記述プログラミング言語です．ですがマークアップ言語としての特性を活かした使い方を解説した説明をあまり見かけません．そこでまずはマークアップ言語とは何なのか，マークアップで何が実現できるのか，それを LaTeX でどのように実現するのかという基本的な部分を紹介します．

6.1 マークアップ言語とは？

1980 年代のことでしょうか，当時文書記述言語としてマークアップ言語というものが脚光を浴びたそうです．文書に対して入れ子型の論理構造を与えることによって汎用性を持たせ，人間が直接理解できる文書の記述に関して研究がなされたそうです．その中でもウェブページを記述する言語して HTML（Hyper Text Markup Language）というものが出現したそうです．これは Tim Berners-Lee 氏が始め開発し，今では W3C が管理しているページ記述言語です．現在は XHTML（Extended Hyper Text Markup Language）へと進化し，統一化が図られています．LaTeX も HTML や XHTML と同じようにマークアップ方式を採用しているページ記述言語です．

6.2 記号とコマンド

LaTeX はコンピュータプログラムですから，人間の意志を理解するためには何か特別な命令を人間から受け付けないとなりません．

「もしかしてこの辺は**太字**にしたいのですか？」

などと対応することはありません．皆無というわけではありませんが，基本的にこちらから命令するまで何もしない性格です．そのため原稿には**コマンド**と呼ばれる特別な記号の綴りを使ったり，いくつかの記号に**特別な意味**を持たせます．ですから LaTeX での記号の意味とコマンドの役割を知っておきましょう．

6.2.1 記号の分類

いきなり TeX/LaTeX の記号の分類を考えるよりも，普段使用している日本語や英語の記号と言語を考えましょう．私たちはコンピュータが理解するのが難しい言語を使ってい

ます．特に日本語はコンピュータが理解しづらいものとなっています．それでも私たち人間はそれらを文脈などを道筋に意味を読み取ります．

「今日は 1,000 円を持って，砂糖さんの商店に塩を買いに行った.」

という 1 文を考えると，人間はすぐに「1,000 円」のコンマ ‘,’ を「お金の 3 桁目に入れる区切り」，最後の句点「.」を「文の終わり」であると理解できます．私たちが複雑な日常文を理解できる理由として，文の中に必ず**文法**が存在するということです．

人間はある程度曖昧な表現を見つけてもそれを柔軟に理解できますが，コンピュータは何しろ 1 と 0 しか分からない融通の利かない機械ですから**正規文法**しか理解できません．正規文法については別の機会に紹介しようと思いますが，とにかく人間の使う日本語とは違いきっかりしているということです．この正規文法に基づいた**正規言語**を理解するのがコンピュータプログラムなわけです．TeX/LaTeX も正規言語しか理解できないのでユーザから入力される記号には全て明確な意味を持たせる必要があります．曖昧な意味を持たせることはできません．

LaTeX ではユーザが出力したい意味を理解するために全ての記号に LaTeX なりの意味を割り当てています．人間がキーボードから ‘<’ という記号を入力しても数学の比較演算子とは理解してくれません．‘$<$’ としなければ「ここからここは数式であり，‘<’ は比較演算子として使う.」という意味を理解してくれません．そのため LaTeX に入力を与えるユーザーは LaTeX の文法を覚える必要があります．詳しく覚える必要はありませんが

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

という 10 個の記号には特別な意味があることを覚えてください．それぞれの記号の使い方はその時々に説明しますので，今はざっと 10 個の記号が特別な意味を持ち，英文字と漢字，それに平仮名や片仮名だけが普通の単語を作るための文字として理解してくれると思ってください．

6.2.2　コマンド

テキストを入力していると ‘<’ というキーボードからの入力が ‘¡’ になってしまいます．これは一体どういうことでしょうか．考えてみると ‘<’ という入力は ‘<’ という記号を出力するという命令ではなく別の命令，‘¡’ を出力するという命令に割り当てられていると考えられます．さらに\% のようなバックスラッシュ（円）の次に記号が来るようなコマンドも存在します．ここで LaTeX のコマンドは「バックスラッシュと文字列」という話ではないことが分かります．正確には「バックスラッシュと記号の綴り」を**コントロールシークエンス**と呼び，特殊記号 1 文字を**コントロールシンボル**と呼びます．LaTeX におけるコマンドは大きく分けると三つに分類できます．

コントロールシークエンス　バックスラッシュ ‘\’（‘¥’）と記号の綴り．**制御綴り**と訳されることもあります．これを本書では狭義の**コマンド**として表現しています．

コントロールワード　バックスラッシュと英字の綴り．例えば ‘\section’ など．

コントロールシンボル　バックスラッシュと英文字以外の綴り．例えば ‘\3’ とか ‘\#’ など．

コントロールスペース　バックスラッシュとスペース一つの綴り．‘\␣’ のこと．

特殊記号　特別な意味を持つ記号．**予約文字**と呼ばれることもあります．例として ‘{’, ‘$’ など．

英数字など　バックスラッシュの付かない普通の文字列．

現段階では大きく分けると

- バックスラッシュと文字列の綴り．
- 特殊な記号．
- 普通の文字列．

の三つがあることを理解してください．本冊子では制御綴り（コントロールシークエンス）のことを**コマンド**と呼び**命令**，**宣言**，**環境**の三つに分類します．

命令　特定の処理がそのときに実行されるコマンド．他の参考書ではこの命令のことを**コマンド**と呼ぶことが多いようです．**引数**を取ることがあり，その引数のことを**要素**と呼んだり，**オプション**と呼んだりします．例として\maketitle や\section などがあります．

宣言　特定の処理がそれ以降継続して行われるコマンド．処理の適用される範囲を限定する（グルーピング）こともできる．引数をとることは稀．よく宣言のことも**命令**や**宣言方命令**とか**宣言型コマンド**と呼ばれます．例として\ttfamily．宣言型のコマンドは命令に比べると少ないので，本冊子でも断り書きとして宣言型コマンドと呼ぶことが多いです．

環境　\begin{⟨何々⟩}と\end{⟨何々⟩}によって要素を囲むコマンド，または囲まれている領域のこと．引数を取ることがあります．例として document 環境などがあります．

6.2.3　コマンドの定義

LaTeX の原稿では新しい命令などの定義をすることができます．

```
\newcommand{⟨命令⟩}[⟨整数⟩][⟨標準値⟩]{⟨定義⟩}
\renewcommand{⟨命令⟩}[⟨整数⟩][⟨標準値⟩]{⟨定義⟩}
```

\newcommand についてですが，この命令によって，まだ定義されていない ⟨命令⟩ を新規に定義することができます．

```
\newcommand{\example}{これは例です．}
```

として，本文中で{\example}と記述すると

これは例です．

という出力になります．さらに

```
\newcommand{\example}[2]{#1 は#2 です．}
```

として，本文中で\example{ボブ}{背が高い}と記述すると，

> ボブは背が高いです．

という出力になります．さらにこの\example 命令に任意引数があっても良いことを宣言するためには次のようにしますが，任意引数も引数の総和に勘定します．

```
\newcommand{\example}[2][標準]{任意引数は #1
で，必須引数は #2 です．}
\example{ほげ}\\ \example[オプション]{ほげ}
```

任意引数は 標準 で，必須引数は ほげ です．
任意引数は オプション で，必須引数は ほげ です．

このように任意引数や必須引数の定義なども，\newcommand 命令を使うことにより実現できます．定義の中で引数は ‘#’⟨n⟩ として扱い，1 から 9 までの整数を使えます．このような定義が何に使えるのかと少し疑問に思う方がいるかも知れませんが，以下の例題をご覧ください．

```
\newcommand{\seq}[2][n]{%
  \{#2_{0},\ldots,\,#2_{#1}\}}
ぶっちゃけ$\seq{a}$って$\seq[k]{x}$だよね．
```

ぶっちゃけ $\{a_0, \ldots, a_n\}$ って $\{x_0, \ldots, x_k\}$ だよね．

\newcommand では任意引数を一つしか設けることができませんが，引数は合計 9 個まで使うことができます．\renewcommand では一度定義した命令を再度定義することができます．

さらに通常 LaTeX でよく見かける環境型のコマンドの定義に関しては以下の四つの命令が使えます．

```
\newenvironment{⟨命令⟩}[⟨整数⟩][⟨標準⟩]{⟨始め⟩}{⟨終わり⟩}
\renewenvironment{⟨命令⟩}[⟨整数⟩][⟨標準⟩]{⟨始め⟩}{⟨終わり⟩}
```

\newenvironment では環境の始めの部分と終わりの部分を定義して，新たに環境型の命令を作成します．引数に関する扱いは\newcommand と同じです．\renewenvironment については一度定義した環境型のコマンドを再度定義する機能があります．中央揃えして書体を強調したい環境は次のように cemph のように作成します．

```
\newenvironment{cemph}%
  {\begin{center}\begin{em}}%
  {\end{em}\end{center}}
ここの文章は通常通り出力され，
\begin{cemph}
この中の文章は中央揃えで強調表示
\end{cemph}
されましたか？
```

ここの文章は通常通り出力され，

この中の文章は中央揃えで強調表示

されましたか？

\providecommand{⟨命令⟩}[⟨整数⟩][⟨標準値⟩]{⟨定義⟩}
\DeclareRobustCommand{⟨命令⟩}[⟨数値⟩][⟨標準⟩]{⟨設定⟩}

という命令も使えます．この\providecommand はすでに命令が定義済みならば何もせず，もし定義されていないならば指定された内容の通りに命令を定義することができます．\DeclareRobustCommand を使った場合は頑丈な命令を作成できます．

▹ **例題 6.1** \DeclareRobustCommand を使った例を示します．次のファイルをタイプセットし，その結果を吟味してください．

```
\documentclass{jarticle}
\DeclareRobustCommand{\joubuyen}{{\ooalign{Y\crcr\hss=\hss}}}
\newcommand{\moroiyen}{{\ooalign{Y\crcr\hss=\hss}}}
\begin{document}
\tableofcontents
\section{もろい{\moroiyen}は壊れやすい}
どうですかねぇ。
\section{丈夫な{\joubuyen}は壊れにくい}
大丈夫ですか？
\end{document}
```

この例では 1 回目のタイプセットで

```
! Illegal parameter number in definition of \reserved@a.
<to be read again> \crcr
l.6 \section{もろい{\moroiyen}は壊れやすい}
?
```

と表示されエラーとなります．さらに 2 回目のタイプセットでは目次ファイル ⟨*file*⟩.toc が作成され，それが読み込まれるのでさらにエラーとなります．⟨*file*⟩.toc と ⟨*file*⟩.aux に何が記述されているのかを吟味してください．⟨*file*⟩.aux には

```
\relax
\@writefile{toc}{\contentsline{section}{\numberline{1}もろい
  {{\lineskiplimit -\maxdimen \unhbox \voidb@x \vtop {\baselineskip
  \z@skip \lineskip .25ex\everycr{}\tabskip \z@skip \halign {##\crcr
  Y\crcr \hss =\hss \crcr }}}}は壊れやすい}{1}}
\@writefile{toc}{\contentsline{section}{\numberline{2}丈夫な
  {\joubuyen}は壊れにくい}{1}}
```

などと出力されており，凄いことになっています．さらに ⟨*file*⟩.toc には

```
\contentsline{section}{\numberline{1}もろい{{\lineskiplimit
  -\maxdimen \unhbox\voidb@x \vtop {\baselineskip \z@skip
  \lineskip .25ex\everycr{}\tabskip \z@skip \halign {####\crcr
  Y\crcr \hss =\hss \crcr}}}}は壊れやすい}{1}
\contentsline{section}{\numberline{2}丈夫な{\joubuyen}
  は壊れにくい}{1}
```

とあります．これが\DeclareRobustCommand で\moroiyen 命令を定義しなかった結果となります．LaTeX の命令を何か別のファイルに書き出す場合は丈夫な命令として定義していた方が無難なようです．ただし，次のように\protect 命令を使った場合はうまく行くことを確認してください．

```
\section{もろい{\protect\moroiyen}は壊れやすい}
```

\protect を使うと\moroiyen が壊れずに ⟨*file*⟩.toc や ⟨*file*⟩.aux に出力されていることも確認してください．

6.2.4　文字やコマンドの区切り

私たち人間はある文や節の区切りをどのように判断しているのでしょうか．一つは文と文のあいだや単語と単語のあいだに挿入される空白です．空白は文字列の区切りを示し，その空白には意味の区切りがあります．では節はどうでしょうか．例えば日本語では

「公園の中のベンチのうえ」

という名詞があったとすると，これはそれぞれ

「公園」「の」「中」「の」「ベンチ」「の」「うえ」

に分けて文を解釈し，**名詞**や**助動詞**などの**品詞**に分けることができます．

もう一つの例としてメールアドレスの場合を考えて見ましょう．メールアドレスはそもそもコンピュータ上で手紙のやり取りをするための住所ですからコンピュータが分かりやすい表現になっていますが，人間にも分かりやすい表記になっています．仮に

```
name@server.co.jp
```

というメールアドレスがあったとします．するとこれは

‘name’ ‘@’ ‘server’ ‘.’ ‘co’ ‘.’ ‘jp’

に分けられます．それぞれ

name　メールアドレスを使っている人の「名前」．
@　‘@’ は ‘at’ の意味でもあって，これ以降の文字は「住所」を表すことを示す．
jp　その人の「国」を表す．
co　その人がどんな「地域（組織）」に所属しているのかを表す．
server　地域の中のどこにいるのかをあらわす住所．
.　住所を区切るために使われている．

という意味合いを持っています．住所の区切りが空白ではなくピリオドなのは仕方のないことです．コンピュータの世界ではなるべく文字列は空白を含んでいないほうが処理が行いやすいのです．さて，これはどのようにして区切りを見つけたのでしょうか．メールア

ドレスの例では ‘@’ や ‘.’ を文字の区切りとして住所を判定しています．LaTeX でも同じようなことをやっています．

では文字列としてのコマンドが LaTeX でどのように解釈されるのかを考えてみましょう．ここでは引数を取る命令を考えてみます．まずは，

```
\newcommand{\hoge}[1]{あ，#1 だよ}
\hoge ほほげほげ．
```

という入力の出力を予想してみましょう．結果は「あ，ほだよほげほげ．」です．この結果から LaTeX では引数には何も指定がなければ 1 文字しか受け取らないということが分かります．

次に

```
\newcommand{\hoge}[1]{あ，#1 だよ}
\hoge {ほげほげ}，ほげ．
```

としてみましょう．結果は「あ，ほげほげだよ，ほげ．」になるでしょう．どうやら引数を波括弧 ‘{}’ で囲むとそれを一つの文字の塊として受け取るようです．

次に二つ引数を取る命令を定義して

```
\newcommand{\hoge}[2]{あ，#1 だよ，ほら#2}
\hoge       {ほげ}          ね．
```

とした場合の結果は「あ，ほげだよ，ほらね．」となるでしょう．もうお分かりですね．LaTeX の引数に複数の文字列を渡したいときは括弧でグルーピングすれば良いのです．

さて今度は英文字ではない ‘@’ を含む命令を定義してみましょう．

```
\newcommand{\h@ge}[2]{あ，#1 だよ，ほら#2}
\h@ge       {ほげ}          ね．
```

これをタイプセットするとエラーになるので Enter キーを 4 回ほど押すと

```
! Missing number, treated as zero.
<to be read again>
                g
l.7 \newcommand{\h@ge}[2]{あ，#1だよ,ほら#2}
?
! You already have nine parameters.
\reserved@a ...def \expandafter \h \reserved@b #10
                                                  g{
l.7 \newcommand{\h@ge}[2]{あ，#1だよ,ほら#2}
?
! You can't use ‘macro parameter character #’ in horizontal mode.
<argument> あ，##
               1だよ,ほら##2
l.7 \newcommand{\h@ge}[2]{あ，#1だよ,ほら#2}
?
! You can't use ‘macro parameter character #’ in horizontal mode.
<argument> あ，##1だよ,ほら##
                           2
l.7 \newcommand{\h@ge}[2]{あ，#1だよ,ほら#2}
?
```

というエラーが表示されるでしょう．これは一体どういうことでしょうか．メールアドレスの例を思い出してほしいのですが，LaTeX ではどうやら ‘@’ をコマンドの区切りとして解釈しているようです．そして引数の受け取り方を考えると

```
\newcommand{\h}[2]{あ，#1 だよ，ほら#2}
\h@ge        {ほげ}              ね．
```

とした場合の結果は「あ，@だよ，ほら ge ほげね.」となることでしょう．

このことから LaTeX においての命令の定義には英字のみにすることが求められるようです．そして英字以外の文字列は，そこをコマンドの区切りとして英字以外の文字列を引数として受け取るということです．

この文字の分類を利用して LaTeX ではマクロの中において特別な処理をしています．マクロは容易に変更してもらっては困るのでユーザからそのマクロを簡単に変更されないようにしています．その方法の一つとしてマクロの中ではアットマーク ‘@’ を英字と同じ分類として扱うのです．‘@’ を英字と同じ分類にすると，そこでコマンドは区切られないので

```
\newcommand{\h@ge}[2]{あ，#1 だよ，ほら#2}
```

のような定義ができるわけです．そして

```
\newcommand{\h@ge}[2]{あ，#1 だよ，ほら#2}
\newcommand{\hoge}{\h@ge}
```

という定義がマクロの中では可能なので，ユーザーから\hoge 命令の実態を隠すことができます．これは開発者と使用者の分離を行うために仕方がないのですが，逆に使用者が自分の思い通りに命令をカスタマイズできないとか，様々な不満があるようです．

実際ヘッダーやフッターを自分流にカスタマイズしたいときはそれらの命令に ‘@’ が含まれているために変更できない，という事態に陥ります．マクロで行っていること，‘@’ を英字と同じ分類にしてコマンドを定義するためには

\makeatletter （‘@’ を英字と同じ分類にする.）
\makeatother（‘@’ を違う分類にする.）

という二つの命令を使います．この命令の中身を見てみると

```
\def\makeatletter{\catcode`\@11\relax}
\def\makeatother{\catcode`\@12\relax}
```

となっています．どうやら ‘@’ の**カテゴリーコード**というものを 11 にすると英字と同じになり，12 にすると違う分類になるようです．このような記号の分類を**カテゴリーコード**と呼びます（表 6.1 参照）．

そのため何かマクロの中のコマンドを自分で変更を加えたいときは

```
\documentclass{jsarticle}
\makeatletter
\newcommand{\h@ge}[2]{あ，#1 だよ，ほら#2}
```

表 6.1　カテゴリーコードの一覧

カテゴリ	意味	標準での割り当て
0	エスケープ文字	\ （¥）
1	グループの開始	{
2	グループの終わり	}
3	数式モードの制御	$
4	配列の要素の区切り	&
5	行末文字	〈改行〉（0x0D）
6	パラメータ文字	#
7	上付き文字	^
8	下付き文字	_
9	無視される文字	なし [*1]
10	空白	␣
11	英文字	A···Z と a···z
12	そのほかの文字	(! ? 1 2 @など
13	アクティブ文字	~
14	コメント文字	%
15	無効文字	〈デリート〉（0x7E）
	以下三つは日本語 TeX のもの	
16	第 1・第 2 水準の漢字	亜，丼など
17	かな，全角アルファベット	あ，ア，ａ，Aなど
18	その他の全角記号	＋，【など

[*1] 標準では割り当てられていない

```
\newcommand{\hoge}{\h@ge}
\makeatother
\begin{document}
\hoge{ほげ}{げほ}.
\end{document}
```

のように ‘@’ を含む箇所を\makeatletter と\makeatother で囲んであげます.

6.2.5　コマンドの引数

引数と取るコマンドに対して文字列を渡した場合の挙動は予想しやすいと思います．ではコマンドに対して制御綴りを渡した場合はどうなるでしょうか.

```
\newcommand{\twoarg}[2]{一つ目の#1，二つ目の
#2}
\twoarg a b とか\twoarg{ほげ}{げほ}とか，
さらに\twoarg{\LaTeX}{\LaTeXe}は？
```

一つ目の a，二つ目の b とか一つ目のほげ，二つ目のげほとか，さらに一つ目の LaTeX，二つ目の LaTeX 2ε は？

どうやら引数を取るコマンドに対してさらに制御綴りを引数に与えても良いようです．では次の場合はどうでしょうか．

```
\newcommand{\twoarg}[2]{一つ目の#1，二つ目の
#2}
ほほう\twoarg\LaTeX\LaTeXe とは粋だね．
では\twoarg\LaTeX2\LaTeXe はどうだね？
```

ほほう一つ目の LaTeX，二つ目の LaTeX 2ε とは粋だね．
では一つ目の LaTeX，二つ目の 2LaTeX 2ε はどうだね？

これは 6.2.4 節でやったことが含まれています．‘LaTeX’ と ‘2’ のあいだで語が区切られて解釈されているので二つ目の引数に ‘2’ だけが渡されています．

6.2.6　コマンド定義中の空白の扱い

コマンドを定義するための命令\newcommand の中では空白が無視されるわけではありません．次の出力の違いを見比べてください．

```
\newcommand{\kuhaku}[1]{ほげと　 #1 だ，}
\newcommand{\Kuhaku}[1]{ほげと#1 だ．}
\kuhaku{だめぽ} \Kuhaku{だめぽ　 }
```

ほげと だめぽだ， ほげとだめぽ だ．

この例では空白が二つ以上続いており，それらが一つの空白として解釈されて，引数と文のあいだに余分な空白が挿入されています．正しくは

```
\newcommand{\kuhaku}[1]{ほげと#1 だ，}
\newcommand{\Kuhaku}[1]{ほげと#1 だ．}
\kuhaku{だめぽ}\Kuhaku{だめぽ}
```

ほげとだめぽだ，ほげとだめぽだ．

のように定義したほうが良いようです．

6.3　グルーピング・入れ子構造

TeX/LaTeX では変数のスコープ（有効範囲）というプログラミングでは当たり前の機能を持っています．これはプログラミングをやったことのない人には馴染みの薄いものなので詳しく説明しましょう．

まず変数には「限られた範囲だけ有効」な**局所変数**と「全ての範囲で」有効な**大域変数**の 2 通りがあります．LaTeX においてもこれは重要な話で，この有効範囲（**スコープ**）を決めるのが波括弧です．

方言のように地方によってアクセントや言葉が違うことがありますが，LaTeX でも地方（範囲）を分けて言葉（定義）の使い分けができます．以下の例題をご自分で試してみてください．

```
\newcount\hoge%          新規に変数の調達
\the\hoge%               変数の値を表示
{%                       グループ 1 の開始
   \hoge=3 \the\hoge%    \hoge に 3 を代入して値を表示
```

```
    {%                      グループ 2 の開始
       \hoge=5 \the\hoge%   \hoge に 5 を代入して値を表示
    }%                      グループ 2 の終了
    \the\hoge%              グループ 2 で hoge に 5 を代入しても？
}%                          グループの終了
\the\hoge%                  ここでの値は？
\hoge=6 \the\hoge%          \hoge の値は?
```

\newcount は新規にカウンタ（レジスタ）を用意し，\the はカウンタなどの値を表示する命令です．さて，以上の入力から「035306」という値を得ることができたでしょうか．このことを考えると「内側の括弧で代入した値は外側の括弧の領域に影響しない．」ということであり，「外側で値を変更しても内側の括弧にある値まで変更できない．」ということです．まさに「グループ」を作ってその中で好き勝手にやっているわけです．

次に書体変更の宣言でどのように書体が変更されるのかを見てみましょう．今回はファミリーを変える\ttfamily とシェープを変える\itshape，そして普通の書体に戻す\normalfont という三つの宣言を使います．

```
roman {\ttfamily tt {\itshape it} tt
\normalfont it} roman
```

roman tt *it* tt it roman

ここでおやっと気づいていただきたいのは\ttfamily という宣言が二つの括弧の中にまで影響しているという点です．先ほどの変数の代入ではこのようにはなりませんでした．どうやら書体の宣言は，その宣言をした場所から内側の括弧までもが有効範囲になっているようです．これは現在の LaTeX の仕様です．宣言ではなく命令としても結果は同じになります．

```
roman \texttt{ tt \textit{it} tt
\normalfont it} roman
```

roman tt *it* tt it roman

しかし\normalfont 命令を使うとタイプライタ体の有効範囲でもそこで通常の書体に戻ってしまいます．こう考えると影響を与えたくない括弧の内側の領域には\normalfont を使うと良いことになります．

```
roman {\ttfamily tt {\normalfont\itshape
it} tt} roman\par
roman \texttt{tt {\normalfont\textit{it}}
 tt} roman
```

roman tt *it* tt roman
roman tt *it* tt roman

命令ではなく宣言型のコマンドのいくつかは括弧の内側まで影響するので，その属性を受けないようにするための工夫が必要になります．

6.3.1 大域化

書体の属性を変更する宣言型のコマンドはスイッチの変更をしているもので特殊なものでした．ここでは最初の大域変数と局所変数の二つをもう一度見直しておきましょう．

括弧の内側で定義した内容を外側でも有効にしたいときはどうすれば良いでしょう．また括弧の外側の定義を内側にも有効にする方法がないと不便でしょう．「用紙サイズ」のようにそう簡単に変わってほしくない変数は括弧の内側でも有効であってほしいものです．それらを括弧の内側でも有効にすることを**大域化**（**グローバル化**）と言います．TeX/LaTeX においても大域化のための\global という命令が用意されています．

```
\newcount\hoge \the\hoge
{% グループ 1
   \global\hoge=3  \the\hoge% 大域に代入
   {% グループ 2
      \the\hoge
      \hoge=5 \the\hoge
   }
   \the\hoge
}
\the\hoge
\hoge=6 \the\hoge
```

とした場合は ‘0 3 35 3 36’ となるでしょうか．3 行目で大域的に ‘3’ を代入していますので内側の括弧にも有効ですし，グループ 2 を出た後の数値も ‘3’ です．そしてグループ 1 を抜けた後でも ‘3’ が代入されています．以上のように\global を使うとそのグループの内側と外側の全ての処理に影響します．

6.4　宣言と命令の違い

例えば center 環境のコマンドを考えると，なぜ環境の内側では全ての行が中央揃えになるのでしょう．一つは

```
\begin{center}
```

によってグループが始まり，

```
\end{center}
```

によってグループが終わらせているために，どこからどこまでが中央揃えなのかが分かっているのでしょう．「**これをまさに中央揃えにしてください．**」と言うよりは「**ここからここまでを中央揃えにしてください．**」というコマンドのほうが都合が良いことに気づくでしょう．非常に長い文章の場合は\centering 命令を使い

```
\centering{A long long ago, a man was doing hoge hoge...}
```

とするよりも center 環境として

```
\begin{center}
 A long long ago, a man was doing hoge hoge...
\end{center}
```

としたほうが分かりやすいでしょう．そう考えるとコマンドには

宣言型コマンド 使用してからそれ以降ずっと有効なコマンド．環境型のコマンドに使われたり，単独で使われる．

命令型コマンド 使用した場所で有効なコマンド．通常は引数に与えられたものを処理する．

の二つがあることになります．

例として命令型の\textsf と宣言型の\sffamily を考えてみましょう．命令型の場合は

```
Roman. \textsf{Roman? \par This is sans serif family.} Roman!
```

のような使い方はできませんが，宣言型ならば新規に sffont 環境を定義できます．

```
\newenvironment{sffont}{\sffamily}{}
Roman.
\begin{sffont}
   Roman? \par This is sans serif family.
\end{sffont}
Roman!
```

Roman. Roman?
This is sans serif family. Roman!

宣言型のコマンドはそれ以降ずっと有効なので有効範囲を決めてあげます．\sffamily などの書体を変更するコマンドはグルーピングする必要があります．

```
Roman! {\sffamily sans serif family.}
Roman!
```

Roman! sans serif family. Roman!

今まで使ってきた\begin{〈何々〉}と\end{〈何々〉}というコマンドは，このグルーピングの作業をやってくれているのです．補足的なことですが

```
\begin{<何々>}  <要素> \end{<何々>}
```

というのは LaTeX の中で

```
{\何々 <始めの処理>  <要素> <終わりの処理>}
```

に変換されるので\sffamily のような宣言も

```
Roman?
\begin{sffamily}
This is sans serif family.
\end{sffamily}
Roman!
```

Roman? This is sans serif family. Roman!

とできます．こうすると特に長い文章が読みやすくなります．

6.5 相互参照

文章の論理構造を明確にしてくれるものの一つに**相互参照**があります．相互参照の仕方は参照したいものにラベルを貼り，挿入したい場所でラベルを参照するという二つの作業に分けられます．相互参照できる項目は以下の四つ程に限られています．

- 章節命令 （\section 命令など）
- 番号付き数式 （equation 環境など）
- float 環境の要素（図や表など）
- enumerate 環境内の個々の項目

要は通し番号のついているものには付けても良いようです．ラベルは単純に貼りたいものに\label 命令で

⟨参照したい要素⟩\label{⟨ラベル名⟩}

のようにします．参照の仕方にはその番号を参照する\ref とページを参照する\pageref の 2 通りがあります．

\ref{⟨ラベル⟩} （通し番号） \pageref{⟨ラベル⟩} （ページ番号）

参照の仕方は以下のようになります．通し番号を参照する\ref 命令は\section 命令のようなものを参照するときに非常に便利です．

```
%\section{相互参照}\label{sec:xr}
詳しくは\pageref{sec:xr}~ページの
\ref{sec:xr}~節で述べているのでそ
ちらを参照されたい．
```

詳しくは 91 ページの 6.5 節で述べているのでそちらを参照されたい．

相互参照や目次を作成しているときはタイプセットを 3 回程行う必要があります．ラベルの名前が重複しないように工夫することも必要です．

6.5.1　相互参照の仕組み

節（見出し）や図表には通し番号付けます．これは同じ名前の節（見出し）が同じページに存在しても区別できるという利点があります．そして節（見出し）を参照するときはその番号を示します．このような機能を実現するために LaTeX では**カウンタ**を使います．ユーザーが特にこのことを意識しなくても半自動的に番号付けなどをやってくれます．一応さわり程度にはその仕組みを説明します．

相互参照する対象が通し番号ですので，節なら節などの要素に応じたカウンタがあらかじめ用意されています．LaTeX では表 6.2 の通りにあらかじめ定義されているカウンタがあります．カウンタは「素の番号」と実際に出力すべき「表示用の番号」と「参照用の文字列」の三つの要素を持っています．例えば

```
\newcounter{section}[chapter]
```

というのは

```
\newcount\c@section % 素の番号用
\def\thesection{\thechapter.\c@section}% 表示用
\def\p@section{\thechapter.\c@section}% 参照用
```

表 6.2 あらかじめ定義されているカウンタ名

カウンタ名	割り当て
part	部見出し
chapter	章見出し
section	節見出し
subsection	小節見出し
subsubsection	少少節見出し
paragraph	段落見出し
subparagraph	小段落見出し
page	ページ番号
equation	式番号
figure	図見出し
table	表見出し
footnote	脚注番号
mpfootnote	minipage 環境中の脚注番号
enumi	一つ目の階層の enumerate 環境の番号
enumii	二つ目の階層の enumerate 環境の番号
enumiii	三つ目の階層の enumerate 環境の番号
enumiv	四つ目の階層の enumerate 環境の番号

などの処理と同じことになります．\newcounter とは新しいカウンタを定義するための命令です．

```
\label{hoge}, \ref{hoge}
```

としたときの動作を実際に見るのが早いと思います．ファイル名を hoge.tex として

```
\documentclass{jsarticle}
\begin{document}
\newcounter{hoge} \thehoge
\refstepcounter{hoge} \thehoge
\end{document}
```

というファイルを作成し，1 回だけタイプセットしてください．ここで\refstepcounter はカウンタの値を一つ増やす命令で\thehoge はカウンタ hoge の値を表示するための命令だと思ってください．結果として端末には `No file hoge.aux` というメッセージが表示されるはずです．この段階で hoge.aux を more か less コマンドで見ると

```
\relax
```

としか出力されていません．\refstepcounter 命令だけでは参照できる状態にはないようです．次に

```
\refstepcounter{hoge} \thehoge
```

の 1 行に対して

```
\refstepcounter{hoge} \thehoge \label{cnt:hoge}
```

のように書き足して 1 回だけタイプセットを行ってください．すると端末には

```
Label(s) may have changed.  Return to get cross-ferecenses right.
```

という LaTeX の警告が表示されます．ラベルが変更されたと思われるので解消しなさいと言われています．ここで hoge.aux を見ると

```
\relax
\newlabel{cnt:hoge}{{1}{1}}
```

という新しい情報が出力されています．これで cnt:hoge という名前のラベルを参照する準備ができていることが分かります．

```
\newcounter{hoge} \thehoge
\refstepcounter{hoge} \thehoge \label{cnt:hoge}
```

という 2 行に対して

```
ref=\ref{cnt:hoge}, page=\pageref{cnt:hoge}
```

の 1 行を付け足して 1 回だけタイプセットすると端末に警告は表示されません．hoge.aux の内容も変わっていません．さて，ここで LaTeX に意地悪をして

```
\setcounter{page}{100}
\newcounter{hoge} \thehoge
\refstepcounter{hoge} \thehoge \label{cnt:hoge}
ref=\ref{cnt:hoge}, page=\pageref{cnt:hoge}
```

として，ページ番号を ‘100’ にしてから 1 回だけタイプセットするとどうなるでしょうか．再び端末には

```
Label(s) may have changed.  Return to get cross-ferecenses right.
```

という警告が表示されてしまいました．そして hoge.aux のファイルの中身は

```
\relax
\newlabel{cnt:hoge}{{1}{100}}
```

に変更されています．

以上の結果から分かるように ⟨*file*⟩.aux には相互参照の情報が保存されていることが分かりました．LaTeX ではそれらを前回のタイプセットの結果が保存されていた ⟨*file*⟩.aux と新しい相互参照の情報を比較してユーザーに対しても警告を出しているということが分かります．\ref 命令と\pageref 命令は相互参照用の情報からカウンタ番号やページ番号をラベルによって知ることができるということです．もう少し詳しい話（**\r@ラベル**）もいつの日にか．

6.5.2　カウンタ

章見出しやページには通し番号が振られています．これらは *LaTeX* **カウンタ**によって制御されています．カウンタはプログラミング言語で言えば int 型（整数）の変数です．カウンタ変数の仕組みや制御の方法を少しは知っておいたほうが後々便利です．この章では変数の基礎を説明します．

例えば jsbook クラスで章（\chpater）の下の階層の節（\section）用のカウンタを定義するには

```
\newcounter{section}[chpater]
```

とします．このようなカウンタの定義には次の命令が使えます．

\newcounter{⟨カウンタ名⟩}[⟨親カウンタ名⟩]
\setcounter{⟨カウンタ名⟩}{⟨数値⟩}
\addtocounter{⟨カウンタ名⟩}{⟨数値⟩}
\stepcounter{⟨カウンタ名⟩}
\refstepcounter{⟨カウンタ名⟩}
\value{⟨カウンタ名⟩}

\newcounter でカウンタを新設します．\setcounter は数値を代入し，\addtocounter は数値を足し，\stepcounter はカウンタの値を一つだけ増やします．\refstepcounter はカウンタを後から参照できるようにラベル用が用意されます．\stepcounter と \refstepcounter によって親カウンタが増えるとその子であるカウンタは 0 にリセットされます．\value はカウンタから親カウンタの値や文字列などを取り除いた純粋なカウンタの値が得られるコマンドです．

カウンタの表示形式は変更するものに以下があります．

\arabic{⟨カウンタ名⟩}	（1, 2, 3, ...）
\roman{⟨カウンタ名⟩}	（i, ii, iii, ...）
\Roman{⟨カウンタ名⟩}	（I, II, III, ...）
\alph{⟨カウンタ名⟩}	（a, b, c, ..., z）
\Alph{⟨カウンタ名⟩}	（A, B, C, ..., Z）
\fnsymbol{⟨カウンタ名⟩}	（*, †, ‡, ...）

例えば節（\section）の見だし番号をローマ数字に変更するのであれば，節見だし用のカウンタ ‘section’ を次のように再定義します．

```
\renewcommand{\thesection}{\Roman{section}}
```

6.6　相互参照の工夫

例えば色について考察した章の中に同じような節見出し，表，図などが存在していたとしましょう．それらのラベルは重複してはいけませんので，何らかの工夫をしておいたほ

うが得策です．良く使われている方法に表 6.3 のように要素に応じてラベルに対して接頭語を付けます．簡単な例として節見出しを参照するときは

表 6.3　要素に応じたラベルの貼り方

要素	接頭語	対象
章見出し	chap:	\chapter
節見出し	sec:	\section
図	fig:	figure 環境中の\caption 命令
表	tab:	\table 環境中の\caption 命令
式	equ:	番号付きの数式（\equation 命令や eqnarray 環境）

```
\section{加法混色}\label{sec:addmix}
ほげは，ほげほげ．
\section{減法混色}\label{sec:submix}
\ref{sec:addmix}~節\pp{\pageref{sec:addmix}ページ}では，ほげほげ．
```

という入力になります．

これは表 6.3 の規則にしたがって何のマクロも作成せずに手動でやるとちょっと大変なことになります．

```
\section{加法混色}\label{sec:addmixcolor}
点$i$における色$c_i$は式~\ref{equ:addmixcolor}によって決まる．
\begin{equation}
 c_i  =  r_i + g_i + b_i\label{equ:addmixcolor}
\end{equation}
その関係は表~\ref{tab:addmixcolor}となる．
\begin{table}[htbp]
 % ここに表が入る．
 \caption{加法混色の表}\label{tab:addmixcolor}
\end{table}
またそれらを図式すると図~\ref{fig:addmixcolor}となる．
\begin{figure}[htbp]
 % ここに図が入る．
 \caption{加法混色の図}\label{fig:addmixcolor}
\end{figure}
\section{減法混色}\label{sec:submixcolor}
\ref{sec:addmixcolor}~節（\pageref{sec:addmixcolor}~ページ）ではほげ．
```

3.1 加法混色

点 i における色 c_i は式 3.1 によって決まる．

$$c_i = r_i + g_i + b_i \tag{3.1}$$

その関係は表 3.1 となる．

表 3.1 加法混色の表

またそれらを図式すると図 3.1 となる．

図 3.1 加法混色の図

3.2 減法混色

3.1 節 (5 ページ) ではほげ．

表 6.3 のような規則に従いマクロを作ります．マクロ側で自動的に接頭語を付けてくれれば人間の作業が減りますし，ミスも少なくなります．

```
\newcommand*{\chaplab}[1]{\label{chap:#1}}%      章のラベル
\newcommand*{\chapref}[1]{第~\ref{chap:#1}~章}% 章の参照
\newcommand*{\seclab}[1]{\label{sec:#1}}%        節のラベル
\newcommand*{\secref}[1]{\ref{sec:#1}~節}%       節の参照
\newcommand*{\figlab}[1]{\label{fig:#1}}%        図のラベル
\newcommand*{\figref}[1]{図~\ref{fig:#1}}%       図の参照
\newcommand*{\tablab}[1]{\label{tab:#1}}%        表のラベル
\newcommand*{\tabref}[1]{表~\ref{tab::#1}}%      表の参照
\newcommand*{\equlab}[1]{\label{equ:#1}}%        式のラベル
\newcommand*{\equref}[1]{式~\ref{equ::#1}}%      式の参照
```

このようなマクロを作成しておけば先程の入力は幾分簡略化できるでしょう．

```
\section{加法混色}\seclab{addmixcolor}
点$i$における色$c_i$は\eqref{addmixcolor}によって決まる．
\begin{equation}
 c_i  =  r_i + g_i + b_i\eqlab{addmixcolor}
\end{equation}
その関係は\tabref{addmixcolor}となる．
\begin{table}[htbp]
 % ここに表が入る．
 \caption{加法混色の表}\tablab{addmixcolor}
\end{table}
またそれらを図式すると\figref{addmixcolor}となる．
\begin{figure}[htbp]
 % ここに図が入る．
 \caption{加法混色の図}\figlab{addmixcolor}
\end{figure}
\section{減法混色}\seclab{submixcolor}
```

```
\secref{addmixcolor}(\pageref{sec:addmixcolor}~ページ) ではほげ.
```

さて，最後の 1 行を見てみると

```
\secref{addmixcolor}(\pageref{sec:addmixcolor}~ページ) ではほげ.
```

という記述が見受けられます．これは人間が手動で接頭語 sec:を付けなければならない例です．これもミスを誘い出す一因になるかもしれませんのでページ番号も参照するようなマクロを作ります．

```
\newcommand*{\fullchapref}[1]{第\ref{chap:#1}章 (\pageref{chap:#1}ページ)}
\newcommand*{\fullsecref}[1]{\ref{sec:#1}~節 (\pageref{sec:#1}ページ)}
\newcommand*{\fullfigref}[1]{図~\ref{fig:#1} (\pageref{fig:#1}ページ)}
\newcommand*{\fulltabref}[1]{表~\ref{tab::#1} (\pageref{tab:#1}ページ)}
\newcommand*{\fullequref}[1]{式~\ref{equ::#1} (\pageref{equ:#1}ページ)}
```

以上のようなマクロを作成しておけば入力が先程よりも簡単になるでしょう．

```
\section{減法混色}\seclab{submixcolor}
\fullsecref{addmixcolor}ではほげ.
```

LaTeX で相互参照を使う機会は 1 回以上あると思いますので（この冊子の例を自分で入力するなどで），この節で紹介したものをマクロパッケージ myref.sty としてまとめておくと便利かもしれません[*1]．

6.6.1　相互参照に関わる LaTeX の警告

コマンドプロンプトやシェルで表示される `LaTeX Warning:` の後に以下に示すような警告が表示されていると，相互参照に関する問題が解消されていないことを示します．

`Label ‘key’ multiply defined` というのは\label 命令で同じラベル名を持つラベルを定義しているということです．ラベルの重複がありますので，該当するラベルに別の名前を付けます．

`Reference ‘key’ on page n undefined` という警告が表示されたのならばラベル名が定義されていないことになります．

`Label(s) may have changed.  Return to get cross-ferecenses right.` が表示されたらラベルの値が変更されたということなので，もう 1 度タイプセットをします．この作業は 1 度で終わらないこともあるのでメッセージが表示されなくなるまでタイプセットを繰り返すこともあります．

ラベルに関する問題はラベルの参照する名前などのスペルミスなども考えられます．

[*1] http://texn.dante.jp/ に置くことにします．

第 7 章

数式の組版

LaTeX は TeX をベースにした組版システムなので数式の組版が得意です．この章では基本的な数式の出力の仕方を紹介します．

7.1 はじめに

LaTeX を使用する醍醐味は数式の組版あると言っても過言ではありません．LaTeX における数式の組み立てでは**グルーピング**が重要です．修飾される要素を明確に区別します．数式は普通の文章とは違い数式環境に記述します．数式は文章とは異なり，変数，数学記号，演算子，分数などの特殊な記述をしなければならないために，明示的に「ここが数式である」と宣言する必要があります．文章の部分を**テキストモード**，数式を含む部分を**数式モード**と呼びます．数式モードはどこから数式をはじめてどこまで数式にするかという始点と終点を決める必要もあります．数式モードでは以下の制約があります．

- 空白や改行は常に一つのスペースとして扱われます．通常は LaTeX 側が自動で空白を挿入しますがユーザーが明示的に空白を挿入することもできます．
- 空行は作成しません．一つの式に対して一つの段落を書くことができます．
- 半角英字はすべて指示がない限り数式イタリック体（*math italic*）になり，自動的に空白が調節されます．

7.2 数式の出力

数式は段落の中に挿入する**文中数式**と別行に挿入する**別行数式**の 2 種類があります．別行数式には番号付きで別行に挿入する equation 環境と複数行の番号付き数式を出力する eqnarray 環境などがあります．

7.2.1 文中数式

文中数式の出力には 3 通りあります．

```
$数式$
\(数式\)
\begin{math}数式\end{math}
```

どれも同じような動作をしますが，‘\(数式\)’ で囲むものが簡単ですのでこれだけ使えば良いでしょう．

math 環境などは記述量が増えるので使わなくても構いませんが，あまりに数式が長くなり見づらいときには math 環境で入れ子にするとすっきりするかも知れません．

```
$a$の 2 乗と$b$の 2 乗を足したものは$c$の 2 乗に
等しいということは\( a^2 + b^2 = c^2 \) と
表せるが{\LaTeX}では
\begin{math}  a^2 + b^2 = c^2  \end{math}
と書くこともできる．
```

a の 2 乗と b の 2 乗を足したものは c の 2 乗に等しいということは $a^2 + b^2 = c^2$ と表せるが LaTeX では $a^2 + b^2 = c^2$ と書くこともできる．

上記の例においてハット ‘^’ は添え字の上付きの機能を持っています．

7.2.2　グルーピング

変数 a の $x+y$ 乗を出力するために LaTeX では一塊の要素を**波括弧**で**グルーピング**します．ここではべき乗を例にとって見てみましょう．

```
\( a^x+y \neq a^{x+y} \)
```

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

グルーピングによって数式の要素を一つのグループにします．数式環境に限りませんが LaTeX では一つにしたい要素をグループとして扱い，波括弧でグループ化を行います．

7.2.3　別行数式

数式を別行に立てる方法は LaTeX では主に 3 通りあります．

```
$$数式$$
\[数式\]
\begin{displaymath} 数式\end{displaymath}
```

これら三つの命令の前後で自動的に改行が入り新しい行から数式が出力されます．両方とも数式を中央揃えで表示します．数式を左揃えにしたければ文書クラスファイルのオプションに *fleqn* を指定します．上記の文中数式と同じで\[数式\] だけを使ったほうが簡単です．displaymath 環境は記述量が増えるので使わなくても構いません．あまりに数式が長くなったときなどには使えるでしょう．

```
別行立て数式は \[ c^2 = a^2 + b^2 \]
のように自動的に中央揃えになります．
```

別行立て数式は

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

のように自動的に中央揃えになります．

```
別行立て数式は
\begin{displaymath}
 a^2 + b^2 = c^2
\end{displaymath}
と書くこともできます．
```

別行立て数式は

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

と書くこともできます．

7.2.4 番号付き数式

文書の中で参照するだろうと思われる数式には番号を付けます．そのような数式を**番号付き数式**と呼び，数式が 1 行の場合は equation 環境で出力することができます．

```
\begin{equation}
数式\label{⟨ラベル⟩}
\end{equation}
```

equation で囲むことにより 1 行の番号付きの数式を出力することが出来ます．番号付きの数式は基本的にラベルを貼ることが出来ます．ラベルの参照の仕方は 6.5 節を参照してください．

```
\begin{equation}
 a^2 + b^2 = c^2  \label{eq:equ}
\end{equation}
式~(\ref{eq:equ}) より$c^2$は$a^2+b^2$に等しい．
```

$$a^2 + b^2 = c^2 \tag{7.1}$$

式 (7.1) より c^2 は $a^2 + b^2$ に等しい．

7.2.5 複数行数式

```
\begin{eqnarray*}
左辺 & (=) & 右辺\\
左辺 & (=) & 右辺
\end{eqnarray*}
```

流れのある複数行の数式や証明などでイコール ‘=’ の位置を揃えるときは eqnarray*環境を使用し，これを**複数行数式**と呼びます．この環境は任意の行数で 3 列の行列に似ています．必ず 1 行にはアンド ‘&’ が二つ，行の終わりには改行 ‘\\’ を書きます．ただし最終行には改行を入れません．また各列における成分は省略することが可能です．

```
\begin{eqnarray*}
f(x)      & = & x^2  \\
f'(x)     & = & 2x
\end{eqnarray*}
```

$$f(x) = x^2$$
$$f'(x) = 2x$$

7.2.6　複数行番号付き数式

後から参照するだろう複数行の数式には番号付けを行います．これを**複数行番号付き数式**と呼び，eqnarray 環境を使って記述します．書式は eqnarray*と同じです．ラベルは 1 行ごとに改行 '\\' の前に貼ることが出来ます．また番号を出力したくない行は\nonumber 命令によって番号を振らないこともできます．

```
\begin{eqnarray}
f(x)       &=& x^2 \label{eq1}\\
f'(x)      &=& 2x  \label{eq2}\\
\int f(x)dx&=& x^3/3+C\nonumber
\end{eqnarray}
式 (\ref{eq1}) を微分したものが式 (\ref{eq2}) である．
```

$$f(x) = x^2 \tag{7.2}$$
$$f'(x) = 2x \tag{7.3}$$
$$\int f(x)dx = x^3/3 + C$$

式 (7.2) を微分したものが式 (7.3) である．

複数行数式はすでに数式モードになっていますのでそれをさらに数式環境で囲むなどの処理をしないでください．くれぐれも**最終行に改行を入れない**でください．

7.3　書体の変更

数式では書体の変更が必要になると思います．例えば行列を表すものはボールド体に変更し数式中で文字を表示するときがあるでしょう．そのようなときは書体変更用のコマンドを使います．数式中では通常のテキストモードで使う書体変更コマンドは使えませんので，数式の書体変更用のコマンドを使います．数式中でしか使用できない書体用コマンドは表 7.1 の通りです．

表 7.1　数式モードにおける書体の変更

書体	命令	出力
標準の書体	\mathnormal	*ABCabc*
ローマン体	\mathrm	ABCabc
サンセリフ体	\mathsf	ABCabc
タイプライタ体	\mathtt	ABCabc
ボールド体	\mathbf	**ABCabc**
イタリック体	\mathit	*ABCabc*
カリグラフィック体	\mathcal	$\mathcal{ABC}$

```
\begin{displaymath}
\int f(x)dx \neq \int f(x)\mathrm{d}x
\end{displaymath}
```

$$\int f(x)dx \neq \int f(x)\mathrm{d}x$$

行列を表現するのにブラックボードボールド体（黒板風書体）を使うことがあるそうです．これは文字が白抜きになりボールド体よりも行列であることが分かりやすくなってい

ます．これを使うには amssymb を読み込みます．数式中で通常のテキストを使いたいときは amsmath パッケージを読み込み\text 命令を使います．命令は表 7.2 となります．

表 7.2　amssymb による数式書体の拡張

書体	命令	出力
フラクトゥール体	\mathfrak	$\mathfrak{ABCabc}$
ブラックボードボールド体	\mathbb	$\mathbb{ABC}$
数式内テキスト	\text	ABC ほげほげ

```
\[ x\in \mathbf{R} \neq x\in \mathbb{R} \]
```

$$x \in \mathbf{R} \neq x \in \mathbb{R}$$

7.4　数式における空白の調節

数式モードでは入力した半角空白が反映されません．LaTeX は数式モードでは自動的に隣り合うから挿入すべき空白を決めています．ですがユーザが空白を調節したほうが綺麗に見える場合があります．ユーザー側で空白を調節するため表 7.3 のコマンドを使います．積分 '$\int$' や全微分 'dx' のあいだにはユーザーが空白を入れると見栄えがします．

表 7.3　数式における空白の制御

空白の大きさ	命令	入力例	出力例
空白なし	␣	dx␣dy	$dxdy$
かなり小さい空白	\,	dx\, dy	$dx\,dy$
小さい空白	\:	dx\: dy	$dx\:dy$
少し小さい空白	\;	dx\; dy	$dx\;dy$
半角の空白	\␣	dx\␣dy	$dx\ dy$
全角の空白	\quad	dx\quad dy	$dx\quad dy$
全角の 2 倍の空白	\qquad	dx\qquad dy	$dx\qquad dy$
負の小さい空白	\!	dx\!dy	$dx!dy$

```
\[ \int\int f(x)dxdy \neq
   \int\!\!\!\int f(x)\ dx\ dy  \]
```

$$\int\int f(x)dxdy \neq \iint f(x)\ dx\ dy$$

7.5　基本的な数式コマンド

数式を書く環境を理解したら実際にそこに記述する記号などを覚えることになります．

7.5.1　添え字

LaTeX での添え字の入力は簡単です．

値^{**上付き**}
値_{**下付き**}

添え字には**上付き**と**下付き**の 2 種類があります．これらの添え字を使うにはグルーピングの必要があります．1 文字だけの添え字のときに丸括弧は必要ありませんが，添え字にしたいものが複数のときはグルーピングの処理が必要です．表 7.4 で例を示しますので参考にしてください．添え字をつけるときに上付きと下付きの順番は関係ありません．添え字は何もないものに対しても添えることが可能です．表 7.4 でもその方法がとられています．

表 7.4　添え字の使い方の例

意味	命令	出力	意味	命令	出力
右上	`x^{a+b}`	x^{a+b}	左上	`{}^{a+b}x`	${}^{a+b}x$
右下	`x_{a+b}`	x_{a+b}	左下	`{}_{a+b}x`	${}_{a+b}x$
右上と右下	`x^{a+b}_{c+d}`	x_{c+d}^{a+b}	左上と左下	`{}^{a}_{b}x`	${}_{b}^{a}x$
右上の右上	`x^{a^{b}}`	$x^{a^{b}}$	左下と右下	`{}_{a}x_{b}`	${}_{a}x_{b}$

```
\( {}^{a+b}_{x+y}A^{a+b}_{x+y} \)
```
$${}^{a+b}_{x+y}A^{a+b}_{x+y}$$

ハット ‘^’ やアンダーバー ‘_’ は別の命令としても用意されています．上付きの\sp と下付きの\sb 命令を使うと良いでしょう．

```
\( A^4_3 \neq A\sp4\sb3 \)
```
$$A^4_3 \neq A^4_3$$

以上のような方法では左側に添え字を付けるときにうまくいかない場合がありますので，Harald harders 氏による leftidx パッケージを使います．

\leftidx{⟨左側添え字⟩}{⟨数式⟩}{⟨右側添え字⟩}
\ltrans{⟨数式⟩}

置換行列の上付き添え字は若干空白を抑えるために\ltrans 命令を使います．

```
\begin{eqnarray*}
{}_a^b\left(\frac{x}{y}\right)_c^d &\neq&
   \leftidx{_a^b}{\left(\frac{x}{y}
   \right)}{_c^d}\\
{}^\mathrm{t} A &\neq& \ltrans{A}
\end{eqnarray*}
```
$$\begin{eqnarray*}
{}_a^b\left(\frac{x}{y}\right)_c^d &\neq&
   \leftidx{_a^b}{\left(\frac{x}{y}
   \right)}{_c^d}\\
{}^\mathrm{t} A &\neq& \ltrans{A}
\end{eqnarray*}$$

7.5.2　数学関数

数式モードでは自動的に英字がイタリック体になります．これは変数を表すためです．‘*d*’ と ‘d’ では数式では違う意味を持ちます．数学関数や極限などは**ローマン体**，まっすぐな書体で書くのが慣わしです．LaTeX ではあらかじめそのような関数が定義されており，すぐに使える命令は表 7.5 の通りです．

表 7.5　主な数学関数

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

```
\[ cos^2x+sin^2x \neq \cos^2x+\sin^2x \]
```

$$cos^2x + sin^2x \neq \cos^2 x + \sin^2 x$$

また\bmod のように**法**を表すための命令もあります．

\bmod{⟨文字列⟩} (2 項演算子として)
\pmod{⟨文字列⟩}

```
\( \mathrm M\bmod{\mathrm N} \neq
   \mathrm M\pmod{\mathrm N} \)
```

$$\mathrm M\bmod{\mathrm N} \neq \mathrm M\pmod{\mathrm N}$$

7.5.3　大きさ可変の数学記号

数式中では修飾するものによって大きさの変わる記号があります．積分記号などがそれにあたります．主な大きさが可変な記号は表 7.6 の通りです．

```
\begin{displaymath}
\int^b_a f(x)dx \neq \sqrt{\frac{1}{f(x)}}
\end{displaymath}
```

$$\int^b_a f(x)dx \neq \sqrt{\frac{1}{f(x)}}$$

```
\begin{displaymath}
\sqrt{\frac{1}{g(x)}+\sqrt{\int f(x)dx}}
\end{displaymath}
```

$$\sqrt{\frac{1}{g(x)}+\sqrt{\int f(x)dx}}$$

```
\begin{displaymath}
\frac{1}{g(x)} + \frac{1}{2x^3+5x^2+8x+5}
\end{displaymath}
```

$$\frac{1}{g(x)} + \frac{1}{2x^3+5x^2+8x+5}$$

表 7.6　大きさ可変の数学記号

種類	命令	出力例
分数	\frac{⟨分子⟩}{⟨分母⟩}	$\frac{分子}{分母}$
根号	\sqrt{⟨値⟩}	$\sqrt{値}$
添え字付き根号	\sqrt[⟨根⟩]{⟨値⟩}	$\sqrt[根]{値}$
添え字付き積分	\int^{⟨上付き⟩}_{⟨下付き⟩}	$\int_{下付き}^{上付き}$
添え字付き総和	\sum^{⟨上付き⟩}_{⟨下付き⟩}	$\sum_{下付き}^{上付き}$

\sum や\int などの添え字は上下に付く場合と右上と右下に付く場合があります．これを変更するには\limits と\nolimits を使います．

```
\limits
\nolimits
```

\limits を添え字を行うコマンドの前に置くと添え字をされる記号の上下に添え字を表示します．\nolimits はその反対のことをします．

```
\begin{eqnarray*}
\sum\nolimits^n_{k=0}k &\neq&
   \sum^n_{k=0}k\\
\int^b_a dx &\neq&\int\limits^b_a dx
\end{eqnarray*}
```

$$\sum\nolimits_{k=0}^{n} k \neq \sum_{k=0}^{n} k$$
$$\int_a^b dx \neq \int\limits_a^b dx$$

```
\begin{eqnarray*}
\lim\nolimits_{n\rightarrow0}n &\neq&
  \lim_{n\rightarrow0} n \\
\prod^n_{i=1}n &\neq&
   \prod\nolimits^n_{i=1}n
\end{eqnarray*}
```

$$\lim\nolimits_{n\to 0} n \neq \lim_{n\to 0} n$$
$$\prod_{i=1}^{n} n \neq \prod\nolimits_{i=1}^{n} n$$

7.5.4　区切り記号と括弧

LaTeX における**区切り記号** (括弧を含む) は何も指定しなければ勝手には大きさが変わりません．区切り記号は

- \left と\right 命令を使って大きさを変える．
- 区切り記号の大きさを指定する．

という二つの方法によって大きさを変更することもできます．

```
\begin{displaymath}
\left[ \Big(x+y\Big) \right]
\end{displaymath}
```

$$\left[\left(x+y\right)\right]$$

括弧で括られたり，区切られる要素に応じて大きさが変更できる区切り記号は表 7.7 となります．括弧などは要素を区切るための記号で，要素をきちんと括るべきです．LaTeX

表 7.7　主な区切り記号

(	(	⌋	\rfloor	↕	\updownarrow	{	\lbrace
)	)	⌊	\lfloor	⇑	\Uparrow	⌉	\rceil
[	[	|	\arrowvert	⇓	\Downarrow	⌈	\lceil
]	]	‖	\Arrowvert	⇕	\Updownarrow	∫	\lmoustache *
{	\{	‖	\Vert	\	\backslash	⎭	\rmoustache *
}	\}	|	\vert	⟩	\rangle	⎛	\lgroup *
|	|	↑	\uparrow	⟨	\langle	⎞	\rgroup *
‖	\\|	↓	\downarrow	}	\rbrace	⎪	\bracevert *

* 大型の区切り記号です．

においては大きさが可変な区切り記号を用いてそれらを書き表します．‘\left’ 命令と ‘\right’ 命令を対で使うと括られた要素が適切な大きさの括弧で区切られます．\left と\right には表 7.7 から記号を選ぶことによって，左右の区切りの対を自由に組み合わせられます．可変の括弧は修飾する式によって自動的に大きさを変更されるのでとても便利です．

```
\begin{displaymath}
\left( \frac{1}{1+\frac{1}{1+x}} \right)
\end{displaymath}
```

$$\left(\frac{1}{1+\frac{1}{1+x}}\right)$$

```
\[ \left\lmoustache \left\{
  \left(\frac{1}{x}+1\right)
  +\left(\frac{1}{x^2}+2\right)
\right\} \right\rmoustache \]
```

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

```
\begin{displaymath}
 \left\uparrow \int f(x)dx \right\downarrow
 +\left\lgroup \int g(x)dx \right\rgroup
\end{displaymath}
```

$$\left\uparrow \int f(x)dx \right\downarrow + \left(\int g(x)dx \right)$$

自分で括弧の大きさを指定することもできます．大きさを指定した場合はそれ以上括弧の大きさが変わりませんので注意が必要です（表 7.8）．

表 7.8　括弧の大きさを指定する例

/	/	(	(	)	)	|	|	‖	|
/	\big/	(	\bigl(	)	\bigr)	|	\bigm|	‖	\bigm|
/	\Big/	(	\Bigl(	)	\Bigr)	|	\Bigm|	‖	\Bigm|
/	\bigg/	(	\biggl(	)	\biggr)	|	\biggm|	‖	\biggm|
/	\Bigg/	(	\Biggl(	)	\Biggr)	|	\Biggm|	‖	\Biggm|

```
\begin{displaymath}
\Biggl\| \Biggl( \int f(x) dx \Biggr)
    \Biggm/ \Biggl( \int g(x) dx \Biggr)
\Biggr\|
\end{displaymath}
```

$$\left\|\left(\int f(x)dx\right)\middle/\left(\int g(x)dx\right)\right\|$$

表 7.8 を見ると分かると思いますが，括弧，いわゆる区切り記号に対して\big や\Big を付けるとその区切り記号を特定の倍率で拡大するという機能があります．左側を区切るには\bigl 類を，関係子としての区切り記号は\bigm 類を，右側を区切る記号には\bigr 類を，特に指定しないならば\big 類を使うようにします．上記の\big 類を使った例と\left と\right による例を見比べてください．

```
\[ \left\|
   \left( \int f(x) dx \right)
   \Biggm/ \left( \int g(x) dx \right)
   \right\| \]
```

$$\left\|\left(\int f(x)dx\right)\middle/\left(\int g(x)dx\right)\right\|$$

片方だけに区切り記号があれば良いときはピリオド ‘.’ でいずれかの記号を省略できます．

```
\[ \left( \left\uparrow
   \int f(x)dx + \int g(x)dx
   \right. \right) \]
```

$$\left(\left\uparrow \int f(x)dx + \int g(x)dx\right.\right)$$

7.5.5　行列

LaTeX における行列は array 環境中に記述します．array 環境はそのままでは数式にはならず math 環境や\[\] の中に入れたり$$の中に入れてあげます．array 環境の基本的な使い方は

```
\begin{array}{列指定子}
 a_{11}  &  ...  &  a_{1n}  \\
  ⋮      &  ⋱    &  ⋮      \\
 a_{m1}  &  ...  &  a_{mn}
\end{array}
```

というように m 行 n 列の行列を書きます．ここでアンド ‘&’ は成分（要素）の区切りを意味し，‘\\’ は行の終わりを意味しています．括弧は必要ならば前述の区切り記号で括ることもできます．表と行列は基本的に同じ構造で，縦の罫線も横の罫線も入れることが出来ます．

```
\begin{array}{列数と縦罫線の指定}
```

の部分では 4 列あるならば

```
\begin{array}{lc|cr}
```

のようにします．このときの ‘l’，‘c’，‘r’ は行列の中の要素の配置場所を指定するものです．真ん中にはテキストバー ‘|’ があります，これは縦方向の罫線を表しています．このような記号を**列指定子**と呼びます．array 環境中で指定できる列指定子は表 7.9 となります．array 環境は入れ子にすることも出来ます．行列の中に行列を書いたりすることも

表 7.9　array 環境の主な列指定子

列指定子	意味
l	行列の縦 1 列を左揃えにする
c	行列の縦 1 列を中央揃えにする
r	行列の縦 1 列を右揃えにする
|	縦の罫線を引く
||	縦の 2 重罫線を引く
@{**表現**}	表現を縦 1 列追加します
p{**長さ**}	ある列の幅の長さを直接指定します
*{**回数**}{**項目**}	回数分だけ項目を繰り返す.

出来ますので便利です.

```
\[ \left( \begin{array}{cc}
      a & b \\ c & d
   \end{array} \right) \]
```

$$\left(\begin{array}{cc} a & b \\ c & d \end{array} \right)$$

横方向に行列が続く場合があるため array 環境の**最後の行に改行は入れません**.

```
\[ \left( \begin{array}{*{2}{c}}
     a & b \\ c  & d
    \end{array} \right)
  \left( \begin{array}{c}
       m \\ n
    \end{array} \right)  =
  \left( \begin{array}{c}
      am+bn \\ cm+dn \\
    \end{array} \right) \]
```

$$\left(\begin{array}{cc} a & b \\ c & d \end{array} \right) \left(\begin{array}{c} m \\ n \end{array} \right) = \left(\begin{array}{c} am+bn \\ cm+dn \end{array} \right)$$

array 環境には次に示すような場合分けを行う使い方もあります.

```
\[ f(x)= \left\{
\begin{array}{cl}
  x & (x > 0)\\
  0 & (x = 0)\\
 -x & (x < 0) \end{array}
\right. \]
```

$$f(x) = \left\{ \begin{array}{cl} x & (x > 0) \\ 0 & (x = 0) \\ -x & (x < 0) \end{array} \right.$$

水平に罫線などを入れたりするときには\hline，要素の中で縦の罫線を引くときには\vline などを使います（表 7.10）．罫線などの使い方は以下の例を見て理解してください.

表 7.10　array 環境中での罫線の命令

命令	意味
\hline	横に引けるだけの罫線を引きます
\hline\hline	引けるだけの 2 重の横罫線を引きます
\vline	要素の中で引けるだけの縦罫線を引きます
\cline{⟨範囲⟩}	要素の罫線を行の範囲を指定して引きます
\multicolumn{⟨数値⟩}{⟨列指定子⟩}{⟨要素⟩}	行をつなげて列指定子通りに要素を出力します

```
\begin{displaymath}
\begin{array}{llc} \hline
\multicolumn{3}{c}{f(x)} \\ \hline
g(x) & h(x) & i(x) \\ \cline{2-2}
j(x)+k(x)+l(x)+m(x)+n(x) & o(x) & p(x)\\
\end{array}
\end{displaymath}
```

$$\begin{array}{llc} \hline \multicolumn{3}{c}{f(x)} \\ \hline g(x) & h(x) & i(x) \\ \cline{2-2} j(x)+k(x)+l(x)+m(x)+n(x) & o(x) & p(x) \\ \end{array}$$

```
\begin{displaymath}
\begin{array}{ll|ll}
a & b & c & d \\ \hline\hline
e & g & h & i \\\cline{3-4}
j & \multicolumn{1}{|l}k & l & m \\\hline
n & o & \multicolumn{1}{l|}p & q \\
r & s & t & \multicolumn{1}{|l|}u \\
v & w & x & y \\ \hline\hline
\end{array}
\end{displaymath}
```

$$\begin{array}{ll|ll} a & b & c & d \\ \hline\hline e & g & h & i \\ \cline{3-4} j & \multicolumn{1}{|l}{k} & l & m \\ \hline n & o & \multicolumn{1}{l|}{p} & q \\ r & s & t & \multicolumn{1}{|l|}{u} \\ v & w & x & y \\ \hline\hline \end{array}$$

悪ふざけが過ぎました，なんか迷路みたいですね．

array 環境の簡易版として行列作成用の\matrix と丸括弧を付ける\pmatrix と\matrix にラベルも付けられる\bordermatrix などの命令があります．ただし\matrix 命令と\pmatrix に関しては amsmath パッケージの matrix 環境や pmatrix 環境を使った方が良いでしょう．

```
\[ \begin{pmatrix} x \\ y \end{pmatrix}
   \begin{pmatrix}
      a & b & c \end{pmatrix} \]
```

$$\begin{pmatrix} x \\ y \end{pmatrix} \begin{pmatrix} a & b & c \end{pmatrix}$$

\bordermatrix 環境の括弧では各成分を区切るにはアンド '\&' を使い，行の終わりには '\cr' 命令を使います．

```
\[ A=\ \bordermatrix{
     & 1 & 2 \cr
  1  & a & b \cr
  2  & c & d }     \]
```

$$A = \bordermatrix{ & 1 & 2 \cr 1 & a & b \cr 2 & c & d }$$

7.6 表示形式の調整

数式を記述する各環境において自動的に各要素の大きさが決められます．文中数式での分数は $\frac{a}{b}$ という出力になりますが，これでは少し小さいので $\displaystyle\frac{a}{b}$ としたいときがあると思います．そのようなときはユーザーが表示形式を変更するには表 7.11 の命令が使えます．あまり多用すると段落のあいだが空きすぎて逆に見栄えが悪くなるのである程度長い数式

表 7.11 数式の表示形式の変更

命令	出力形式	例（$\frac{a}{b}$）
\displaystyle	別行立て形式	$\displaystyle\frac{a}{b}$
\textstyle	文中数式形式	$\frac{a}{b}$
\scriptstyle	添え字形式	$\scriptstyle\frac{a}{b}$
\scriptscriptstyle	添え字の中の添え字形式	$\scriptscriptstyle\frac{a}{b}$

を文中に入れているときは別行立てにするのが良い方法です．また文中の数式に限りませ

んが，分数は $\frac{a}{b}$ と書くよりも a/b とするほうが一般的で見やすいのでスラッシュによる表記にしたほうが良いでしょう．

```
\(f(x)\) の不定積分 \(\int f(x)dx\) と
\(\displaystyle \int f(x)dx\) は{\LaTeX}
では少し違うし分数は$\frac{a}{b}$と書く
よりも$a/b$と書くほうが一般的である．
```

$f(x)$ の不定積分 $\int f(x)dx$ と $\displaystyle\int f(x)dx$ は LaTeX では少し違うし分数は $\frac{a}{b}$ と書くよりも a/b と書くほうが一般的である．

```
\[ \frac{1}{1+\frac{1}{1+\frac{1}{1+x}}}
   \neq \frac{1}{\displaystyle 1+
   \frac{1}{\displaystyle 1+
   \frac{1}{1+x}}} \]
```

$$\frac{1}{1+\frac{1}{1+\frac{1}{1+x}}} \neq \frac{1}{\displaystyle 1+\frac{1}{\displaystyle 1+\frac{1}{1+x}}}$$

```
\( \int^b_a f(x)dx \neq
   {\displaystyle\int^b_a g(x)dx}
\)
```

$\int^b_a f(x)dx \neq {\displaystyle\int^b_a g(x)dx}$

7.7　数式モード中の記号

記号の中には数式モード中でしか使えないものがほとんどです．以下の記号は\(\) で囲むなど，数式環境の中で使用しないと `! Missing $ inserted.` のようなエラーが表示されます．

7.7.1　ギリシャ文字

数式中の変数ならびに定数にはギリシャ文字を使うのが一般的です．ギリシャ小文字は表 7.12，小文字の変体文字は表 7.13，大文字は表 7.14 となります．ギリシャ

表 7.12　ギリシャ小文字

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

小文字においてオミクロン ‘o’ だけはアルファベットの ‘o’ と同じため特別に記号が用意されていません．逆に ‘\o’ は文中で使うべき記号であり，この命令を数式中で使うと `LaTeX Warning: Command \o invalid in math mode on input line 30.` のように警告が表示されます．

```
\begin{eqnarray*}
\cos^2\theta+\sin^2\theta &\neq&
   \cos^2x + \sin^2x
\end{eqnarray*}
```

$$\cos^2\theta + \sin^2\theta \neq \cos^2 x + \sin^2 x$$

表 7.13 ギリシャ小文字の変体文字

ε	\varepsilon	ϑ	\vartheta	ϖ	\varpi
ϱ	\varrho	ς	\varsigma	φ	\varphi

一筆書きできる小文字が使われた名残でしょうか.

表 7.14 ギリシャ大文字

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

ギリシャ大文字でもアルファベットと同じ文字は特別な記号が用意されておりません. ギリシャ小文字と同じようにオミクロン '\O' を数式中で使うと次のような警告が表示されます.

```
LaTeX Warning:  Command \O invalid in math mode on input line 40.
```

さらにギリシャ大文字の A, B, E, Z, H, I, K, M, N, O, P, T, X はそのままではイタリック体となって変数を意味してしまいますので定数としてのギリシャ大文字を出力するためには\mathrm を使います.

```
\begin{eqnarray*}
       A      &\neq& \mathrm A \\
  F(x)+C      &\neq& F(x)+ \mathrm C \\
  \mathit{foo}&\neq& \mathrm{foo}
\end{eqnarray*}
```

$$A \neq \mathrm{A}$$
$$F(x) + C \neq F(x) + \mathrm{C}$$
$$\mathit{foo} \neq \mathrm{foo}$$

7.7.2　関係子や演算子などの数学記号

表 7.15　関係子

以下のコマンドの前に\not コマンドを付ければその関係子の否定になります

$\leq$	\le	$\in$	\in	$\sqsupseteq$	\sqsupseteq	$\neq$	\neq
$\prec$	\prec	$\notin$	\notin	$\dashv$	\dashv	$\doteq$	\doteq
$\preceq$	\preceq	$\geq$	\ge	$\ni$	\ni	$\propto$	\propto
$\ll$	\ll	$\succ$	\succ	$\equiv$	\equiv	$\models$	\models
$\subset$	\subset	$\succeq$	\succeq	$\sim$	\sim	$\perp$	\perp
$\subseteq$	\subseteq	$\gg$	\gg	$\simeq$	\simeq	$\mid$	\mid
$\sqsubseteq$	\sqsubseteq	$\supset$	\supset	$\asymp$	\asymp	$\parallel$	\parallel
$\vdash$	\vdash	$\supseteq$	\supseteq	$\approx$	\approx	$\bowtie$	\bowtie
$\smile$	\smile	$\frown$	\frown	$\cong$	\cong		

表 7.16　2 項演算子

$\pm$	\pm	$\cdot$	\cdot	$\setminus$	\setminus	$\ominus$	\ominus
$\mp$	\mp	$\cap$	\cap	$\wr$	\wr	$\otimes$	\otimes
$\times$	\times	$\cup$	\cup	$\diamond$	\diamond	$\oslash$	\oslash
$\div$	\div	$\uplus$	\uplus	$\triangle$	\bigtriangleup	$\odot$	\odot
$*$	\ast	$\sqcap$	\sqcap	$\triangledown$	\bigtriangledown	$\bigcirc$	\bigcirc
$\star$	\star	$\sqcup$	\sqcup	$\triangleleft$	\triangleleft	$\dagger$	\dagger
$\circ$	\circ	$\vee$	\vee	$\triangleright$	\triangleright	$\ddagger$	\ddagger
$\bullet$	\bullet	$\wedge$	\wedge	$\oplus$	\oplus	$\amalg$	\amalg

表 7.17　大型演算子

これらは大きさが可変です

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

```
\( \vec{a}+\vec{b}\neq \vec{a+b}
   \neq \overrightarrow{a+b} \)
```

$$\vec{a}+\vec{b}\neq \vec{a+b}\neq \overrightarrow{a+b}$$

表 7.18　小さいアクセント

これらの小さいアクセントは大きさが変わりません

$\hat{a}$	\hat{a}	$\check{a}$	\check{a}	$\breve{a}$	\breve{a}	$\acute{a}$	\acute{a}
$\grave{a}$	\grave{a}	$\tilde{a}$	\tilde{a}	$\bar{a}$	\bar{a}	$\dot{a}$	\dot{a}
$\ddot{a}$	\ddot{a}	$\vec{d}$	\vec{a}				

表 7.19　大きいアクセント

大きいアクセントは大きさが可変です

$\overline{m+M}$	\overline	$\overbrace{m+M}$	\overbrace
$\underline{m+M}$	\underline	$\underbrace{m+M}$	\underbrace
$\overleftarrow{m+M}$	\overleftarrow	$\widehat{m+M}$	\widehat
$\overrightarrow{m+M}$	\overrightarrow	$\widetilde{m+M}$	\widetilde

```
\begin{displaymath}
 \overbrace{a+b+c+d+e+f+g}^{h+i+j+k}+
 \underbrace{l+m+n}_{o+p+q}
\end{displaymath}
```

$$\overbrace{a+b+c+d+e+f+g}^{h+i+j+k}+\underbrace{l+m+n}_{o+p+q}$$

表 7.20　矢印

$\leftarrow$	\leftarrow	$\longrightarrow$	\longrightarrow	$\leftrightarrow$	\leftrightarrow
$\Leftarrow$	\Leftarrow	$\Longrightarrow$	\Longrightarrow	$\Leftrightarrow$	\Leftrightarrow
$\hookleftarrow$	\hookleftarrow	$\longmapsto$	\longmapsto	$\rightleftharpoons$	\rightleftharpoons
$\leftharpoonup$	\leftharpoonup	$\hookrightarrow$	\hookrightarrow	$\Longleftrightarrow$	\Longleftrightarrow
$\leftharpoondown$	\leftharpoondown	$\rightharpoonup$	\rightharpoonup	$\updownarrow$	\updownarrow
$\longleftarrow$	\longleftarrow	$\rightharpoondown$	\rightharpoondown	$\Updownarrow$	\Updownarrow
$\Longleftarrow$	\Longleftarrow	$\uparrow$	\uparrow	$\nearrow$	\nearrow
$\rightarrow$	\rightarrow	$\Uparrow$	\Uparrow	$\swarrow$	\swarrow
$\Rightarrow$	\Rightarrow	$\downarrow$	\downarrow	$\searrow$	\searrow
$\mapsto$	\mapsto	$\Downarrow$	\Downarrow	$\nwarrow$	\nwarrow

```
\begin{displaymath}
  (p\rightarrow r)\vee
  (q\rightarrow s)
\end{displaymath}
```

$$(p \rightarrow r) \vee (q \rightarrow s)$$

```
\[ \forall{x}\forall{y}(
     P(x,y)\vee(f(x)\wedge g(x))) \]
```

$$\forall x \forall y (P(x,y) \vee (f(x) \wedge g(x)))$$

表 7.21　特殊な数学記号

$\aleph$	\aleph	∂	\partial	$\bot$	\bot	$\natural$	\natural
$\hbar$	\hbar	∞	\infty	$\angle$	\angle	$\sharp$	\sharp
$\imath$	\imath	$\prime$	\prime	$\triangle$	\triangle	$\clubsuit$	\clubsuit
$\jmath$	\jmath	$\emptyset$	\emptyset	$\forall$	\forall	$\diamondsuit$	\diamondsuit
ℓ	\ell	∇	\nabla	$\exists$	\exists	$\heartsuit$	\heartsuit
$\wp$	\wp	$\surd$	\surd	$\neg$	\neg	$\spadesuit$	\spadesuit
$\Re$	\Re	|	|	\	\backslash		
$\Im$	\Im	$\top$	\top	$\flat$	\flat		

```
\( e^{j\theta}=\Re{\{e^{j\theta}\}}
   +\Im{\{e^{j\theta}\}}
   =\cos\theta+j\sin\theta\)
```

$$e^{j\theta} = \Re\{e^{j\theta}\} + \Im\{e^{j\theta}\} = \cos\theta + j\sin\theta$$

表 7.22　点

...	\ldots	$\cdots$	\cdots	$\vdots$	\vdots	$\ddots$	\ddots

```
\[ (a_0+a_1+\cdots+a_n)
   \neq \{a_0,a_1,\ldots,a_n\} \]
```

$$(a_0 + a_1 + \cdots + a_n) \neq \{a_0, a_1, \ldots, a_n\}$$

7.7.3　標準ではない数学記号

LaTeX 2ε からはこぼれた記号類を出力するためには，Frank Mittelbach 氏が作成した latexsym を読み込むと良いでしょう．すでに amssymb か amsfonts を読み込んでいるならば，そちらに定義されているので latexsym をさらに読み込まなくても良いです．

表 7.23　標準ではない数学記号

$\mho$	\mho	$\bowtie$	\Join	$\Box$	\Box	$\Diamond$	\Diamond
$\leadsto$	\leadsto	$\sqsubset$	\sqsubset	$\sqsupset$	\sqsupset	$\lhd$	\lhd
$\unlhd$	\unlhd	$\rhd$	\rhd	$\unrhd$	\unrhd		

7.8　定義や定理など

\theorem 命令を使うと新規に定義型や定理型の環境を作成できます．

```
\newtheorem{⟨名前⟩}{⟨ラベル⟩}[⟨親カウンタ⟩]
\newtheorem{⟨名前⟩}[⟨定義済みの環境⟩]{⟨ラベル⟩}
```

章や節などを通し番号の前に付けるにはその ⟨親カウンタ⟩ を表 6.2 から選びます．別々の環境で同じ通し番号を使いたい場合は ⟨定義済みの環境⟩ を指定します．具体的な使用例として

```
\newtheorem{Prob}{問題}[chapter]
\newtheorem{Exe}[Prob]{例題}
```

をプリアンブルに記述しておけば以下のように使えます．

```
\begin{Exe}\label{Hoge:ware}
この冊子は難しいか．答えは簡単だ．
\end{Exe}
\begin{Prob}\label{Geho:yueni}
この冊子は hoge かどうか考えよ．
\end{Prob}
例題~\ref{Hoge:ware}より
問題~\ref{Geho:yueni}が導かれる．
```

▷ **例題 7.1** この冊子は難しいか．答えは簡単だ．

▶ **問題 7.2** この冊子は hoge かどうか考えよ．

例題 7.1 より問題 7.2 が導かれる．

実際の出力は異なると思います．\theorem 命令は定理型や定義型の環境を作成するために作られたので日本語用には思うようにカスタマイズできないようです．

7.8.1 定理型環境のカスタマイズ

Frank Mittelbach 氏が作成した theorem は LaTeX における\theorem 命令を拡張したパッケージです．このパッケージは例えば「定理型」や「定義型」だけでなく，「問題型」や「例題型」などの環境を作成するときに満足の行く出力になると思われます．$\mathcal{AMS}$-LaTeX に含まれる amsthm というパッケージもありますが Frank Mittelbach 氏が作成した theorem を使ったほうが便利だと思います．定理型の環境を新設するときは LaTeX の \theorem 命令と同じように

```
\newtheorem{⟨環境名⟩}{⟨名前⟩}
```

によって行います．さらに章などの親カウンタに連動させたい場合は

```
\newtheorem{⟨環境名⟩}{⟨名前⟩}[⟨カウンタ名⟩]
```

のようにしますし，同系の環境を作成するときは

```
\newtheorem{⟨環境名⟩}[⟨同系の環境名⟩]{⟨名前⟩}
```

として定義します．theorem パッケージではさらにそれぞれの定理型環境の書式を以下の命令で変更できます．

```
\theoremstyle{⟨スタイル⟩}
\theorembodyfont{⟨書式⟩}
\theoremheaderfont{⟨書式⟩}
```

⟨書式⟩ に対しては書体変更用の宣言型の命令を使います．⟨スタイル⟩ には以下の六つが使えます．

plain　標準の\theorem 命令と同じ書式にします.

break　〈名前〉を出力した後に改行をします.

margin　通し番号を余白に出力します.

change　通し番号と〈名前〉を入れ替えます.

marginbreak　'margin' に付け加え，それを出力した後に改行します.

changebreak　'change' に付け加え，それを出力した後に改行します.

theorem パッケージで「例題 2.1，参考 2.2，問題 2.3」のような環境を作成したければ

```
{\theorembodyfont{\normalfont}
\theoremheaderfont{\normalfont\bfseries}
\newtheorem{Exam}{例題}
\newtheorem{Refer}[Exam]{参考}
\newtheorem{Prob}[Exam]{問題}}
```

とすると良いでしょう.

7.9　雑多なこと

不定積分を表現したり定積分を表現したりする次の場合を考えてみましょう.

```
%\usepackage{txfonts}
\[ \int f(x)dx + \int g(y)dy +
   \iint h(x,y)dxdy  \]
```

$$\int f(x)dx + \int g(y)dy + \iint h(x,y)dxdy$$

この場合は新規に\intx や\iintxy などを定義すると手間が省けるでしょう.

```
\newcommand{\intx}[1]{\int#1dx}
\newcommand{\inty}[1]{\int#1dy}
\newcommand{\iintxy}[1]{\iint#1dxdy}
\[\intx{f(x)}+\inty{g(y)}+\iintxy{h(x,y)}\]
```

$$\int f(x)dx + \int g(y)dy + \iint h(x,y)dxdy$$

あまり複雑な数式になるとマクロを書くよりも直接書いたほうが良いかも知れません.

ある線形微分方程式 $dy/dx + P(x)y = Q(x)$ の一般解を表現するために

```
\[ y=e^{-\int P(x)dx}\left\{
     \int{Q(x)e^{\int P(x)dx}dx+\mathrm{c}}
     \right\} \]
```

$$y = e^{-\int P(x)dx}\left\{\int Q(x)e^{\int P(x)dx}dx + \mathrm{c}\right\}$$

というのを何回も書くのはエネルギーの無駄ですから，公式通りに新規に命令を作ると汎用的に $P(x)$ や $Q(x)$ を書くことができます.

```
\newcommand{\my}{%
  \ensuremath{dy/dx+P(x)y=Q(x)}}
\newcommand{\mypq}[2]{\ensuremath{%
   e^{\int{#1}dx}\left\{\int{{#2}%
   e^{\int{#1}dx}dx+\mathrm{c}}\right\}}}
$P(x)=x^2+\pi$として$Q(x)=e^x$とすると{\my}の
一般解$y$は\[\mypq{(x^2+\pi)}{e^x}\] となる.
```

$P(x) = x^2+\pi$ として $Q(x) = e^x$ とすると $dy/dx+P(x)y = Q(x)$ の一般解 y は

$$e^{\int (x^2+\pi)dx}\left\{\int e^x e^{\int (x^2+\pi)dx}dx + \mathrm{c}\right\}$$

となる.

何らかの数式が公式として確立している場合はそれをマクロとして作成しておくと便利です．マクローリン展開やテイラー展開を毎回書くのは面倒ですから次のような使い方をすると良いでしょう．

```
\newcommand{\macl}[2][x]{\ensuremath{%
 f(#2)+\frac{1}{1!}f'(#2)(#1-#2)+%
 \frac{1}{2!}f''(#2)(#1-#2)^2+\cdots+%
 \frac{1}{k!}f^{(k)}(#2)(#1-#2)^k+\cdots}}
\newcommand{\Macl}[2][x]{\ensuremath{%
 \sum^{\infty}_{k=0}\frac{1}{k!}%
 f^{(k)}(#2)(#1-#2)^k}}
関数$f(z)$の$z=0$におけるテイラー展開は
\(\macl[z]{0}\) であり\(\Macl[z]{0}\)
となるので$z=0$における級数は\[
 f(z)=\sum^{\infty}_{k=0}\frac{1}{k!}
 f^{(k)}(0)z^k
\] となり，これをマクローリン展開と呼ぶ．
```

関数 $f(z)$ の $z=0$ におけるテイラー展開は $f(0)+\frac{1}{1!}f'(0)(z-0)+\frac{1}{2!}f''(0)(z-0)^2+\cdots+\frac{1}{k!}f^{(k)}(0)(z-0)^k+\cdots$ であり $\sum_{k=0}^{\infty}\frac{1}{k!}f^{(k)}(0)(z-0)^k$ となるので $z=0$ における級数は

$$f(z)=\sum_{k=0}^{\infty}\frac{1}{k!}f^{(k)}(0)z^k$$

となり，これをマクローリン展開と呼ぶ．

偏微分記号が多く出てくる数式を考えます．

```
\[ \frac{\partial{f}}{\partial{x}}+
   \frac{\partial^2{f}}{\partial{x}^2}+
   \frac{\partial^3{f}}{\partial{x}^3} \]
```

$$\frac{\partial f}{\partial x}+\frac{\partial^2 f}{\partial x^2}+\frac{\partial^3 f}{\partial x^3}$$

毎回このように記述するのは疲れますので次のようにマクロを作成して用います．

```
\newcommand{\pdif}[3][]{\ensuremath{\frac{%
  \partial^{#1}{#2}}{\partial{#3}^{#1}}}}
\[ \pdif{f}{x}+\pdif[2]{f}{x} \]
```

$$\frac{\partial f}{\partial x}+\frac{\partial^2 f}{\partial x^2}$$

このようにしても良いのですが，変数が二つ以上の場合は手動で対処します．

```
\newcommand{\pdif}[3][]{\ensuremath{\frac{%
  \partial^{#1}{#2}}{\partial{#3}^{#1}}}}
\[ \pdif[2]{f}{x} + \pdif{\sp2f}{xy} +
   \pdif[2]{f}{y} \]
```

$$\frac{\partial^2 f}{\partial x^2}+\frac{\partial^2 f}{\partial x y}+\frac{\partial^2 f}{\partial y^2}$$

\partial と\frac をごちゃごちゃ書くよりはこのほうがすっきりしているでしょう．

作成中の文書の分野を考えてあらかじめ公式の一部分をマクロとして作成するのも有効かも知れません．マクロは同じ文書の中で何度も出てくる公式どおりの数式には有効ですが，たった一度しか登場しないような数式に対してわざわざマクロを作成する必要はありません．

7.9.1 記号の積み重ね

イコール '=' のうえに 'def' をのせて '$\stackrel{\text{def}}{=}$' のような記号を出したいときがあります．これには\stackrel という命令が使えます．一つ目の引数を二つ目の引数のうえに載せて関係子を作ります．

\stackrel{⟨上の記号⟩}{⟨下の記号⟩}

```
\newcommand{\defeq}{%
   \stackrel{\mathrm{def}}{=}}
\( x \defeq p(t)+q(t)+r(t) \)
```

$x \stackrel{\mathrm{def}}{=} p(t) + q(t) + r(t)$

記号の積み重ねとは少し違うのですが，次のような数式を出力するときもあるでしょう．この例では\substack という amsmath パッケージに含まれる命令を使っています．

```
\begin{displaymath}
 \sum^l_{i=1} \sum^m_{j=1} \sum^n_{k=1}
   p_i q_j r_k \neq \sum_{
   \substack{i\le 1\le l \\ j\le 1 \le m
   \\ k\le 1 \le n}} p_i q_j r_k
\end{displaymath}
```

$$\sum_{i=1}^{l}\sum_{j=1}^{m}\sum_{k=1}^{n} p_i q_j r_k \neq \sum_{\substack{i\le 1\le l\\ j\le 1\le m\\ k\le 1\le n}} p_i q_j r_k$$

7.9.2　記号の重ね合わせ

二つの記号を重ね合わせて新しい記号を作りたいときがあります．\ooalign と\crcr 命令を組み合わせるとうまくできます．

{\ooalign{⟨一つ目⟩\crcr⟨二つ目⟩}}

二つの記号の内で横幅の広いほうの幅が優先されます．二つの記号を中心に重ね合わせたいときは\hss という空白を挿入する命令を使います．さらに文字列に\not を使っても演算子の否定のようにはなりませんので

```
\newcommand{\cnot}[1]{\ooalign{／\crcr{\hss{#1}\hss}}}
```

のような定義をしておくと良いでしょう．スラッシュは全角を使っています．

```
\newcommand{\pile}[2]{%
  {\ooalign{#1\crcr#2}}}
\newcommand{\cpile}[2]{{\ooalign{{%
  \hss#1\hss}\crcr{\hss#2\hss}}}}
\newcommand{\cnot}[1]{%
  \ooalign{／\crcr{\hss{#1}\hss}}}
ほげほげ$\pile Y=$は定数$\cpile Y=$のほげで
あり，\cnot{A}は\pile/A とは別物なのである．
```

ほげほげ ¥ は定数 ¥ のほげであり，$\not A$はA とは別物なのである．

7.9.3　数式の太字

何らかの理由である数式の一部や，ある数式全体を太字にすることがあるそうです．方法として

- \mathbf 命令を使う．

- \boldmath と\unboldmath を使って太字かどうかを切り替える.
- amsmath に含まれる amsbsy パッケージの\boldsymbol 命令を使う.
- bm パッケージの\bm 命令を使う.

などがあります. これは使用している数式書体によっては使えないことがあります. txfonts や pxfonts を使うとなんら問題なく出力できます. 一つ目の\boldmath と\unboldmath は**数式モード中で使うことができません**.

```
\(\mathbf{\int^a_b f(x)dx}  \neq\)
  \boldmath  \(\int^a_b f(x)dx  \neq\)
  \unboldmath\(\int^a_b f(x)dx \)
```

$$\mathbf{\int^a_b f(x)dx} \neq \boldsymbol{\int^a_b f(x)dx} \neq \int^a_b f(x)dx$$

\mathbf の場合はギリシャ文字などの特定の記号しか太字にならないうえにイタリック体ではなくローマン体になってしまいます. もう少し局所的に使いたい場合は amsbsy の\boldsymbol を使います.

```
\(\mathbf{\int^a_b f(x)dx}     \neq
  \boldsymbol{\int^a_b f(x)dx} \neq
  \int^a_b f(x)dx \)
```

$$\mathbf{\int^a_b f(x)dx} \neq \boldsymbol{\int^a_b f(x)dx} \neq \int^a_b f(x)dx$$

amsbsy を使うよりも bm パッケージの\bm を使うほうが安全です.

```
\(\mathbf{\int^a_b f(x)dx}     \neq
  \bm{\int^a_b f(x)dx} \neq
  \int^a_b f(x)dx \)
```

$$\mathbf{\int^a_b f(x)dx} \neq \boldsymbol{\int^a_b f(x)dx} \neq \int^a_b f(x)dx$$

結論として\bm 命令を使うようにすると思い通りの結果になるのではないかと思います.

7.9.4 高さを揃える

ルート記号などを使っているとルートの高さが揃わずに見栄えが悪くなるときがあります. これには数式中でルートなどの高さを揃える\mathstrut 命令が使えます.

```
\[ \overline{\sqrt a + \sqrt b  \neq
   \sqrt{\mathstrut a}+
   \sqrt{\mathstrut b}} \]
```

$$\overline{\sqrt{a} + \sqrt{b} \neq \sqrt{a} + \sqrt{b}}$$

分かりづらいのですが実は高さのみならず, 深さも\mathstrut によって自動的に調整されています.

もう少し高度な命令として\phantom, \vphantom, \hphantom の三つが用意されています. \phantom 命令は引数に与えられた要素だけの高さと幅と深さを持った空白を作成します. \vhpantom は引数に与えた要素の高さと同じ目には見えない箱を作成します. \hphantom はその横方向バージョンです.

```
\[ \sqrt{\int f(x)dx}+\sqrt{g(x)}\neq
   \sqrt{\int f(x)dx}+\sqrt{%
   \vphantom{\int f(x)dx} g(x)}  \]
```

$$\sqrt{\int f(x)dx}+\sqrt{g(x)}\neq\sqrt{\int f(x)dx}+\sqrt{g(x)}$$

もう一つ\smash という命令もあり，これは引数に与えられた要素の高さと深さを 0 にする魔法のようなものです．\smash と\vphantom を組み合わせると要素の幅はそのままで高さと深さを 0 にしたうえで\vphantom で指定した高さと深さの見えない箱を作成できるので，**高さや深さを揃えるのに**使えます．

```
\[ \underbrace{a+b}+\underbrace{i+j}\neq
   \underbrace{\smash{a+b}\vphantom{i+j}}
    + \underbrace{i+j}   \]
```

$$\underbrace{a+b}+\underbrace{i+j}\neq\underbrace{a+b}+\underbrace{i+j}$$

7.9.5　スマートな分数の書き方

文中数式中で分数を出力する\frac 命令を使うと $\frac{a}{b}$ となります．このような分数の書き方はスマートではありません．a/b と書くと一般的な文中の分数のスタイルとなります．

```
\[ \frac{\frac{a}{b}}{c}\neq
   \frac{a/b}{c}             \]
```

$$\frac{\frac{a}{b}}{c}\neq\frac{a/b}{c}$$

このような分数のスタイルは別行数式にも当てはまります．別行数式において分数を記述しており，その分母・分子上にさらに分数を書く，連分数を記述する場合などはスラッシュ '/' による表記をするとスマートになります．ただしスラッシュによる表記では**適宜丸括弧を補います**．

```
\begin{displaymath}
 \frac{\frac{a-b}{c}}{d} \neq
 \frac{a-b/c}{d} \neq  \frac{(a-b)/c}{d}
\end{displaymath}
```

$$\frac{\frac{a-b}{c}}{d}\neq\frac{a-b/c}{d}\neq\frac{(a-b)/c}{d}$$

```
\begin{displaymath}
\frac{x+f(x)}{x-g(x)} \neq
(x+f(x))/(x-g(x))     \neq
\bigl(x+f(x)\bigr)\bigm/\bigl(x-g(x)\bigr)
\end{displaymath}
```

$$\frac{x+f(x)}{x-g(x)}\neq(x+f(x))/(x-g(x))\neq(x+f(x))\,/\,(x-g(x))$$

7.9.6　場合分けなど

一つの式から解が複数に**場合分け**される場合\cases 命令が使えますが amsmath の cases 環境のほうがうまく行くでしょう．

```
\begin{cases}
要素 1 \\ 要素 2 \\ ...
\end{cases}
```

```
\( f(x) = \begin{cases}
  \,x & \quad(x>0)\\ \,0 & \quad(x=0)\\
  \,-x & \quad(x<0)
  \end{cases} \)
```

$$f(x) = \begin{cases} x & (x > 0) \\ 0 & (x = 0) \\ -x & (x < 0) \end{cases}$$

他にも\choose のように要素を縦に並べて括弧を付ける命令があります.

\choose （丸括弧付き）
\brack（角括弧付き）
\brace（波括弧付き）
\atop（括弧なし）

\choose などは全体を波括弧で括ってあげるとうまく出力できます.

```
\[ {a+b\choose x+y}\neq
   {a+b\brack x+y} \neq
   {a+b\brace x+y} \neq
   {a+b\atop x+y}  \]
```

$$\binom{a+b}{x+y} \neq \begin{bmatrix} a+b \\ x+y \end{bmatrix} \neq \begin{Bmatrix} a+b \\ x+y \end{Bmatrix} \neq \genfrac{}{}{0pt}{}{a+b}{x+y}$$

7.9.7 数式モード中の空白と書体

数式用の環境では自動的に要素の前後の記号の種類になどにより空白が調節されますから意図していた結果と異なる場合があります.

```
\emph{fool}は\(fool\) にはなりませんから
\[ fool \neq \mathit{fool}. \]
```

fool は $fool$ にはなりませんから

$$fool \neq fool.$$

‘fool’ という文字が全て数式中では変数と解釈され，それぞれ LaTeX が適切だと思う空白を挿入してくれています．これから分かるように数式モード中ではユーザが明示的に空白を調節すると良い場合があります.

```
$10\times5,000=50,000$円も払えるか！\par
$10\times5{,}000=50{,}000$円も払えるの？\par
```

$10 \times 5,000 = 50,000$ 円も払えるか！
$10 \times 5{,}000 = 50{,}000$ 円も払えるの？

上記の例ではコンマ ‘,’ が恐らく何かの区切りとして解釈されたのでしょう，意図していたものよりも広くなっています．同じように感嘆符 ‘!’ などは逆に空白が挿入されません．ですから\, 命令で若干の空きを挿入します.

```
\[  \frac{(p-1)! (q-1)! (r-1)!}
  {p! q! r!} \neq
\frac{(p-2)!\,(q-3)!\,(r-4)!}
  {\,p!\,q!\,r!}  \]
```

$$\frac{(p-1)!(q-1)!(r-1)!}{p!q!r!} \neq \frac{(p-2)!\,(q-3)!\,(r-4)!}{\,p!\,q!\,r!}$$

感嘆符 ‘!’ の例を見ると分かりますが数式モード中では斜体になっていません．このように数式モード中でも斜体にならない記号がいくつかあります．\textit では記号もイ

タリック体になりますが数式中の\mathit を使うといくつかの記号が斜体にならないばかりか，空白制御が行われません．

```
\textit{This is text mode?!}\par
\(\mathit{Is\ this\ text\ mode?!}\)\par
\(\mathit{Is this text mode?!}\)\par
\(Is this text mode?!\)
```

This is text mode?!
Is this text mode?!
Isthistextmode?!
Isthistextmode?!

いずれの場合も疑問符 '?' はイタリック体にはなっていません．このように数式中では明示的にイタリック体に書体を変更する命令を使ってもローマン体のままの記号があります．

7.9.8　ダイヤグラムの例

これはただの遊びで作ったものですからあまり参考にしないでください．このような無謀なこともできるという程度に見てください．

```
\newcommand{\law}[1]{\mathop{\hbox%
   to5em{\rightarrowfill}}\limits#1}
\newcommand{\raw}[1]{\mathop{\hbox%
   to5em{\leftarrowfill}}\limits#1}
\newcommand{\rar}[2]{%
   \Biggm#1{\scriptstyle#2}}
\newcommand{\lar}[2]{%
   {\scriptstyle#2}\Biggm#1}%%
\[ \begin{array}{rcl}
A & \law{\sp a} & B\\
\lar{\downarrow}{f} & \rule{5em}{0pt}
  & \rar{\uparrow}{g} \\
C & \raw{\sb b} & D\\
\end{array} \]
```

$$\begin{array}{rcl}
A & \law{\sp a} & B\\
\lar{\downarrow}{f} & \rule{5em}{0pt}
  & \rar{\uparrow}{g} \\
C & \raw{\sb b} & D\\
\end{array}$$

第 8 章

図表の組版

「渡辺君，先生はもう今年で 63 になるけど，先生のころは論文なんてみんな手書きだったからね，図も自分で雲形定規を使って描いてたよ．それに図にはスクリーントーンも工夫して貼ってたよ，懐かしいなぁ．君たちの世代はそんなことやったことないんだろうねぇ．」（渡辺徹『雲のうえの声』より）

数十年前までは論文作成に必要なものといえば糊とはさみとカッターなどのアナログな道具でした．今ではほとんどの論文を電子的に提出できるようになりました．LaTeX ではある程度の手順を踏めば非常に高品質に電子的な図表の組版が可能です．

8.1 図表の基礎

図や表を LaTeX では浮動体（表 8.1） と呼ばれる場所に一度退避させることができます．このようにすると LaTeX は LaTeX 自身が最適だと思われる場所に図表を出力しようと努力します．浮動体として退避させた図や表は少し制限の多い条件で組版されます．表は tabular 環境で作成し，番号付けしたければ table 環境中に入れ子にします．図は picture 環境や画像ファイルを指定し，番号付けしたければ figure 環境中に入れ子にします．このようにするとそれらの図表は浮動体として扱われます．レポートや論文では図表に通し番号を付けるのは必須ですから，全ての表は table 環境の中へ，図は figure 環境の中に入れるのが良いでしょう．

表 8.1　浮動体の種類

	表	図
入れる環境	table 環境	figure 環境

LaTeX が内部でどのように，これら浮動体を配置しているのかという難しい部分を押さえなければ自分の思い通りの位置に図表を配置することができない場合が多いでしょう．余り図の出力位置などを気にしなければそれで良い問題です．

浮動体を挿入するときに指定するのはその配置場所です．基本的に LaTeX は図表をページの最上部か最下部に配置しようとして，それでも無理なときは別ページへと出力します．ユーザーはこれら浮動体の配置場所を指定することが出来ます．指定できる場所は表 8.2 となります．位置指定は複数指定することが可能です．これらの位置指定は table 環境や figure 環境の任意引数として渡します．figure 環境で例を示すと

```
\begin{figure}[htbp]
ここに図が入ります．
```

表 8.2　浮動体の位置指定

記号	浮動体の配置する場所
h	まさにその場所に配置しようと試みます
t	ページ上部に配置しようと試みます
b	ページ下部に配置しようと試みます
p	浮動体を別ページに配置しようと試みます
!	無理やりその場所に配置します

```
\end{figure}
```

のように使います．

一つここで注意しなければならないことは，図表の見出しの位置です．**図見出しは図の下部に，表見出しは表の上部に**見出しをつけます．これについては**図表見出し**を出力する\caption 命令の位置を変えるだけです．

```
\caption{〈図表見出し\label{〈ラベル〉}〉}
```

figure 環境中に表を入れたり，table 環境中に図を入れたりすることが出来ます．他にも環境中に文字列を挿入することも可能です．

8.2　表

LaTeX で表を作るための環境は

tabbing 環境　タブを制御することによって表を作成する．

tabular 環境　高度な表も作成することができる汎用的な表作成環境．

array 環境　tabular 環境と機能は類似しているが数式の行列などに使われることが多い．

の三つが用意されております．array 環境は 7.5.5 節にて紹介していますのでそちらを参照してください．tabbing 環境も簡単に表が作成できる環境なのですが，tabular のほうが記述が楽だと思いますので，ここでは tabular のみを紹介します．tabular 環境は次のように記述します．

```
\begin{tabular}{列指定子}
 a_11  &  ...  &  a_1n  \\
  ⋮    &  ⋱    &   ⋮    \\
 a_m1  &  ...  &  a_mn
\end{tabular}
```

行列とほぼ同じです．違うのは数式環境には入れなくても良いということです．

列指定子とはその tabular 環境における表の列数や縦方向の罫線などを決めるものです．tablar 環境で使用できる主な列指定子は表 8.3 の通りです．tabular 環境における

表 8.3 tabular 環境の主な列指定子

列指定子	意味
l	行列の縦 1 列を左揃えにする
c	行列の縦 1 列を中央揃えにする
r	行列の縦 1 列を右揃えにする
|	縦の罫線を引く
||	縦の 2 重罫線を引く
@{**表現**}	表現を縦 1 列追加します
p{**長さ**}	ある列の幅の長さを直接指定します
*{**回数**}{**項目**}	回数分だけ項目を繰り返す．

各要素（成分）はアンド ‘&’ で区切ります．‘\\’ を行の終わりとしますので例えば 1 行 3 列の表は次のようになります．

```
\begin{tabular}{ccc}
 \LaTeX2.09 & \LaTeXe & \LaTeX3\\
\end{tabular}
```

LaTeX2.09 LaTeX 2ε LaTeX3

横方向に罫線を引くには\hline，要素の中で縦の罫線を引くときには\vline などを使います（表 8.4）．横方向の罫線を引くには\hline を，行を連結するには\multicolumn

表 8.4 tabular 環境中での罫線の命令

命令	意味
\hline	横に引けるだけの罫線を引きます
\hline\hline	引けるだけの 2 重の横罫線を引きます
\vline	要素の中で引けるだけの縦罫線を引きます
\cline{⟨範囲⟩}	要素の罫線を行の範囲を指定して引きます
\multicolumn{⟨数値⟩}{⟨列指定子⟩}{⟨要素⟩}	行をつなげて列指定子通りに要素を出力します

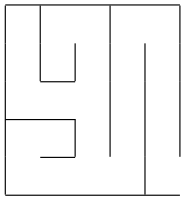
を使います．

```
\begin{tabular}{|c|c|c|}
\hline
\multicolumn{3}{|c|}{\LaTeX} \\\hline
\LaTeX2.09 & \LaTeXe & \LaTeX3 \\\hline
\end{tabular}
```

LaTeX		
LaTeX2.09	LaTeX 2ε	LaTeX3

罫線を利用して迷路のようなものも作れます．

```
\begin{tabular}{|ccc|c|c|}
\hline
& \multicolumn{1}{|c}{ } & &
      \multicolumn{1}{c}{} &  \\
& \multicolumn{1}{|c|}{} & & & \\
\cline{2-2}
 &  &  &  &   \\\cline{1-2}
& \multicolumn{1}{c|}{} &  & & \\
\cline{2-2}
&  & \multicolumn{1}{c}{} & &
      \multicolumn{1}{c}{}  \\
\hline
\end{tabular}
```



レポートや論文では表には表見出しを付けて中央揃えにするのが望ましいと思われますので以下のようなフォーマットになります．

```
\begin{table}[htpb]
 \begin{center}
 \caption{表の出力例}\label{tab:tabular:example}
  \begin{tabular}{llcr}
   \hline
   出力例 & 1 & 2 & 3 \\ \hline
   \LaTeX の遷移& \LaTeX2.09  & {\LaTeXe}& \LaTeX3 \\\hline
  \end{tabular}
 \end{center}
\end{table}
```

上記のソースの出力例が表 8.5 となります．毎回このような記述をしていたのでは疲れま

表 8.5　表の出力例

出力例	1	2	3
LaTeX の遷移	LaTeX2.09	LaTeX 2ε	LaTeX3

すので，表用の mytab 環境を次のように定義します．

```
\newenvironment{mytab}[3][htbp]
  {\begin{table}[#1]\begin{center}\caption{#2}\label{#3}}
  {\end{center}\end{table}}
```

こう定義しておけば

```
\begin{mytab}[htbp]{中央揃えで見出しのある表の環境}{tab:hoge}
\begin{tabular}{lll}
\LaTeX2.09 & \LaTeXe & \LaTeX3\\
\end{tabular}
\end{mytab}
```

のように使うことができるわけです．

8.2.1 表中の脚注

tabular 環境中での脚注はうまく出力できないことが多いようです．その場合は\footnotemark と\footnotetext の二つを使います．

\footnotemark[⟨番号⟩]
\footnotetext[⟨番号⟩]{⟨注釈内容⟩}

\footnotemark で脚注記号を表示し，\footnotetext に注釈を書きます．

```
\begin{tabular}{|c|c|c|} \hline
  一つ目\footnotemark[1] &
  二つ目\footnotemark[2] &
  三つ目\footnotemark[3]\\ \hline
\end{tabular}
\footnotetext[1]{表中一つ目の脚注です．}
\footnotetext[2]{表中二つ目の脚注です．}
\footnotetext[3]{表中三つ目の脚注です．}
\\ちょっと表示が変になっています．
```

一つ目[*1]	二つ目[*2]	三つ目[*3]

ちょっと表示が変になっています．

[a] 表中一つ目の脚注です．
[b] 表中二つ目の脚注です．
[c] 表中三つ目の脚注です．

上記の方法ではうまくいかない場合は手動で脚注を付けることもできます．

```
\begin{tabular}{|c|c|c|}\hline
  一つ目${}^{a}$ &
  二つ目${}^{b}$ &
  三つ目${}^{c}$\\\hline
\end{tabular}
\\{\small${}^{a}$表中一つ目の脚注です．}
\\{\small${}^{b}$表中二つ目の脚注です．}
\\{\small${}^{c}$表中三つ目の脚注です．}
```

一つ目 [a]	二つ目 [b]	三つ目 [c]

[a] 表中一つ目の脚注です．
[b] 表中二つ目の脚注です．
[c] 表中三つ目の脚注です．

8.2.2 tabular 環境の出力位置

実は tabular 環境は列指定子の前に任意引数を一つとります．これは表の位置と段落の位置を調整するものです．tabular 環境で作成された表の上部と段落の位置を合わせるときは ‘t’ を，下部ならば ‘b’ を，中央ならば ‘c’ を選びます．

```
\newcommand{\testtab}[1][c]{~ほげ~
  \begin{tabular}[#1]{|c|} \hline
  ほげ\\ ほげ\\ \hline\end{tabular}}
\testtab \testtab[t] \testtab[b]
```

ほげ | ほげ
ほげ | ほげ | ほげ
ほげ | ほげ | ほげ
ほげ

8.2.3　表作成支援ツール

LaTeX で 0 から表を組むのは初心者には辛いかもしれません．GUI ベースのプログラムで表を作成し，それを LaTeX の tabular 環境の記述に変換するツールを使うと良いでしょう．Microsoft の Excel を使っている場合は中尾誠氏の Excel2latex

http://www32.ocn.ne.jp/~butcher_bird/Mac/Excel.html

や浦壁厚郎氏の Exel2tabular

http://www.ne.jp/asahi/i/love/E2T/

などがありますので参考にしてください．これらのプログラムは Microsoft の Excel で作成された表を LaTeX のソースに変換します．

Microsoft の Excel ではなく OpenOffice.org の Calc を使っているならば阿部昌平氏の Calc2LaTeX

http://web.hc.keio.ac.jp/~fr000056/calc2latex/indexj.html

というものがあるそうです．これを使えば Calc で作成した表を tabular 環境に変換し，表として LaTeX に貼り付けることが出来ます．

8.3　図

なんといっても LaTeX では画像に関する多くの処理をデバイスドライバに頼るしかないので，自分の使おうとしているデバイスドライバがどのような画像処理に対応しているのかを知っておきましょう．PDF を作成したいならば Dvipdfm*x*，PostScript ならば dvips を使うと良いでしょう．

図の挿入に関しての方法は大きく分けて 2 通りあります．一つはペイントソフトなどで書いた画像をそのまま取り込む方法，もう一つは LaTeX の picture 環境で図を直接書く方法です．二つ目の方法の詳しいことは『LaTeX コンパニオン』[29] と『LaTeX グラフィックスコンパニオン』[30] を参照してください．一般的な図の取り込み方としては別のソフトウェアで Encapsulated PostScript（EPS）形式で画像を保存し，それを graphicx パッケージで読み込むのではないかと思われます．商用のソフトウェアであれば大抵で EPS 出力をサポートしています．現在確認できているだけも，Photoshop，Illustrator，Gnuplot，Mathematica，Matlab などがあります．そのほかにも汎用 CAD ソフトなども対応しているようです．Unix 系 OS の場合 LaTeX と相性が良いと思われるのは Tgif や Ngraph でしょう．それらを Windows 環境にインストールするには少々骨の折れる作業ですが，Cygwin 上の XFree86 で日本語のフォントや Canna などを一緒にインストールするとできるそうです．

8.3.1 EPS 画像の張り込み

EPS とは Encapsulated PostScript の略で Adobe 社の開発したページ記述言語で，PostScript にプレビュー用の情報を付加したファイル形式です．単一ページでベクトル画像を保存するのに良く使われるます．EPS 形式の画像の場合は dvips を使うとファイル形式の変換を必要とせずにそのまま PS ファイルに取り込むこともできます．Dvipdfm*x* などは 1 度 Ghostscript の pdfwrite を使って EPS から PDF に変換した PDF ファイルを取り込みます．

LaTeX での基本的な EPS 画像の扱い方を説明します．

1. 各アプリケーションで EPS 保存オプションをモノクローム 256 色（8 ビット），PostScript レベル 2 などに設定してから保存します．フォントはアウトライン化（ベクトル化）できる場合は行います．カラー印刷する場合はカラーで構いませんが容量は大きくなります．
2. 文書のプリアンブルで graphicx パッケージを使うことを宣言します．
   `\usepackage[`**デバイスドライバ**`]{graphicx}`
   PS 形式の文書を出力するならば，*dvips* を指定します．PDF を作成したいときは Dvipdfm*x* を使うために *dvipdfm* を指定します．
3. 図を挿入したい場所に`\includegraphics` 命令を使ってファイル名を示します．
   `\includegraphics[`**設定**`]{`**ファイル**`}`
   仮引数には表 8.6 などのオプションが使えます．

表 8.6 graphicx パッケージで使える主なオプション

設定項目名	説明	値
`width=`⟨幅⟩	出力する図の幅を指定します	長さ，100 mm など
`height=`⟨高さ⟩	出力する図の高さを指定します	長さ，100 mm など
`angle=`⟨角度⟩	反時計回りの回転角度を指定します	0 < 値 < 360
`scale=`⟨数値⟩	図の拡大・縮小率を指定します	0 < 値

例えば Ghostscript の下には `examples` というディレクトリがあります．そこに `golfer.eps` という EPS 画像があります．この EPS 画像ファイル `golfer.eps` を LaTeX の文書に張り込むには `golfer.eps` を LaTeX の原稿のあるディレクトリにコピーして

```
\usepackage[]{graphicx}
```

をプリアンブルに書きます．そして `document` 環境の中で

```
\includegraphics[width=5cm]{golfer.eps}
```

とします．これを platex でタイプセットし，整形された DVI ファイル ⟨*file*⟩`.dvi` をプレビューアで見ることができます．

上記の操作が問題なくできるでしょうか．基本的に LaTeX で既存の画像を張り込むときは LaTeX に標準で含まれている graphicx パッケージを使うことになると思われます．

既存の JPG，BMP，EPS などの画像は LaTeX の力ではどうしようもありありません．それらの画像に関してはデバイスドライバに一任しています．そのため取り込むことのできる画像はデバイスドライバに依存します．それでも画像の張り込みに関しては graphicx パッケージという統合的な方法が提供されています．このパッケージを使うことによって任意のデバイスドライバに合わせたコマンドを記述しなくても良いように工夫がされています．

まずこの graphicx を使うためにプリアンブルに以下の 1 行を追加します．

```
\usepackage{graphicx}
```

このとき使用するデバイスドライバが重要で最終的に PostScript 形式か，DVI 形式か，それとも PDF 形式が必要なのかによってオプションを変えます．Windows で印刷することも考えるならば最終的に PDF 形式にすると良いのかもしれません．そうするとデバイスドライバは Dvipdfm*x* を使うことになるでしょうから，Dvipdfm*x* の場合は

```
\usepackage[dvipdfm]{graphicx}
```

とします．Dvipdfm*x* の場合も graphicx においては *dvipdfm* と同じ設定になりますから *dvipdfmx* と書いてはいけません．

Unix 系 OS ならば PostScript のほうが良いでしょうから

```
\usepackage[dvips]{graphicx}
```

とします．dvipsk であろうが pdvips だろうが *dvips* オプションを使います．他には xdvi や dviout も指定できます．dviout の場合は dviout がインストールされているフォルダの GRAPHIC/LATEX2E/dviout.def というファイルを$TEXMF/tex/latex/graphics/にコピーしてください．

手持ちの画像の形式を判断して使用するデバイスドライバを考えるのが良いでしょう．標準的に指定できるデバイスは

dvips xdvi dvipdf dvipdfm pdftex dvipsone dviwindo emtex dviwin
pctexps pctexwin pctexhp pctex32 truetex tcidvi vtex oztex textures

などです．デバイスドライバによっては graphicx 用の独自の設定ファイル 〈ドライバ〉.def が含まれていることもあるので確認してください．

既存の画像は基本的に\includegraphics 命令で読み込みます．

\includegraphics[〈設定〉]{〈ファイル名〉}

Windows の方で手持ちの画像のほとんどがビットマップで存在するならば dviout をデバイスドライバに選択すれば良いでしょう．EPS 画像が多いならば 1 度 EPS から PDF に変換してから Dvipdfm*x* を使うのが良いと思われます．Unix 系 OS ならば手持ちの画像

を EPS に変換して dvips を使うことになるでしょう．試しにご自分の持っている画像〈ファイル〉を〈デバイス〉で取り込めるのかを試してみください．

```
\documentclass{jarticle}
\usepackage[デバイス]{graphicx}
\begin{document}
\includegraphics{ファイル}
\end{document}
```

dviout の場合 EPS 画像を取り込むときは Ghostscript にて EPS を PDF に変換してから画像を表示しますから dviout の Ghostscript に関する設定を適切に行ってください．画像によってはページをはみ出したりしている場合があるでしょうし，表示が大きすぎる場合があるでしょう．その場合は取り込みに関する設定をします．

height=〈高さ〉 単位付きで画像の高さを指定します．

totalheight=〈総合的な高さ〉 単位付きで画像の総合的な高さを指定します．

width=〈幅〉 単位付きで画像の幅を指定します．

scale=〈数値〉 画像の拡大率を指定します．

angle=〈角度〉 **反時計回りに**画像を回転する角度を指定します．

bb=〈画像の位置情報〉 画像のどの領域を使うべきかを指定します．‘bb=0 0 640 480’とすると原点を $(0, 0)$ として縦横 ‘640×480’ の領域を使うようにします．

noclip 画像用に使うべき領域を元の画像がはみ出している場合に画像を切り抜かないようにします．

clip 画像が確保された領域よりも大きい場合は切り抜きします．

draft 実際に画像を張り込まずに画像が占有するだろう領域を枠による代替表示になり，ファイル名も表示します．

keepaspectratio 拡大縮小したときに縦横比を保存するようにします．graphicx パッケージの標準では保存されます．

レポートや論文などで図には**図見出し**を付けて**中央揃え**にするのが望ましいと思われますので

```
\begin{figure}[htbp]
 \begin{center}
   \includegraphics[width=10cm]{images/file.eps}
   \caption{図見出し}\label{fig:samplefig}
 \end{center}
\end{figure}
```

のように使うことになるでしょう．ただし，これを毎回書くのは面倒なので次のような図用の myfig 命令を作成します．

```
\newcommand{myfig}[4][width=.8\textwidth]{%
\begin{figure}[htbp]%
   \centering\includegraphics[#1]{#2}%
```

```
    \caption{#3}\label{fig:#4}%
  \end{figure}}
```

このように定義しておけば次のように使えます．

```
以上の考察から図~\ref{fig:sample}のような
図が得られる．
\myfig[width=100pt,clip]{images/file.eps}{図の張り込みの例}{sample}
```

浮動体の図は DVI ファイルに出力されるときに思いもよらない場所まで旅をしますので，思い通りの場所に図を出力できなくても気にしないでください．そもそも図表に対して「上記の図はなんちゃら」とか「下記の図はなんちゃら」という表現は間違いで，全ての図表は「図 3.1 はなんちゃら」のように番号で参照します．ですから本来は図や表がどのような場所に旅立っても困らないはずです．

図などを反時計回りに 90°回転させることがあるでしょう．その場合は\rotatebox 命令を使います．他にも便利な命令があります．

\rotatebox[⟨設定⟩]{⟨角度⟩}要素

これは\includegraphics の任意引数に ‘angle’ を使ったことと同じです．\rotatebox は図に限らずあらゆる要素を回転します．⟨設定⟩ の項目には以下のようなものがあります．

origin=⟨ラベル⟩　要素を回転するための原点を指定します．左 ‘l’，右 ‘r’，中央 ‘c’，上部 ‘t’，下部 ‘b’ が指定できます．

x=⟨長さ⟩　x 方向の原点の位置を直接 ⟨長さ⟩ を指定します．

y=⟨長さ⟩　y 方向の原点の位置を直接 ⟨長さ⟩ を指定します．

```
\rotatebox{70}{文字列など}の
\rotatebox[origin=c]{60}{回転とか}は
\rotatebox[origin=b]{50}{どうよ？}
\rotatebox{30}{だめぽ．}
```

文字列など の 回転とか は どうよ? だめぽ.

要素を**拡大縮小**するには\scalebox を使います．

\scalebox{⟨横の拡大率⟩}[⟨縦の拡大率⟩]{⟨要素⟩}

⟨拡大率⟩ には長さを指定します．

```
\scalebox{2.3}{拡大縮小}\par
\scalebox{3}[1]{拡大縮小}
```

拡大縮小
拡大縮小

要素の**反転**には\reflectbox を使います．

\reflectbox{⟨要素⟩}

```
\reflectbox{文字列の反転}\par
\reflectbox{山は山}\par
\scalebox{-1}[1]{これも反転}
```

文字列の反転
山は山
これも反転

リサイズには\resizebox を使います.

\resizebox{〈幅〉}{〈高さ〉}{〈要素〉}

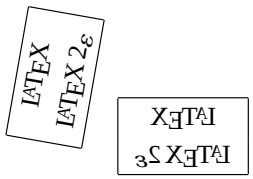
要素のリサイズ後の幅を〈幅〉に，高さを〈高さ〉にします．どちらか一方の拡大・縮小率に合わせたいときは‘!’を使います.

```
\resizebox{!}{1cm}{リサイズ}\par
\resizebox{3cm}{!}{リサイズ}
```

リサイズ
リサイズ

以上の\rotatebox，\scalebox，\reflectbox，\resizebox は文字列，表，図，minipage 環境などの段落などにも使えます.

```
\newcommand{\testtab}{\begin{tabular}{|c|}
\hline \LaTeX\\ \LaTeXe \\ \hline %
\end{tabular}}
\rotatebox{80}{\testtab}~
\reflectbox{\testtab}
```



8.3.2 他のプログラムから張り込む方法その 1

例えば Excel や PowerPoint などのグラフや図表を張り込む場合大きく分けて次の 3 通りの方法があります．対象となるグラフや図表，ページは基本的に PS 形式の画像に変換することができます．この小節の話は Windows に限定します.

1. 該当のページなどをスクリーンショットでビットマップ画像として保存しその画像を EPS-Conv や ImageMagick で EPS 画像に変換してから張り込む.
2. ページなどの大きさを調整してから PostScript プリンターを通してプリンターファイルとして保存し，バウンディングボックスを eps2eps で調整してから張り込む.
3. ページなどの大きさを A4 の紙いっぱいにファイルとして印刷し，それを graphicx パッケージで回転と拡大・縮小をして張り込む.

一つ目と二つ目は手間がかかるうえに印刷品質も悪いのでここでは説明しません．三つ目の方法を解説します.

手順は以下の四つのようになります．Microsoft の Excel のグラフを取り込む手順を例に出しています.

1. まず，Windows の[設定]で[プリンタの追加]をします．追加するプリンタは「ローカルプリンタ」で使用するポートは「ファイルへ出力」を選択し，ここでは「HP

Color Laser Jet PS」というプリンタソフトウェアを選択します．テストページを印刷する場合は，出力ファイル名を ‘hoge.eps’ のように拡張子を.eps にして保存します．正しく印刷されいるかどうかを確認するには Ghostscript か GSView などで見ることができます．

2. 次に元となるグラフやページを作成します．このときモノクロ印刷を想定してグラフを調整します．印刷したいグラフを選択したならば，[ページ設定]で印刷の向きは「横」，用紙サイズは「A4」，[余白]タブで上下左右，ヘッダー，フッターの余白をすべて「0」に設定し，[ヘッダー，フッター]タブで何も出力しないことを選択し，[グラフ]タブで，印刷するグラフのサイズを「用紙サイズ」に合わせます．印刷品質は「白黒印刷」にするとファイルサイズも小さくなり処理速度も向上します．印刷プレビューで問題がないことを確認したならば，アプリケーション側での設定は終了です．
3. [ファイル]から[印刷]を選び，プリンタの名前を先程追加したプリンタに設定します．‘OK’ ボタンを押し絶対パスでファイル名を指定してファイルへ出力すれば，目的のグラフを EPS で保存することが出来ます．このとき保存するファイル名は半角英数字とし拡張子は..eps としてください．ファイルの保存先を原稿と同じ場所にすると問題も少ないでしょう。
4. 出力した filename.eps を LaTeX 文書中で参照します．graphicx パッケージを使って

   ```
   \includegraphics[scale=0.4,angle=-90]{filename}
   ```

   とすれば丁度良いあんばいで取り込めます．A4 の紙にグラフを出力した場合，0.4 倍の縮小率で，時計回りに 90 度回転させると LaTeX 文書の中で丁度良いサイズとして張り込むことが出来ます．

以上の 2–4 の作業をグラフの数だけ繰り返すことにより，LaTeX に対して Excel で作成されるグラフとほぼ同じイメージを挿入することが可能です．フォントがビットマップフォントでギザギザになるときはプリンタの設定で「TrueType フォントダウンロードオプション」のような項目があるはずですので，これを「アウトライン」にすると良いでしょう．

8.3.3　他のプログラムから取り込むとき

Mathematica や Illustrator からグラフや画像を取り込むときには幾つかコツが必要です．135 ページ 8.3.2 節 での Excel での張り込み方が他のアプリケーションでも適用できる場合が多いので，上記の方法を試してみてください．

どのプログラムを使用していても最終的に出力したい画像のサイズを元のプログラム側で調節してから LaTeX に張り込むようにすると問題も少ないでしょう．graphicx パッケージの拡大縮小を使うと印刷品質が落ちます．各プログラムにおける設定方法は以下の通りです．

Illustrator ver.8 まず文字はアウトライン化します．ツールバーの[別名で保存]でファイル形式を ‘Illustrator EPS’ として保存します．EPS 形式での保存オプションで「サムネールを作成」の**チェックを外して**,「フォントデータを含む」のチェックをはずしてください．ポストスクリプトのレベルは ‘PostScript level 2’ としてください．プレビューについては ‘8-bit IBM PC’ で良いと思います．Illustrator の場合はサイズが自動的に調節されますので用紙サイズを設定する必要はありません．誤ってサムネールを付けた場合バージョンによってエラーになるので Tomas Rokicki 氏の fixill.pl を使うと良いでしょう．Perl スクリプトで書かれており，実行するためには Perl が必須で

```
fixill < input.eps > output.eps
```

のように使います．

photoshop ver.7 [ファイル], [複製を保存]を選び「保存形式」を ‘Photoshop EPS’ にして保存する．保存オプションで「エンコーディング」は ‘ASCII’ とする．ビットマップ画像はエンコーディングで圧縮しないほうが印刷品質が良いでしょう．

Mathematica ver.4 ツールバーから[ファイル]の[特殊な形式で保存]を選び[TeX(X)]を選びます． そうすると数式やグラフなどが自動的に LaTeX 2ε 形式に保存されます．またグラフは EPS 形式で `filename.eps` という名前で保存されます．Mathematica の場合出力される EPS 画像のバウンディングボックスが正常に出力されないことがあるので LaTeX で正しく処理できない場合があります．出力された `filename.eps` というファイルをテキストエディタで開けば

```
%%BoundingBox: 91.5625 3.1875 321.938 190
```

のような記述があります．これは画像を平面上のどこに配置するかを指定するもので，左から 2 次元平面上の始点の x_0 と y_0，終点の x と y に対応します．また，通常はこの値は整数値が推奨されます．上記の数値を四捨五入して整数に直して取り込んでください．

MATLAB グラフを表示している MATLAB プログラムのウィンドウのツールバーにある[ファイル]から[エクスポート]を選び，ファイルの種類を ‘EPS Level 2’ にし，任意の名前をつけて保存します．Illustrator 形式での出力もサポートされていますので，お持ちの場合はグラフを編集できるのではないでしょうか．

8.3.4 図を二つ横に並べる

2 段組の場合はそのようなことはありませんが，1 段組の場合は一つの図だけでは両脇が開いてしまうのでそこに二つの図を ‘(a)’ と ‘(b)’ として挿入したいときがあります．このようなときは `minipage` 環境を使います．以下のように入力する例もあります．

```
\begin{figure}[htbp]
  \begin{minipage}{.47\textwidth}
      \centering% ここに図 (a) を入れる
      (a) 初期値$c=0.6$
  \end{minipage}
```

```
\hfill
  \begin{minipage}{.47\textwidth}
      \centering% ここに図 (b) を入れる
      (b) 初期値$c=1.0$
  \end{minipage}
  \caption{1 段組で横に図を二つ並べる}
\end{figure}
```

両方の図の番号を別にしたいときも同様に記述します.

8.3.5　EPS 以外の画像の張り込み

LaTeX では EPS 以外の画像の張り込みも可能ですが少々癖があります．お手元には GIF や BMP，JPEG などのビットマップ画像があると思います．ビットマップ画像を EPS に変換できるツールは無料でたくさんあり，有名なのが ImageMagick です．Windows 専用ですが EPS-Conv

http://hp.vector.co.jp/authors/VA023018/

というソフトウェアを導入することにより，現存するほとんどの画像を EPS に変換可能です．これらのアプリケーションはあくまでビットマップを EPS 形式で保存するだけなのでファイルサイズが非常に大きくなります.

デバイスドライバとして Dvipdfm*x* を使うと EPS 形式以外の画像でも取り込むことが出来ます．変換しなくても取り込める形式は PDF，JPEG，PNG，MetaPost の 4 種類です．Windows の BMP などは低圧縮の JPEG などに変換します．`image.png` があった場合端末などで

```
▉ ebb image.png
```

とすれば `image.png` 用の `image.bb` が作成されます．ebb は dvipdfm に同封されています．この `image.bb` は画像ファイルの縦横を正しく扱うためのファイルです．`image.bb` を見れば分かりますが，中身は

```
%%Title: ./image.png
%%Creator: ebb Version 0.5.2
%%BoundingBox: 0 0 595 841
```

となっています．‘BoundingBox’ とは原点座標と画像の縦横の長さの値です．次にソースファイルを以下のようにします.

```
\documentclass{jsarticle}
\usepackage[dvipdfm]{graphicx}
\begin{document}
\includegraphics[width=3cm]{image.png}
\end{document}
```

後はいつも通りにタイプセットして DVI ファイルを生成し Dvipdfm*x* で PDF を作成すれば良いことになります.

8.3.6 EPS から PDF への変換

デバイスドライバとして Dvipdfm*x* で手持ちの EPS 画像を取り込むようにしている場合，Dvipdfm*x* を実行するたびに毎回 Ghostscript を使って EPS から PDF に変換する処理が行われます．処理時間の短縮のために，あらかじめ PDF に変換しておけば良いでしょう．EPS 形式の画像を PDF へ変換する方法はいくつかあるそうですが，簡単で安全な方法を紹介します．まず Ghostscript の 'pdfwrite' の力を借りる方法です．BoundingBox の設定されている EPS 画像を epstopdf によって PDF に変換します.

```
▌epstopdf filename.eps
```

とすると filename.pdf が作成されます．日本語が使われている EPS がうまく処理できないなどの問題があるかもしれません．その場合は GNU の Ghostscript のバージョン 7.07 を使うと良いでしょう．以下のようなシェルスクリプト epspdf

```
#!/bin/bash
FILE=$1
fig=‘basename $FILE .eps‘
epstopdf $FILE
egrep "%%BoundingBox:" $FILE > $fig.bb
```

を作成し/usr/local/bin/などに複製して

```
▌epspdf filename.eps
```

とすると PDF ファイル filename.pdf とバウンディングボックス情報 filename.bb が作成されます．同じディレクトリの中の全ての EPS 画像を PDF に変換したいときは

```
#!/bin/sh
for f in ‘ls *.eps‘; do
   fig=‘basename $f .eps‘
   epstopdf $f
   grep "^%%BoundingBox:" $f > $fig.bb
done
```

を epspdfs という名前で同じように複製します．このようにして処理した PDF は LaTeX の原稿で

```
\documentclass{jsarticle}
\usepackage[dvipdfm]{graphicx}
\begin{document}
\includegraphics{filename.pdf}
\end{document}
```

のようにして取り込むことができます.

8.4　描画の方法

LaTeX で図を取り扱う手段はいくつも存在します．写真のような画像を graphicx パッケージなどを使って張り込む方法と，1 から描画する方法です．graphicx パッケージを用いて既存の画像を張り込む方法は 8.3 節を参照してください．画像をまだ作成していない段階での描画の方法を紹介します．

描画の方法は大きく分けて二つあります．一つは LaTeX 自身の能力で描画する方法と \special 命令を使い他のプログラムへ描画をゆだねる方法です．一般に LaTeX における描画の能力は TeX 譲りのシステムのお陰で貧弱なものとなっています．簡単な図を作成するならば LaTeX に備わっている picture 環境による描画を行うのが手軽です．

8.4.1　べた書きによる図の作成

もっとも簡単な描画の方法として LaTeX でべた書きを行う，verbatim 環境を使うことが考えられます．verbatim 環境内では文字が**等幅**に近い字詰めで組まれるので，原稿で入力している表示と DVI ファイルへの出力が同じようになります．ものは試しですのでやってみてください．

```
\begin{verbatim}
   ∧__∧    ∧__∧                ∧__∧    ∧__∧
  (´∀`)(´∀`)              (´∀`)(´∀`)
 ⊂    つ ⊂    つ            ⊂    つ ⊂    つ
  | |   |   |   | |            | |   |   |   | |
  (_) _) (_  (_)              (_) _) (_  (_)
\end{verbatim}
```

verbatim 環境内では半角の空白を使わずに全角の空白を使うと良いでしょう．半角の空白はテキストエディッタで入力している分の空白が挿入されるわけではありませんので．

内田昭宏氏の作成した plain2 というツールを使うと全角記号を組み合わせることによって LaTeX 用の図表を作成することもできます．

8.4.2　曲線の描画

ベクトル画像などでは**ベジェ曲線**とか**スプライン曲線**という**近似曲線**が使われていると多くの参考書で記述されています．ベクトル画像を知るうえでベジェ曲線の原理を知っておくと，曲線を描くときに頭の中で曲線をイメージしやすいと思いますので紹介しておきます．滑らかな曲線を描くためには多くの点座標が必要になると思う人もいるでしょうが，ある程度滑らかな曲線を描くためには 3 点あれば十分です．曲線を描くための点（制御点）が少なければ少ないほど情報量は少なくなるので，少ない制御点で滑らかな曲線を描く方法が過去に模索されました．その中でもベジェ曲線は高々二つの基準点と一つの制御点（2 + 1 点）があれば現在私たちが Illustrator などでよく見かける曲線になります．こ

の原理が Illustrator のペンツールに活かされていますので，お持ちの方は確認していただくと良いでしょう．ただ Illustrator の場合はユーザの見えない箇所で様々な工夫がなされています．

曲線を描くためにいま n 個の制御点がありその i 番目の座標を $P_i = (x_i, y_i)$ として式 (8.1) と式 (8.2) で表す曲線をベジェ曲線と呼びます．

$$P(u) = \sum_{i=0}^{n-1} P_i \, B_{i,n}(u) \tag{8.1}$$

$$B(u) = \frac{n!}{i!\,(n-i)!} u^i (1-u)^{n-i} \tag{8.2}$$

これがベジェ曲線の一般形ですが，例として Type1 フォントでも使われている？ 2 次ベジェ曲線を示します．平面座標に $P_0 = (-1, 1)$，$P_1 = (0, 0)$，$P_2 = (1, 1)$ があるとすれば式 (8.1) と式 (8.2) より

$$\begin{aligned} P(u) &= P_0 B_{0,2} + P_1 B_{1,2} + P_1 B_{1,2} \\ &= P_0 (1-u)^2 + P_1 2u(1-u) + P_2 u^2 \end{aligned} \tag{8.3}$$

となり式 (8.3) に対して無数の u を与えれば滑らかな曲線を描けます．これは 3 次元でも同様に計算できるので便利な式です．例の基準点，制御点とベジェ曲線は図 8.1 の通りになります．このような原理を知っておくと後ほど紹介する LaTeX の picture 環境で使用

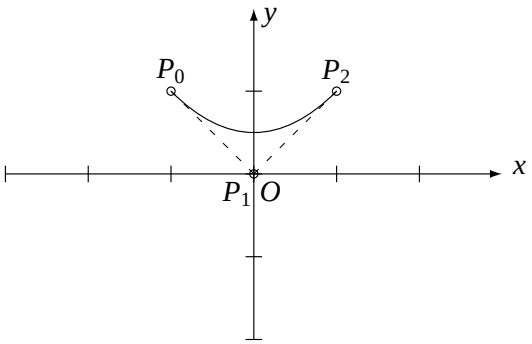


図 8.1 制御点と式から得られるベジェ曲線

できる命令の理解に役立つことでしょう．ただし LaTeX での多くのベジェ曲線を描くコマンドはもっと計算の少ないアルゴリズムを使っている場合がありますし，デバイスドライバに描画を任せていることもあります．

8.4.3 picture 環境による描画

LaTeX の力を使った描画を行うには特別な環境，描画専用の picture 環境で作業を使います．picture 環境では基準となる長さを決めてその相対的な距離によって描画を行います．このとき基準となる長さ\unitlength を決めます．

\begin{picture}(x, y)(x_0, y_0)
描画内容
\end{picture}

picture 環境の中に描画したい内容を記述します．picture 環境に渡す ‘(x, y)’ は必須引数ですが ‘(x_0, y_0)’ は任意引数です．‘(x, y)’ には座標における picture 環境の大きさを横方向は x で縦方向は y で指定します．これには単位などを付けずに数値で指定します．‘(x_0, y_0)’ には原点の位置を指定します．

何らかの要素を配置するには\put か\multiput を使います．x と y は単位 \unitlength に従属します．

\put(x, y){⟨要素⟩}
\multiput(x, y)(Δx, Δy){⟨回数⟩}{⟨要素⟩}

\put 命令は座標 (x, y) に ⟨要素⟩ を置くだけの命令です．\multiput は座標 (x, y) を基点とし，二つ目の座標 (Δx, Δy) をベクトルとして (Δx, Δy) の変化量に応じて要素を回数分だけ繰り返して配置します．この他に 2 次ベジェ曲線を描く\qbezier 命令があります．

\qbezier(x_1, y_1)(x_2, y_2)(x_3, y_3)

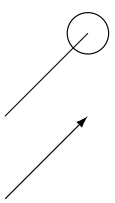
‘(x_1, y_1)’ を始点，‘(x_2, y_2)’ を基準点，‘(x_3, y_3)’ 終点として 2 次ベジェ曲線を描きます．

⟨要素⟩ には次のようなコマンドが標準として使えます．

\line(x, y){⟨長さ⟩}
\vector(x, y){⟨長さ⟩}
\circle*{⟨直径⟩}
\oval(幅, 高さ)[⟨位置指定⟩]

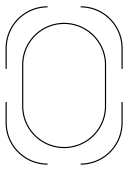
\line は ‘(x, y)’ をベクトルとして ⟨長さ⟩ 分の線分を描きます．\vector は\line の終端に矢印をつけたものです．\circle*は直径を指定して円を描きます．アスタリスク ‘*’ を付けないと円の内側が塗りつぶされません．\oval は幅と高さを指定して楕円を描きます．

```
\setlength{\unitlength}{1mm}
\begin{picture}(40,30)
\put(10,10){\line(1,1){10}}
\put(10,0){\vector(1,1){10}}
\put(20,20){\circle{5}}
\end{picture}
```



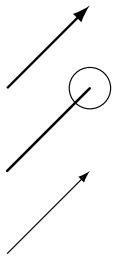
楕円を描く\oval 命令の任意引数の ⟨位置指定⟩ には楕円のどの部分を出力するかを指定します．それぞれ上部 ‘t’，下部 ‘b’，左 ‘l’，右 ‘r’ となり，複合的に使用できます．

```
\setlength{\unitlength}{1mm}
\begin{picture}(50,30)
\put(8,12){\oval(10,15)[tl]}
\put(8,8){\oval(10,15)[bl]}
\put(10,10){\oval(10,15)}
\put(12,12){\oval(10,15)[tr]}
\put(12,8){\oval(10,15)[br]}
\end{picture}
```



picture 環境中での線の太さは\thinlines と\thicklines の二つで調整します．\thinilnes のほうが細く\thicklines のほうが太くなります．picture 環境中の全ての線分に有効になります．

```
\setlength{\unitlength}{1mm}
\begin{picture}(50,30)
\thicklines
\put(10,10){\line(1,1){10}}
\put(10,20){\vector(1,1){10}}
\thinlines
\put(10,0){\vector(1,1){10}}
\put(20,20){\circle{5}}
\end{picture}
```



8.4.4 picture 環境の拡張その 1——epic

LaTeX での標準の picture 環境のコマンドもデバイスに依存しないので汎用性があるのですが，それではあまりにも表現力に乏しいのが現状です．そこでこの picture 環境の拡張が行われてきました．picture 環境の限らず LaTeX での描画は 1980 年代後半からさまざまな方法が模索され，拡張され続けました．その中でも Sunil Podar 氏による epic は picture 環境の拡張版としては定評があります．epic では LaTeX の picture 環境で使用できるコマンドのほかに以下の命令が拡張されています．

\multiputlist \matrixput \grid \picsquare \dottedline
\dashline \drawline \jput \putfile

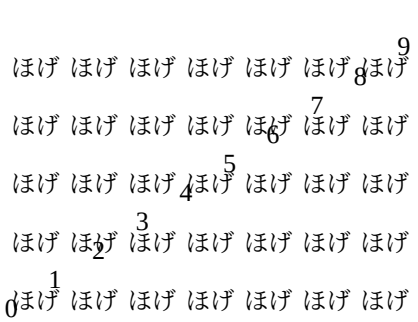
このほかに dottedjoin，dashjoin，drawjoin の三つの環境が定義されています．座標の変化量を $(\Delta x, \Delta y)$ として複数の項目を配置する\multiputlist 命令があります．

\multiputlist(x, y)(Δx, Δy)[⟨tbrl⟩]{⟨複数の項目⟩}

座標上に行列のように要素を繰り返して配置する\matrixput 命令もあります．

\matrixput(x, y)(Δx_1, Δy_1){⟨n_1⟩}(Δx_2, Δy_2){⟨n_2⟩}{⟨要素⟩}

```
\setlength{\unitlength}{1pt}
\begin{picture}(150,110)(0,0)
\multiputlist(0,0)(15,10)%
  {0,1,2,3,4,5,6,7,8,9,10}
\matrixput(0,0)(20,0){7}(0,20){5}%%
  {\mbox{ほげ}}
\end{picture}
```



座標系を表現するのに格子を描くには\grid 命令が使えます.

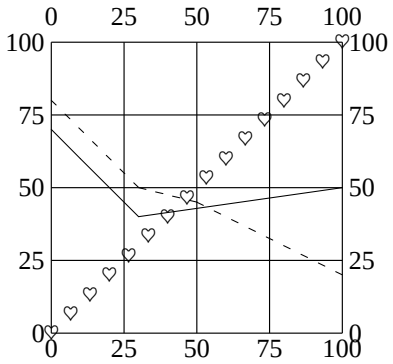
\grid(幅, 高さ)(Δ幅, Δ高さ)[⟨x 座標の初期値,y 座標の初期値⟩]

他にも点線や破線などを描くコマンドがあります.

\dottedline[⟨点の種類⟩]{⟨間隔⟩}$(x_1, y_1)(x_2, y_2)\cdots(x_n, y_n)$
\dashline{⟨破線の長さ⟩}[⟨間隔⟩]$(x_1, y_1)(x_2, y_2)\cdots(x_n, y_n)$
\drawline$(x_1, y_1)(x_2, y_2)\cdots(x_n, y_n)$

\dottedline は点線を，\dashline は破線を，\drawline は折れ線を描くために使います．点線や破線は折れても構いません.

```
\setlength{\unitlength}{1pt}
\begin{picture}(150,120)(0,0)
\put(0,0){\grid(100,100)(25,25)[0,0]}
\dottedline[$\heartsuit$]{10}(0,0)(100,100)
\dashline{3}(0,80)(30,50)(50,45)(100,20)
\drawline(0,70)(10,60)(30,40)(100,50)
\end{picture}
```



8.4.5　picture 環境の拡張その 2——eepic

Sunil Podar 氏による epic を改良・拡張した Conrad Kwok 氏の作成した eepic があります．これは epic の改良・拡張版でありますので使用するときは

```
\usepackage{epic,eepic}
```

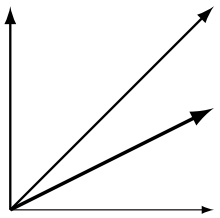
として epic も先に読み込んでおきます．LaTeX の picture 環境で使用できる\line, \circle*, \oval の拡張が行われています．epic のコマンドも全て再定義されています．eepic はこれらの命令を Tpic スペシャルに置き換えていますので描画力は高いのですが，デバイスドライバが Tpic スペシャルに対応している必要があります．dviout，dvips, Dvipdfm*x* などは対応しているようです．デバイスドライバによって Tpic スペシャルの解釈が若干異なるようですので，出力を確認してデバイスを選択してください.

線の太さに関するコマンドが新たに定義されています.

\allinethickness{⟨太さ⟩} \Thicklines

\allinethickness はこの命令を使った後の picture 環境中にある全ての線の太さを変更します．\Thicklines は\thicklines よりもこのコマンドを宣言した後の線の太さを太くします．

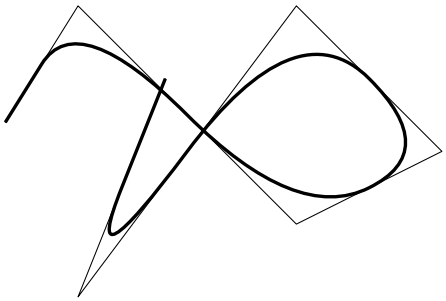
```
\begin{picture}(100,100)(0,0)
\thinlines \put(0,0){\vector(1,0){70}}
\thicklines\put(0,0){\vector(1,1){70}}
           \put(0,0){\vector(0,1){70}}
\Thicklines\put(0,0){\vector(2,1){70}}
\end{picture}
```



eepic では\drawline よりも\path を使い，\qbezier よりも\spline を使うと良いでしょう．\spline は始点と終点以外は制御点として Chaikin 曲線を描きます．

\path(x_1, y_1)(x_2, y_2)$\cdots$(x_n, y_n) \spline(x_1, y_1)(x_2, y_2)$\cdots$(x_n, y_n)

```
\setlength{\unitlength}{1pt}
\begin{picture}(150,100)(0,0)
\path(0,60)(25,100)(100,25)(150,50)%
     (100,100)(25,0)(55,75)
\Thicklines
\spline(0,60)(25,100)(100,25)(150,50)%
       (100,100)(25,0)(55,75)
\end{picture}
```



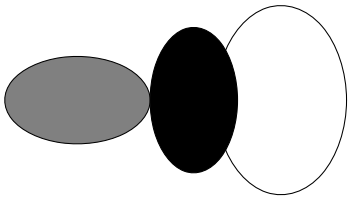
卵形の楕円を描くのに\ellipse を，弧を描くには\arc を使います．アスタリスクを付けると領域を塗りつぶします．

\ellipse*{⟨幅⟩}{⟨高さ⟩} \arc{⟨長さ⟩}{⟨始点角度⟩}{⟨終点角度⟩}

⟨始点角度⟩ の値は $[0, \pi/2]$ の範囲に，⟨終点角度⟩ の値は [始点角度, 始点角度 + 2π] の範囲にします．領域の塗りつぶしには\filltype を使います．アスタリスクを付けた場合の\circle*と\ellipse*の領域の塗りつぶす種類を ‘black’，‘white’，‘shade’ の三つから選択します．

\filltype{⟨種類⟩}

```
\begin{picture}(150,100)(0,0)
\filltype{shade}
\put(10,50){\ellipse*{50}{30}}
\filltype{black}
\put(50,50){\ellipse*{30}{50}}
\put(80,50){\ellipse{45}{65}}
\end{picture}
```



8.4.6　picture 環境の拡張その 3——pict2e

Hubert Gäßlein 氏と Rolf Niepraschk 氏による pict2e は picture 環境の拡張として 2003 年頃に公表されたものです．デバイスドライバの力を借りて picture 環境で使用できるコマンドを再定義していますし，新しいコマンドも定義されています．今のところ

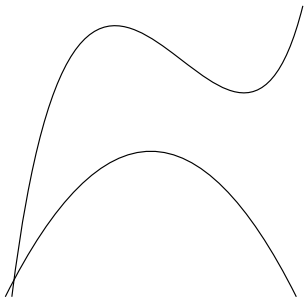
dvips xdvi pdftex vtex dvipdfm

などのデバイスドライバをサポートしています．picture 環境におけるほとんどのコマンドを拡張してあります．\circle で描ける円の大きさにも制限はありません．ベジェ曲線に関しては\cbezier 命令が追加されました．

\qbezier(x, y)(x, y)(x, y)
\cbezier(x, y)(x, y)(x, y)(x, y)

\qbezier は 2 次ベジェ曲線用，\cbezier は 3 次ベジェ曲線用の命令です．

```
\setlength{\unitlength}{1pt}
\begin{picture}(100,100)(0,0)
\qbezier(0,0)(50,100)(100,0)
\cbezier(0,0)(25,200)(75,0)(100,100)
\end{picture}
```



8.5　他のプログラムによる描画

広く使われている描画ツールを紹介します．以下の描画ツールで作成した図などはそれぞれ指定された方法でデバイスドライバが適切に解釈できる状態になければなりません．

8.5.1　Tpic による描画

Unix 系 OS の描画ツールとして Brian Kernighan 氏が開発した PIC があります．これを Tim Morgan 氏が TeX に移植した Tpic を使うと簡単な図形ならばすぐに描くことができます．Unix 系 OS を持っているならば PIC というプログラム導入されているでしょう．ただ PIC は日本語化されていないかもしれないので注意してください．Tpic はほとんど

の機能を TeX ではなくデバイスドライバに任せています．これらの命令は\special 命令の中に記述されています．Tpic の出力する特有な命令を Tpic スペシャルと言います．Tpic スペシャルに対応しているドライバは dvips や dviout，Dvipdfm*x* などです．多くのドライバが対応していますが，デバイスドライバによる解釈の違いなどもありますので若干注意が必要でしょう．

例題程度に紹介しておきます．例えば図 8.2 のような図を作成したいとしましょう．PIC の入力ファイル hoge.pic に以下のような記述をします．

図 8.2　Tpic の使用例

```
.PS
 box "PIC file"
 arrow "pic" above
 box "\textsc{Tpic} file"
 arrow "input" above dashed
 box "{\LaTeX} file"
 arrow "{\LaTeX}" above
 box "DVI"
 arrow "special" above dashed
 ellipse "Picture" fill
.PE
```

この hoge.pic を pic で TeX 用に出力するために-t オプションを付けて

```
pic -t hoge.pic > hoge.tex
```

とすることで hoge.tex 中の\graph に図形が格納されます．これを file.tex で使用するためには

```
\input{hoge}
\begin{center}
{\box\graph}
\end{center}
```

とするとその場所に中央揃えでグラフを挿入できます．適宜 figure 環境に入れるなどします．

8.5.2　メタな描画プログラム

Donald Knuth 氏がフォントデザイン用に開発した METAFONT があります．これに対して John Hobby 氏が描画に関するアルゴリズムを追加したり，出力形式を EPS にした METAPOST という描画プログラムを開発しました．METAFONT は TeX のフォント形

式ファイル ⟨*file*⟩.gf を出力するのに対して METAPOST は EPS 形式ファイル ⟨*file*⟩.n を出力するので今では METAPOST が広く使われいます．METAPOST に関する日本語情報は少ないのが現状です．しかし METAFONT から変更・追加された箇所があったとしても，Donald Knuth 氏の *METAFONTbook* [48] が参考になると思います．

METAFONT をちょっと触ってみましょう．端末から

```
mf
```

とすると

```
This is METAFONT, Version 2.7182 (Web2C 7.3.9)
**
```

のようにアスタリスク ‘*’ が二つ表示されてファイルの入力を促しています．ここでは実験的に，“\relax” と入力して改行します．するとアスタリスクが一つなるはずです．

```
**
```

これで準備は万全です．とりあえず点を表示してみましょう．

```
* drawdot (0,0); showit; <ENTER>
```

今度は直線を描くために

```
* draw (0,0)..(100,0); showit;
```

としてみましょう．曲線などは

```
* draw (0,100)..(100,100)..(100,0); showit; <ENTER>
```

とすると雰囲気がつかめるでしょう．終了するときは

```
end.
```

と入力します．

今度は METAPOST を少し体験してみましょう．‘mp’ か ‘mpost’ を端末から実行すれば良いはずです．今度は METAPOST のファイル hoge.mp を用意します．

```
beginfig(1)
u=100;
draw (u,u)--(2u,u)--(2u,2u)..cycle;
draw (u,u)..(2u,u)..(2u,2u)..(u,2u)..cycle;
endfig;
end.
```

これを見ても何がなんだかわかりませんが，とりあえず保存しておきます．端末から

```
mpost hoge.mp
```

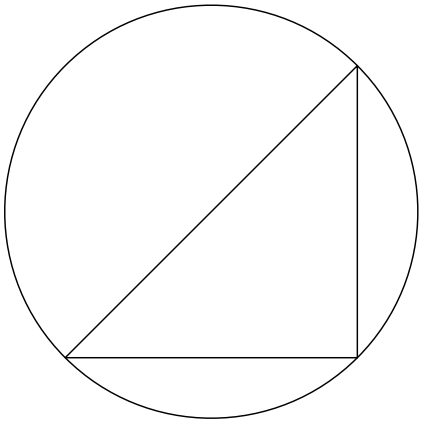


図 8.3　METAPOST の出力例

とします．そうすると EPS 形式の hoge.1 が作成されますのでご覧ください．Ghostscript などで見ると図 8.3 のような円とその円に内接する直角二等辺三角形が表示されます．環境によって日本語化された METAPOST を使うためには mpost ではなく jmpost を使うことになるかもしれません．角藤亮氏の pTeX を使っている Windows の方は jmpost のはずです．

8.5.3　PSTricks

Timothy Zandt 氏らによる PSTricks は\special 命令中に PostScript 命令を記述することによって PostScript の描画機能を TeX/LaTeX で使えるようにするパッケージです．PostScript 命令を多用することからデバイスドライバとして dvips などを想定していたり PostScript 対応プリンタでの使用が推奨されます．PSTricks の詳しい使い方は日本語訳で 70 ページ分の『LaTeX グラフィックスコンパニオン』 [30] の第 4 章を参照してください．基本的なマクロを読み込むためには pstricks パッケージを読み込みます．特定のパッケージを個別に読み込むこともできます．色を使うためには grahics パッケージに含まれる color パッケージではなく pst-col パッケージを読み込みます．全ての機能を使うときは pst-all を読み込みます．標準的に以下のようにするとうまく行くと思います．

```
\usepackage[dvips]{graphicx}
\usepackage[dvips,usenames]{pst-col}
\usepackage{pst-all}
```

具体的に 3 次元射影，画像の EPS 変換，グラデーション，木構造，回路図，プロットなどさまざまなことができます．口絵 II に状態遷移図の例を示します．

PSTricks を Dvipdfm*x* で使うにはまず面倒な方法として pstricks を使って描いた図形を一つ一つ EPS ファイルに変換する方法です．これは

```
\documentclass{jsarticle}
\usepackage[dvips]{graphicx}
```

```
\usepackage[dvips,usenames]{pst-col}
\usepackage{pst-all}
\pagestyle{empty}
\begin{document}
\begin{pspicture}(100,100)
 描画内容
\end{pspicture}
\end{document}
```

のようなファイル fig1.tex を作成し，これを端末などから

```
▉ platex fig1
▉ dvipsk -Pdl -E -o fig1.eps fig1
▉ epstopdf fig1.eps
▉ egrep "^
```

ということを図形の数だけこなし，図形を使用したい LaTeX の原稿に PDF ファイル fig1.pdf として graphicx の\includegraphics で張り込みます.

```
%\usepackage[dvipdfm]{graphicx}
\includegrpahics[bb={0 0 100 100}]{fig1.pdf}
```

CV Radhakrishnan 氏と CV Rajagopal 氏らによる pdftricks を使うと少しは楽になります．これはもともと pdfTeX で PSTricks を使うためのものですが，pLaTeX にも流用できそうです．まずは pdftricks に付属の pst2pdf というシェルスクリプトを少し変更します.

```
for f in $FIGURES ; do
  latex $fig
  dvips -Ppdf -E -o $fig.eps $fig
  epstopdf $fig.eps
done
```

という部分で ‘latex’ はもちろん ‘platex’ にしますし．‘dvips’ もご自分のシステムに合うように変更してください．epstopdf で変換した PDF のバウンディングボックスは結構まともななんですが

```
egrep "^%%BoundingBox:"      $FILE > $fig.bb
```

の 1 行も追加しておくと良いでしょう.

第 9 章

LaTeX の応用

以下に示すコマンドなどはレポートや論文には必要不可欠という程の要素ではありませんので，このような機能もあるという程度でご覧ください．

9.1 ページレイアウトの簡単な設定

9.1.1 版面のレイアウト

版面のレイアウトを行う場合にはそれぞれの長さに対して直接値を代入する方法があります．LaTeX で一般的に設定できる版面を調節する長さは図 9.1 の通りです．このような版面を視覚的に確認するには layout パッケージが使えます．このパッケージは使用されているクラスファイルから版面のレイアウトを出力します．使用方法は document 環境中で\layout 命令を使うだけです．

まずはページ全体の余白に関する長さです．

\voffset　横組みにおいて用紙の左上の部分に入れる縦方向の余白．この値を 0 にしてもすでに 1 インチ分の余白が挿入される．本当に用紙の左上端から使うならば \voffset を ‘-1in’ に設定します．

\hoffset　横組みにおいて用紙の左上の部分に入れる横方向の余白．縦方向と同じようにすでに 1 インチ分の余白が挿入されています．

\oddsidemargin　ページが奇数のときに挿入される左側の余白．文書クラスオプションに *oneside* を使っていると全てのページに\oddsidemargin が挿入されます．

\evensidemargin　ページが偶数のときに挿入される左側の余白．文書クラスオプションに *twoside* を使っているときだけ有効で *oneside* では意味がありません．

ヘッダの設定に関する長さです．

\topmargin　\voffset とヘッダの間隔です．

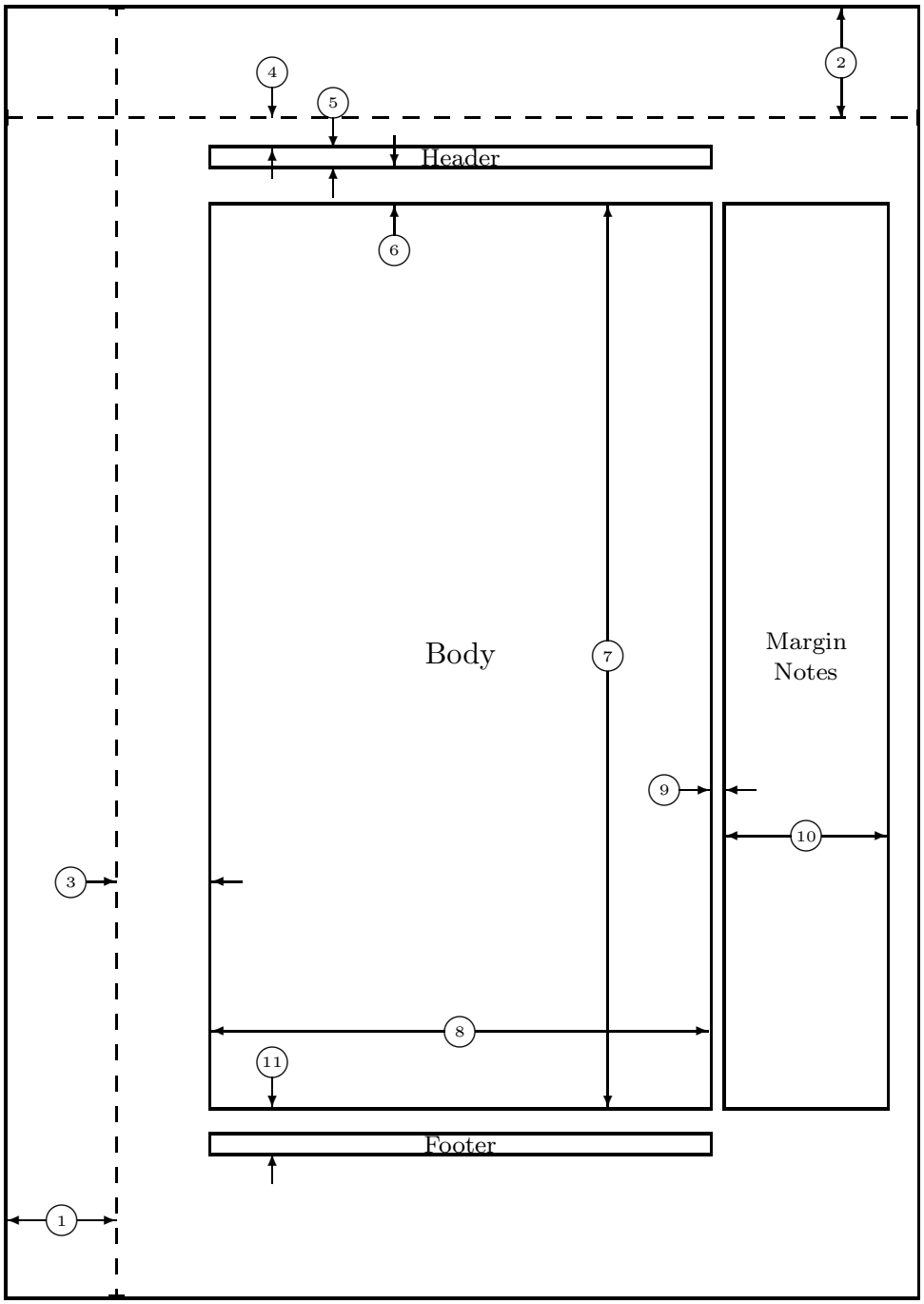
\headheight　ヘッダの高さです．

\headsep　ヘッダと本文領域の間隔です．

\footskip　フッタ下部と本文領域の最下部との間隔です．

本文領域や傍注領域に関わる長さです．

\textheight　本文領域の高さです．ヘッダやフッタの高さは含まれません．




```
 1   one inch + \hoffset          2   one inch + \voffset
 3   \oddsidemargin = 62pt        4   \topmargin = 20pt
 5   \headheight = 12pt           6   \headsep = 25pt
 7   \textheight = 592pt          8   \textwidth = 327pt
 9   \marginparsep = 10pt        10   \marginparwidth = 106pt
11   \footskip = 30pt                 \marginparpush = 5pt (not shown)
     \hoffset = 0pt                   \voffset = 0pt
     \paperwidth = 597pt              \paperheight = 845pt
```

図 9.1　版面のレイアウトに使用できる長さ

\textwidth　本文領域の幅です.
\marginparwidth　傍注の幅です.
\marginparpush　傍注と傍注のあいだの縦方向の長さです.
\marginparsep　傍注と本文領域との間隔です.
\columnsep　2 段組以上での段と段の間隔です.
\columnseprule　2 段組以上での段と段のあいだに入る罫線です.

通常ここで紹介した長さはクラスファイル側でフォントサイズやクラスオプションに応じて適切に設定されますので悪戯に変更しないでください. 相手先の都合で「1 行何文字 1 ページ何行」のような設定などをしなければならないときは無理やり

```
\setlength{\textwidth}{33zw}
\setlength{\textheight}{40\baselineskip}
```

とすることもできます.

もっと簡単に版面の設定をしたいならば梅木秀雄氏の作成した geometry を使うのが良いでしょう.

ある環境などにおけるその時々の文章幅を保持している\linewidth という長さがあります. この長さを使うとその環境において文章幅いっぱいの図を張り込むということもできるようになります.

```
\begin{quote}
 linewidth=\the\linewidth
 \begin{quote}
  linewidth=\the\linewidth
 \end{quote}
\end{quote}
```

```
linewidth=176.52332pt
    linewidth=158.02957pt
```

9.1.2　ヘッダやフッタの設定その 1

ヘッダやフッタなどに出力されるページ番号などを変更したいときがあると思います.

\pagestyle{⟨表示形式⟩}

\pagestyle 命令を記述したページから指定した表示形式に変更されます. 指定できる形式は表 9.1 の通りです. 'myheadings' では

表 9.1　ヘッダやフッタの指定

命令	内容
empty	ページ番号を表示しない
plain	フッタ中央部に表示する
headings	ヘッダにページ番号と章・節名を表示する
myheadings	ユーザー定義の表示形式にする

\markright {⟨ヘッダ⟩}
\markboth {⟨偶数ヘッダ⟩}{⟨奇数ヘッダ⟩}

の二つの命令によってヘッダの出力を指定します．片面印刷のときに\markright を使います．両面印刷には\markboth を使います．2004 年度版の「ほげほげ大学」の論文集を作成しているのであれば

```
\pagestyle{myheadings}
\markboth{ほげほげ大学論文集}{2004 年度版}
```

とすると良いでしょう．

任意の 1 ページだけのヘッダ・フッタは

\thispagestyle{⟨表示形式⟩}

とすることでそのページだけ変えることができます．

ページ番号の表示形式を変更するには

\pagenumbering{⟨表示形式⟩}

と記述すればその場所から指定した形式で 1 ページ目からカウントしてページ番号を表示します．指定できる表示形式は表 9.2 の通りです．ここで注意することはアルファベットにした場合は，最大 26 ページまでしかカウントできないということです．27 以上になった場合の対策は別にすることになります．

表 9.2　ページ番号の種類の指定

命令	内容	出力例
arabic	アラビア数字	1, 2, 3, . . .
roan	ローマ数字	i, ii, iii, . . .
Roman	ローマ数字	I, II, III, . . .
alph	アルファベット小文字	a, b, c, . . . , z
Alph	アルファベット大文字	A, B, C, . . . , Z

9.1.3　ヘッダ・フッタの変更その 2

LaTeX が標準で用意してくれているページスタイルでは寂しい，そう思う人も多いでしょう．自分で全て定義することもできますが，Piet Oostrum 氏が作成した fancyhdr を使うと比較的簡単にページスタイルをカスタマイズできます．fancyhdr は fancyheadings の後継で，ページのヘッダーとフッターをカスタマイズできるマクロです．まず fancyhdr を使うために

```
\usepackage{fancyhdr}
\pagestyle{fancy}
```

をプリアンブルに記述します．fancyhdr ではヘッダ・フッタを六つに分割しています．

\lhead	\chead	\rhead
（\headrulewidth）		
本文領域		
（\footrulewidth）		
\lfoot	\cfoot	\rfoot

\lhead, \chead, \rhead の三つはヘッダに使い\lfoot, \cfoot, \rfoot の三つはフッタに使います．\headrulewidth はヘッダ下部の罫線の太さで，\footrulewidth はフッタ上部の罫線の太さです．図だけのページや表だけのページはシンプルなヘッダ・フッタにしたり，ヘッダ下部の罫線を引かない場合があります．その場合は

```
\def\headrulewidth{\iffloatpage{0pt}{.4pt}}
```

と定義すると良いでしょう．\lhead などの他の命令も同様に変更できます．

例えば jarticle クラスで次のようなヘッダ・フッタ

		資本主義社会の崩壊
本文領域		
名無しの権兵衛		**ページ番号**

を設定したければ以下のように記述します．

```
\lhead{} \chead{}
\rhead{資本主義社会の崩壊}
\lfoot{名無しの権兵衛}
\cfoot{}
\rfoot{\textbf{\thepage}}
\def\headrulewidth{.4pt}
\def\footrulewidth{.4pt}
```

書籍用クラス jbook などでクラスオプションに *twoside* が指定されている場合は偶数ページと奇数ページを個々に設定します．例えば次のようなヘッダ・フッタにしたいとします．

第 1 章　ほげ	1.1　ほげ
本文	
2	

1.2　ほげ	第 1 章　ほげ
本文	
3	

ヘッダ・フッタの奇数ページと偶数ページの場所を区別するために以下のような設定になっています．

EL(H)	EC(H)	ER(H)
本文		
EL(F)	EC(F)	ER(F)

偶数ページ側

OL(H)	OC(H)	OR(H)
本文		
OL(F)	OC(F)	OR(F)

奇数ページ側

先程の出力を得るためには\fancyhead と\fancyfoot 命令を使います．

```
\documentclass{jbook}
\usepackage{fancyhdr} \pagestyle{fancy}
\fancyhead[ER,OL]{\rightmark}
\fancyhead[EL,OR]{\leftmark}
\fancyfoot[EC,OC]{\textbf{\thepage}}
```

欧文のクラスではこれで良いのですが和文では\chaptermark と\sectionmark の定義を fancyhdr を読み込んだ後に次のように定義するのが普通でしょう.

```
\def\chaptermark#1{\markboth{%
  \ifnum \c@secnumdepth >\m@ne
    \if@mainmatter
      \@chapapp\thechapter\@chappos\hskip1zw
    \fi
  \fi
  #1}{}}%
 \def\sectionmark#1{\markright{%
   \ifnum \c@secnumdepth >\z@ \thesection \hskip1zw\fi
   #1}}%
```

9.1.4　ページ/総ページ

ヘッダ・フッタの設定をフッタだけに「ページ/総ページ」にしたい場合があるでしょう．これは例えば次のようにプリアンブルに記述します.

```
\AtEndDocument{\label{lastpage}}
\makeatletter
 \newcommand{\ps@total}{%
 \let\@mkboth\@gobbletwo
 \let\@oddhead\@empty
 \let\@evenhead\@empty
 \def\@oddfoot{\normalfont\hfil--\thepage/\pageref{lastpage}--\hfil}%
 \let\@evenfoot\@oddfoot}
\makeatother
\pagestyle{total}
```

この場合は\ps@total によって新規に 'total' というページスタイルを定義しています. ページ番号の書体を変えたいときは\normalfont を\bfseries などに変更します.

9.2　レイアウトの制御

LaTeX ではユーザーが意図的に改行や改ページを行わなくても良いように工夫されています．どうしても自分の思い通りにページをレイアウトできないときは強制的なレイアウト命令を使います．**ページ区切り**を制御したいならば

\newpage　改ページします．2 段組の場合は次の段までの改ページになります.

\clearpage　未出力の浮動体を配置してから改ページします．2 段組の場合は本当の次のページまで改ページされます．

\cleardoublepage　次のページが奇数ページになるように改ページします．これを**奇数起こし**とか**改丁**と呼びます．

\samepage　指定した場所でできる限り改ページを抑制します．

の四つの命令が使えます．

空白を制御するには以下の四つの命令が使えます．

\hspace{⟨長さ⟩}　長さ分の横方向の空白を挿入します．行頭では有効ではありません．

\hspace*{⟨長さ⟩}　行頭でも横方向の空白を挿入します．

\vspace{⟨長さ⟩}　長さ分の縦方向の空白を挿入します．ページの先頭・末尾では有効ではありません．

\vspace*{⟨長さ⟩}　ページの先頭・末尾でも縦方向の空白を挿入します．

これらの空白制御の命令では単位付きの長さで指定します．

```
\hspace{1cm}空白制御用のコマンドは行頭
では意図的に\vspace{1cm}アスタリスクを付けます．
\par
\hspace{1cm}段落の途中に縦方向\hspace{1cm}
の空白を挿入すると，段が改行されてから
縦に空白が挿入されます．
```

空白制御用のコマンドは行頭では意図的にアスタリスクを付けます．

段落の途中に縦方向　　の空白を挿入すると，段が改行されてから縦に空白が挿入されます．

9.3　その他のコマンド

LaTeX で用意されているその他のコマンドを紹介します．

9.3.1　日付

LaTeX のプログラムを実行した段階で，その原稿をタイプセットした日付を保存しています．

\today（日付）
\number\year（年）
\number\month（月）
\number\day（日）

使用しているクラスファイルによって出力が違います．jarticle などでは**\西暦**や**\和暦**という命令を使って\today の西暦表示と和暦表示を変更できます．奥村晴彦氏の jsclasses で標準は西暦になっており，アスキーの jclasses では和暦が標準です（個人的には天皇制の名残のような和暦を使うのは好ましくないと感じていますし，ビジネス文書でわざわざ和暦を使っても世界に置いてけぼりを食らうだけだと思っています，個人的に）．欧文のクラスファイルではその言語の標準的な表示方法で出力されます．

```
今日は{\number\year}年{\number\month}月
{\number\day}日です．略して{\today}．
```

今日は 2005 年 3 月 20 日です．略して 2005 年 3 月 20 日．

9.3.2　LaTeX のロゴ

\TeX	TeX
\LaTeX	LaTeX
\LaTeXe	LaTeX 2ε

奥村晴彦氏の jsclasses ではこれらに加えて次のロゴが用意されています．

\pTeX	pTeX
\pLaTeX	pLaTeX
\pLaTeXe	pLaTeX 2ε

```
「実は僕も{\TeX}使っています．」\\
「え？{\LaTeX}じゃないんですか？」\\
「いやぁ，僕は{\TeX pert}だからさ．
{\LaTeXe}も使ってませんよ．」\\
「ということは{\pTeX}は使うんですね？」
```

「実は僕も TeX 使っています．」
「え？ LaTeX じゃないんですか？」
「いやぁ，僕は TeXpert だからさ．LaTeX 2ε も使ってませんよ．」
「ということは pTeX は使うんですね？」

9.4　単位・通貨の出力について

文中でも数式中でも単位は基本的にはローマン体で出力するのが普通ですので

```
それは$y=30cm$となるので合計 300mm になる．
```

それは $y = 30cm$ となるので合計 300mm になる．

のような入力はおかしい訳です．この場合は

```
それは$y=30\,\mathrm{cm}$となるので
合計 300\,mm になる．
```

それは $y = 30\,\mathrm{cm}$ となるので合計 300 mm になる．

としたほうが良いでしょう．このように単位は数式中でも使うことがあるかもしれませんので\ensuremath で単位用の命令を作成します．例えば長さの単位である mm は

```
\newcommand{\mm}{\ensuremath{\,\mathrm{mm}}}
```

と定義しちゃいます．ただし LaTeX ですでに定義されている短い命令は再定義しないほうが無難ですので，リットル\l などは

```
\newcommand{\litter}{\,\ensuremath{\mathit{l}}}
```

とします．日本ではリットルをイタリック体にすることもあるようですが，基本的に単位は全てローマン体にしても良いようです．分野によっても区別が違うので調べてください．毎回単位を定義するのも大変なので単位用のマクロ units.sty を以下のように作成します．

```
%File:units.sty
\newcommand{\mm}{\ensuremath{\,\mathrm{\milli mm}}}
\newcommand{\cm}{\ensuremath{\,\mathrm{cm}}}
\newcommand{\Km}{\ensuremath{\,\mathrm{Km}}}
\newcommand{\mg}{\ensuremath{\,\mathrm{mg}}}
\newcommand{\Kg}{\ensuremath{\,\mathrm{Kg}}}
\newcommand{\cc}{\ensuremath{\,\mathrm{cc}}}
\newcommand{\litter}{\,\ensuremath{l}}
\newcommand{\Ohm}{\ensuremath{\,\mathrm{\Omega}}}
```

「あの単位の命令はなんだったかなぁ.」と考えているよりも\mathrm などを使ったほうが早いかもしれません.

通貨などを出力するためには LaTeX に標準で含まれる textcomp パッケージを使うと良いでしょう. これは古いエンコーディングだと使えませんので fontenc パッケージを読み込み

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

とします. フォントがビットマップになるということも危惧されますので特に不都合がなければ txfonts や pxfonts を併用して

```
\usepackage{txfonts,textcomp}% とか pxfonts
```

とするのがベターだと思います. 表 9.3 が textcomp によって使用できる記号一覧です. tectcomp にも含まれていない通貨などを探しているときは CTAN の

```
CTAN/info/symbols/comprehensive/
```

に記号の見本がありますのでそちらを参照してください.

9.5　あらかじめ定義されている見出しの変更

「目次」や「参考文献」などの見出しは\tableofcontents 命令や thebibliography 環境によって出力されます. この見出しの文字を変更するには次のようにします.

```
\renewcommand{\refname}{関連書籍}
```

標準的な和文の文書クラスでは表 9.4 の見出しが定義されています. \bibname 命令は jreport や jbook などでの定義で (j)article では\refname となっています. 奥村晴彦氏の jsclasses では節見出し番号の前と後にも文字列を表示できるようになっています.

```
\renewcommand{presectionname}{第}
\renewcommand{postsectionname}{節}
```

のように\presectionname や\postsectionname を**再定義します**.

表 9.3　textcomp で使える記号

„	\textquotestraightdblbase	⚭	\textmarried	{	\textlquill
—	\texttwelveudash	♪	\textmusicalnote	}	\textrquill
—	\textthreequartersemdash	~	\texttildelow	¢	\textcent
←	\textleftarrow	=	\textdblhyphenchar	£	\textsterling
→	\textrightarrow	˘	\textasciibreve	¤	\textcurrency
␢	\textblank	ˇ	\textasciicaron	¥	\textyen
$	\textdollar	˵	\textgravedbl	¦	\textbrokenbar
'	\textquotesingle	˶	\textacutedbl	§	\textsection
*	\textasteriskcentered	†	\textdagger	¨	\textasciidieresis
=	\textdblhyphen	‡	\textdaggerdbl	©	\textcopyright
⁄	\textfractionsolidus	‖	\textbardbl	ª	\textordfeminine
0	\textzerooldstyle	‰	\textperthousand	⅁	\textcopyleft
1	\textoneoldstyle	•	\textbullet	¬	\textlnot
2	\texttwooldstyle	°C	\textcelsius	Ⓟ	\textcircledP
3	\textthreeoldstyle	$	\textdollaroldstyle	®	\textregistered
4	\textfouroldstyle	¢	\textcentoldstyle	¯	\textasciimacron
5	\textfiveoldstyle	*f*	\textflorin	°	\textdegree
6	\textsixoldstyle	₡	\textcolonmonetary	±	\textpm
7	\textsevenoldstyle	₩	\textwon	²	\texttwosuperior
8	\texteightoldstyle	₦	\textnaira	³	\textthreesuperior
9	\textnineoldstyle	₲	\textguarani	´	\textasciiacute
⟨	\textlangle	₱	\textpeso	µ	\textmu
−	\textminus	£	\textlira	¶	\textparagraph
⟩	\textrangle	℞	\textrecipe	·	\textperiodcentered
℧	\textmho	⸘	\textinterrobang	※	\textreferencemark
◯	\textbigcircle	¡	\textinterrobangdown	¹	\textonesuperior
Ω	\textohm	₫	\textdong	º	\textordmasculine
⟦	\textlbrackdbl	™	\texttrademark	√	\textsurd
⟧	\textrbrackdbl	‱	\textpertenthousand	¼	\textonequarter
↑	\textuparrow	¶	\textpilcrow	½	\textonehalf
↓	\textdownarrow	฿	\textbaht	¾	\textthreequarters
`	\textasciigrave	№	\textnumero	€	\texteuro
★	\textborn	⅋	\textdiscount	×	\texttimes
o|o	\textdivorced	℮	\textestimated	÷	\textdiv
†	\textdied	○	\textopenbullet		
🍃	\textleaf	℠	\textservicemark		

9.6　目次再見

目次に出力される項目を制御したいときがあります．例えば見出し命令にアスタリスクをつけた場合（\chapter*など）は通し番号が付かずに目次にも出力されません．これを目次にも書き出すには

```
\addcontentsline{toc}{chapter}{なんとか}
```

表 9.4 定義済みの見出しの変更

命令	意味	標準的な定義
\prepartname	部見出し番号の前の文字	第
\postpartname	部見出し番号の後の文字	部
\prechaptername	章見出し番号の前の文字	第
\postchaptername	章見出し番号の前の文字	章
\contentsname	目次の見出しの	目次
\listfigurename	図目次の見出し	図目次
\listtablename	表目次の見出し	表目次
\bibname	thebibliography 環境の見出し	参考文献
\indexname	theindex 環境の見出し	索引
\figurename	図見出し番号の前の文字	図
\tablename	表見出し番号の前の文字	表
\appendixname	appendix 環境での見出しの前の文字	付録

とします.

```
\addcontentsline{⟨拡張子⟩}{⟨種類⟩}{⟨要素⟩}
\addtocontents{⟨拡張子⟩}{⟨要素⟩}
```

例えば文書クラスに jreport を使っていた場合に「謝辞」のような章を出力するときは

```
\chapter*{謝辞}
ありがとう，本当にありがとう．
```

のように入力しますが，これを目次にも追加するには

```
\chapter*{謝辞}\addcontentsline{toc}{chapter}{謝辞}
ありがとう，本当にありがとう．
```

のように\chapter*の直後に記述します．目次や図目次のある部分で改ページしたいときには

```
\addtocontents{toc}{\newpage}
\addtocontents{lof}{\newpage}
```

のようにします.

9.7 他段組

LaTeX では通常 1 段組と 2 段組しか制御できません.

```
\onecolumn
\twocolumn[⟨要素⟩]
\columnsep　（2 段組のときの段間）
\columnseprule　（2 段組のときの段間に引く罫線の太さ）
```

1 段組みにするためには\onecolumn を使い，2 段組にするには\twocolumn を使います．\twocolumn は改ページをしてから 2 段組を作成しようとします．そのため任意引数に何らかの要素を与えるとその要素をページ上部に 1 段組で出力します．

```
\columnsep 2zw
\columnseprule .4pt
\twocolumn[{\large\LaTeXe はどうです？}]
ここからの文章が 2 段組になるでしょう．{\LaTeX}での
他段組の実現は難しいそうです．
```

2 段組みにすると図表は用紙の文章幅\textwidth ではなく 1 段分の幅\columnwidth で張り込むことになります．また以下の二つの環境が使えます．

table*環境
figure*環境

table 環境や figure 環境にアスタリスクを付けるとその環境を 1 段分の幅でページの下部か上部に配置しようとします．

\twocolumn を使って 2 段組をすると最終ページの段の高さが揃わないので，格好悪いでしょう．これは multicol パッケージで 2 段組にすると段が揃いますし，balance パッケージを使っても可能です．

9.7.1　3 段組以上

Frank Mittelbach 氏の作成した multicol を使うと最高で 10 段まで段組できます．自動的に段の終わりの最終ページの文章の高さを揃えてくれます．ただし商用利用のためには作者の許諾と使用料が必要です．

\begin{multicols{⟨段数⟩}
文章内容
\end{multicols}

multicols 環境の場合は改ページされずに同じページに違う段組を混在できます．余り好ましくないことなので多用しないほうが良いでしょう．以下のような入力

```
\begin{multicols}{4}
このパッケージでは 10 段組まで他段組出来ますが， 同ページに違う段数の要素
を組み込むのは余り好ましいことではないので，特別な理由がない限り使うべき
ではありません．このパッケージでは段の終わりの最終ページの文章を自動で揃
えるので，従来の組版の規則にも合っています．
\end{multicols}
```

とう入力があるとすれば，次のような出力を得ることが出来ます．

このパッケージでは 10 段組まで他段組出来ますが, 同ページに違う段数の要素を組み込むのは余り好ましいことではないので, 特別な理由がない限り使うべきではありません．このパッケージでは段の終わりの最終ページの文章を自動で揃えるので, 従来の組版の規則にも合っています．

9.8 長さ

TEX/LATEX における**長さ**には伸縮するものとしないものの 2 種類があります．伸縮するものを**可変長の長さ**と呼び，伸縮しないものを**固定長の長さ**と呼びます．可変長の長さは**スキップ**と呼ばれることが多いようですが，本書では可変長の長さと呼びます．可変長の長さには**長さ**，**縮み率**，**伸び率**の三つの属性を持っています．固定長の長さは決まった値しか持ちません．可変長の長さは縮み率と伸び率に従ってバネのように伸縮します．長さの定義には\newlength が使えます．

```
\newlength{〈綴り〉}
\setlength{〈綴り〉}{〈長さ〉}
\addtolength{〈綴り〉}{〈長さ〉}
```

\setlength で長さを設定します．\addtolength では元の長さにさらに長さを足します．\newlength で定義された長さは可変長にも固定長にもなっても良いことになっています．まず\newlength で新規に長さを定義します．次に\setlength で値を決めます．

```
\newlength{\newa} newa=\the\newa\par
\setlength{\newa}{10mm} newa=\the\newa\par
\setlength{\newa}{10mm plus 3mm minus 2mm}
\par newa=\the\newa\par
\addtolength{\newa}{3mm} newa=\the\newa
```

newa=0.0pt
newa=28.45274pt
newa=28.45274pt plus 8.53581pt minus 5.69054pt
newa=36.98856pt plus 8.53581pt minus 5.69054pt

可変長の長さに値を足しても縮み率と伸び率には影響しないのが例の出力から分かるでしょう．

長さを設定するには次の命令も使えます．

```
\settowidth{〈綴り〉}{〈要素〉}
\settoheigth{〈綴り〉}{〈要素〉}
\settodepth{〈綴り〉}{〈要素〉}
```

長さに対して要素の幅を代入するには\settowidth を使います．高さに関しては\settoheight，深さには\settodepth を使います．このような命令の使い道を少し紹介しておきます．

```
%\newlength{\newa}
\newcommand{\fakewidth}[1]{%
  \settowidth{\newa}{#1}%
  \framebox[\the\newa][c]{\strut}}
解は{$\int f(x)dx$}となる．\par
解は\fakewidth{$\int f(x)dx$}となる．
```

解は $\int f(x)dx$ となる．
解は [ ] となる．

9.9　箱の操作

まずは LaTeX で用意されている**箱**について説明します．これらは ltboxes.dtx で定義されています．LaTeX における箱というのは文章や段落，数式や図表などの要素を格納する領域のようなものです．LaTeX の箱には**高さ**と**幅**と**深さ**の 3 種類の長さを持っています．さらに箱のどの点を基準にするかという**基準点**という座標も持ち合わせています．

9.9.1　枠のない箱

LaTeX ではなんとも簡単に複数の要素を一つの箱に収めることができます．

\makebox[⟨幅⟩][⟨位置⟩]{⟨要素⟩}

\makebox では箱の幅と箱の中の要素の位置を指定できます．箱の幅よりも要素の幅が狭いときに箱の左側に配置 ‘l’，中央に配置する ‘c’，右側に配置する ‘r’，最後に要素を均一に配置する ‘s’ の四つを使うことができます．

```
\makebox[3zw][l]{ほげ}と
\makebox[3zw][c]{げほ}と
\makebox[5zw][r]{あれ}と
\makebox[5zw][s]{G o o d !}です．
```

ほげ　と　げほ　と　　あれと G o o d !です．

要素の幅分の箱を作りたければ\mbox を使います．

\mbox{⟨要素⟩}

引数を省略すると要素分の幅を確保し\makebox を使うよりも効率が良いです．

```
\hspace*{\fill} 単なる予想ですが，この箱の
中では恐らく\mbox{改行が起こりません．}
```

単なる予想ですが，この箱の中では恐らく改行が起こりません．

9.9.2　枠のある箱

複数の要素を一つの塊として扱うようにするのが LaTeX における箱の役割のようなものです．箱には枠を付けることもできます．

\framebox[⟨幅⟩][⟨位置⟩]{⟨要素⟩}

\framebox も\makebox とほぼ同じですが罫線の太さ\fboxrule と罫線と要素の間隔\fboxsep の二つの長さを設定できます．\fboxrule は罫線の太さを，\fboxsep は枠と要素との距離を長さで指定します．

```
\framebox[3zw][l]{ほげ}と
{\fboxrule=3pt\framebox[3zw][c]{げほ}}と
\framebox[5zw][r]{あれ}と
\framebox[5zw][s]{G o o d !}です．
```

ほげ と げほ と あれ と G o o d ! です．

\makebox と同じように引数を省略すると要素分の幅を確保する\fbox が使えます．

\fbox{⟨要素⟩}

```
{\fboxsep=0pt\fbox{ぴったりだね}}．でも
{\fboxrule=.8pt\fbox{ちょっと太い}}
```

ぴったりだね．でも ちょっと太い

9.9.3 広範囲な箱

指定した箱の大きさで段落を組む\parbox 命令もあります．標準では字下げがされませんので必要があれば\parindent に長さを代入してください．

\parbox[⟨位置⟩][⟨高さ⟩][⟨要素の位置⟩]{⟨幅⟩}{⟨文字列⟩}

\parbox で作成された箱の基準をどこにするのかを ⟨位置⟩ で指定します．位置には上部 ‘t’，中央 ‘c’，下部 ‘b’ の三つが使えます．標準では中央になります．

```
\parbox{13zw}{段落が終わる命令\par
を使っても改行されますが\par
標準では字下げされません．}
```

段落が終わる命令
を使っても改行されますが
標準では字下げされません．

```
\parbox[c]{4zw}{箱が中央に．}\ldots
\parbox[t][3zw][c]{4zw}{文字が中央，
上が基準}\ldots
\parbox[b][3zw][t]{4zw}{文字が下に，
下が基準}\ldots
\parbox{3zw}{文字が上で下を基準}
```

箱が中央に．…文字が中央，上が基準…文字が下に，下が基準…文字が上で下を基準

ページのような箱を組む minipage 環境もあります．

\begin{minipage}[⟨位置⟩]{⟨幅⟩} ページ内容 \end{minipage}

minipage 環境では段落が組まれますし，脚注の出力も可能です．

```
この環境は~
\begin{minipage}[t]{7zw}
ページを組みあげるので脚注\footnote{脚注です．}
もページの中に出力されます．
\end{minipage}
~のようになります．
```

この環境は ページを組み あげるので脚注[a] もページの中に出力されます． のようになります．

[a] 脚注です．

9.9.4　箱の保存と使用

ある要素を箱の中に保存し，それを再利用できればエネルギー消費を減らすことができます．

```
\newsavebox{⟨綴り⟩}
```

箱を保存するためには保存する場所の確保を\newsavebox で行います．LaTeX が使っても良い箱は数が限られているのであらかじめいくつくらい使うのかを宣言してあげます．宣言するときはバックスラッシュを先頭に付けます．箱の中に要素を保存するときは\savebox か\sbox 命令を使います．

```
\savebox{⟨綴り⟩}[⟨幅⟩][⟨要素の位置⟩]{⟨要素⟩}
\sbox{⟨綴り⟩}{⟨要素⟩}
```

箱の幅や要素の位置を指定するときは\savebox を使います．もちろん幅を指定しないと要素の位置は指定できません．上記の命令のほかにも lrbox 環境があります．行に収まるくらいの要素を lrbox 環境の中に記述すると ⟨綴り⟩ の箱に代入します．

```
\begin{lrbox}{⟨綴り⟩}
要素
\end{lrbox}
```

これまでの命令では箱を用意してその中に要素を保存するだけですので，保存した箱を\usebox 命令で使います．

```
\usebox{⟨綴り⟩}
```

\usebox 命令を使うと ⟨綴り⟩ の箱に保存されている要素を複数回再利用できます．

```
\newsavebox{\hoge}
\savebox{\hoge}{\LaTeXe から\LaTeX3 へ}
\usebox{\hoge}, \usebox{\hoge}.\par
\usebox{\hoge}, \usebox{\hoge}.\par
\sbox{\hoge}{}  \usebox{\hoge}
```

LaTeX 2ε から LaTeX3 へ, LaTeX 2ε から LaTeX3 へ.
LaTeX 2ε から LaTeX3 へ, LaTeX 2ε から LaTeX3 へ.

LaTeX の作業領域は限られていますので\setbox 命令で ⟨綴り⟩ の箱を使い終わったら空にします．

9.9.5　箱の上げ下げ

ある要素を箱の中に入れて，さらに上げ下げを同時に行う\raisebox 命令もあります．

```
\raisebox{⟨上げ下げ⟩}[⟨高さ⟩][⟨深さ⟩]{⟨要素⟩}
```

\raisebox 命令の中には文字列や他の箱も挿入できます．

```
ありゃま，\raisebox{1zw}{ほげほげ}は
\raisebox{-1zw}{どれどれ}で，
\raisebox{1.5zw}{\fbox{枠付きの箱}}だよ．
```

ありゃま，ほげほげ は どれどれ で，[枠付きの箱] だよ．

この命令でこんな悪ふざけもできます．

```
\fbox{ほげ，\raisebox{1zw}{ほげ}
ほげ，\raisebox{-1zw}{ほげ}
ほげ，\raisebox{1zw}{ほげ}
ほげ，\raisebox{-1zw}{ほげ．}}
```

ほげ ほげ
ほげ，ほげ，ほげ，ほげ，
ほげ ほげ．

9.9.6　罫線と下線

箱とは違うのですが**罫線**をここで紹介しておきます．

\rule[⟨上げ率⟩]{⟨幅⟩}{⟨高さ⟩}

\rule 命令は使いものになります．見えない罫線を引くこともできます．例えば幅が 0 pt でも高さのある罫線，高さが 0 pt でも幅のある罫線が使えますから，こんな使い方もできるわけです．枠の見える状態での例を見てください．

```
ほげ\fbox{\rule{0pt}{3zw}\rule{4zw}{0pt}}
ほげ\fbox{\rule{0pt}{3zw}\rule{2zw}{0pt}}
```

ほげ ほげ

箱とは違うのですが**下線**も紹介しておきます．下線は\underline を使います．

\underline{⟨要素⟩}

\underline の中に箱を入れることもできますし，何を入れても構いません．

```
\underline{\fbox{枠付きの箱}の下線}の調
子はどうですか？そりゃぁ，\underline{下
線}ですよ．
```

枠付きの箱の下線 の調子はどうですか？そりゃぁ，
下線 ですよ．

9.9.7　枠付きの箱その 2

LaTeX の標準では\fbox と\framebox 命令が使えます．

```
\fbox{これは枠付きの箱} \\
\framebox[10zw][c]{ほげほげ}\\
\fbox{\fbox{2 重枠の箱}} \\
{\fboxrule=1pt\fbox{枠の太い箱}}\\
{\fboxsep=0pt\fbox{文字と枠がぴったりな箱}}
```

これは枠付きの箱
ほげほげ
2 重枠の箱
枠の太い箱
文字と枠がぴったりな箱

Timothy Zandt 氏による fancybox を使ってみると枠付きの箱を出しやすいでしょう．枠付きの箱に文字列を入れるための命令として\shadowbox，\doublebox，\ovalbox，\Ovalbox の四つがあります．

```
\shadowbox{影付きの箱} \\
\doublebox{2 重の枠付きの箱 \\
\ovalbox{辺が丸い箱} \\
\Ovalbox{太くて辺が丸い箱}}
```

影付きの箱

2 重の枠付きの箱 辺が丸い箱 太くて辺が丸い箱

fancybox を使うと Sbox 環境というものが使えて，これは環境内のものを箱（レジスタ）である\TheSbox に保存します．このようにして保存した箱を\fbox などで囲みます．例えば minipage を枠で囲む fminipage 環境を作成するのであれば

```
\newenvironment{fminipage}%
  {\begin{Sbox}\begin{minipage}}%
  {\end{minipage}\end{Sbox}\fbox{\TheSbox}}
```

とします．

```
\newenvironment{fminipage}%
  {\begin{Sbox}\begin{minipage}}%
  {\end{minipage}\end{Sbox}\fbox{\TheSbox}}
\begin{fminipage}{.8\linewidth}
この環境は枠で囲まれます．というか，
囲んでくれないと困ります．
\end{fminipage}
```

この環境は枠で囲まれます．というか，囲んでくれないと困ります．

数式環境などを枠で囲もうと思うとき，数式番号も含めて枠の中に入れるのはちょっと面倒かもしれません．\fbox などの命令を使うとそこから数式を組み立てるモードではなく文章を組み立てるモードになりますので，もう 1 度数式環境を書きます．

```
\( y=f(x)+\fbox{\(C\)}\)
\begin{equation}
 \fbox{\(y=f(x)\)}
\end{equation}
```

$y = f(x) + \boxed{C}$

$$\boxed{y = f(x)} \tag{9.1}$$

番号付きの数式を文章幅いっぱいに枠で囲むには

(\linewidth (文章幅) −2\fboxrule −2\fboxsep)

を計算します．これには calc パッケージが使えます．

```
%\usepackage{calc,fancybox}
\fbox{\parbox{%
  (\linewidth-2\fboxrule-2\fboxsep)}{
\begin{equation} y=f(x) \end{equation}}}
```

$$y = f(x) \tag{9.2}$$

eqnarray 環境は枠で囲もうと思うと fancybox の場合は Beqnarray が用意されていますので，こちらの環境を\fbox などで囲みます．

```
\fbox{\begin{Beqnarray}
 y&=&f(x)\\
  &=&1/x
\end{Beqnarray}}
```

$$\boxed{\begin{aligned} y &= f(x) && (9.3)\\ &= 1/x && (9.4)\end{aligned}}$$

table 環境や figure 環境などで見出しも含まれる要素は\Sbox で一度保存したものを枠で囲むようにします．こうすると

```
%\usepackage{calc,fancybox}
\newenvironment{ftable}[1][htbp]%
  {\begin{table}[#1]
   \begin{Sbox}\begin{minipage}{%
  (\linewidth-2\fboxrule-2\fboxsep)}}%
  {\end{minipage}\end{Sbox}\fbox{\TheSbox}\end{table}}
```

のように ftable や ffigure のような環境が定義できます．この出力例が表 9.5 となります．

LaTeX2.09	LaTeX 2ε	LaTeX3

表 9.5　LaTeX の歴史

fancybox では center 環境のように行揃えを行う環境があります．これを枠で囲むには Bcenter，Bflushleft，Bflushright 環境を使います．

```
\fbox{\begin{Bcenter}
ここは \\ 中央揃えで \\ 枠付きなのだぁ．
\end{Bcenter}}
```

ここは
中央揃えで
枠付きなのだぁ．

```
\fbox{\begin{Bflushleft}
ここは \\ 左揃えでぇ〜 \\ 枠付き．
\end{Bflushleft}}
```

ここは
左揃えでぇ〜
枠付き．

```
\fbox{\begin{Bflushright}
ここはですねぇ \\ 右揃えで \\ 枠付き．
\end{Bflushright}}
```

ここはですねぇ
右揃えで
枠付き．

minipage 環境の中で行揃えの命令を使うことも考えられますが，その場合は横幅を指定しないといけないので，これらの環境は便利でしょう．箇条書き環境も同様に minipage の中に入れて\fbox で囲むことも考えられますが，fancybox では Bitemize，Benumerate，Bdescription の三つがすでに定義されているので，これらの環境を枠で囲みます．

```
\fbox{\begin{Benumerate}
 \item ほげほげ.
 \item あれあれあれあれ？.
 \item どれどれ. \end{Benumerate}}
```

1. ほげほげ.
2. あれあれあれあれ？.
3. どれどれ.

\verb 命令を枠で囲むときは\VerbBox 命令を使います.

```
\VerbBox{\fbox}{%
  \verb|\VerbBox{\fbox}{\verb+ 文章 +}|}
```

\VerbBox{\fbox}{\verb+ 文章 +}

verbatim 環境を枠で囲む場合はやはり先に Sbox 環境で囲んで\TheSbox を枠で囲むようにします．この場合 fverbatim 環境を定義したほうが便利でしょう．ただし，この場合\VerbatimEnvironment 命令と Verbatim 環境を併せて使わないとエラーになります.

```
\newenvironment{fverbatim}[1]%
  {\VerbatimEnvironment \begin{Sbox}%
    \begin{minipage}{#1} \begin{Verbatim}}%
  {\end{Verbatim}\end{minipage}\end{Sbox}%
    \fbox{\TheSbox}}
\begin{fverbatim}{.8\linewidth}
このべた書き環境は枠で囲まれて
出力されます.
\end{fverbatim}
```

このべた書き環境は枠で囲まれて
出力されます.

このほかにも fancybox にはファイルから読み込んだ行をべた書き環境に出力する命令なども用意されています.

9.10　空白の挿入

LaTeX にはいろいろ空白が用意されているのですが，それらは**空き**に含まれます．単語間に挿入される程度の空きを基準とするとその 4 倍の空きを ‘quad’（クワタ）と呼びます．和文組版では空きの基準となるのは全角 1 文字分の幅であり，これを**全角空白**などと呼びます．全角空白一つ分の空きを**全角空き**，全角空白二つ分の空きを**倍角空き**と呼びます．さらに 4 分の 1 の場合は**四分空き**，六分の 5 ならば**二分三分**と呼んだりします．欧文の ‘quad’ と和文の「クワタ」では若干長さが異なりますので，本冊子では二つを区別して表します.

9.10.1　水平方向の空き

水平方向の空きにはその両側での改行を許すものと許さないものがあります．主な空きを制御する命令は表 9.6 の通りです．表 9.6 は基本的に空きの前後での改行を行っても良いことになっています.

```
ユーザが{\quad}原稿の中{\qquad}で
空きの調節を\ するのはいかがなものか.
```

ユーザが　原稿の中　　で空きの調節を するのはいかがなものか.

表 9.6　改行を許す水平方向の空き

命令	意味
\␣	適切な**単語間空白**（約 1/4 quad 分）
\quad	1 quad 分の空き
\qquad	2 quad 分の空き
\enspace	1/2 quad 分の空き
\enskip	適切な約 1/2 quad 分の空き
\thinspace	1/5 quad 分の空き
\negthinspace	-1/5 quad 分の空き

表 9.6 の命令は改行を許しますが表 9.7 では空きの前後での改行を許しません．改行を

表 9.7　改行を許さない水平方向の空き

命令	意味
\,	3/18 quad 分の空き
\:	4/18 quad 分の空き
\;	5/18 quad 分の空き
\!	-3/18 quad 分の空き
~	適切な**単語間空白**

許さないので行頭・行末が不揃いになるときがあります．

```
Donald~E. Knuth made \TeX\@.
Leslie~Lamport made \LaTeX\@.
```

Donald E. Knuth made TeX. Leslie Lamport made LaTeX.

自分で水平方向の空きの長さを指定するならば\hspace* 命令が使えます．

```
\hspace*[⟨長さ⟩]
```

アスタリスクをつけると行頭・行末でも使えるようになります．

```
ほげほげ\hspace{2zw}ほげほげ\par
\hspace*{-2zw}ほげほげ．
```

ほげほげ　　ほげほげ
ほげほげ．

奥村晴彦氏の jsclasses を使っているときには ‘pt’ や ‘cm’ などの単位は使わずに ‘truept’ や ‘truecm’ などを使わないと長さがずれます．これが面倒ならば文章で使われているフォントに応じて基準の変わる ‘em’ や ‘zw’ などを使ってください．

9.10.2　垂直方向の空き

自分で長さを指定する垂直方向の空きにおいては\addvspace と\vspace*の二つが使えます．\vspace*はアスタリスクを付けないとページの最上部・最下部では有効になり

ません．あらかじめ長さの決まっている垂直方向の空きとして\smallskip，\midskip，\bigskip の三つがありますが，これは**スキップ**と呼ばれるもので可変長の空きが挿入されます．「大体で良いからこれくらいの空きを入れてね．」程度の意味を持っています．垂

表 9.8　垂直方向の空き

命令	意味
\smallskip	3 pt±1 pt の空き
\medskip	6 pt±2 pt の空き
\bigskip	12 pt （+4 pt か −2 pt） の空き

直方向の空きは紙面の多くの部分を空きで占有するので無駄が多くなります．LaTeX では図表と段落のあいだやそのほか必要と思われるところには半自動的に空きが挿入されるようになっておりますので，闇雲に垂直方向の空きを挿入するのは好ましくないと思われます．

長さを自分で指定して空きを挿入する場合は\vspace*と\addvspace が使えます．

```
\addvspace{⟨長さ⟩}
\vspace*{⟨長さ⟩}
```

\vspace*のアスタリスクを外すとページの最上部・最下部での空きの挿入が有効になりません．\addvspace は直前の空きがどれくらいかも調べているので\vspace よりも適当な空きを挿入します．

この\vspace*{2zw}だと全角 2 文字分の垂直方向の空きが挿入されると思われます．

このだと全角 2 文字分の垂直方向の空きが挿入される

と思われます．

9.11　伸縮する糊

LaTeX ではバネのように伸縮する**糊**のような便利な道具があります．これは活版印刷時代における活字職人が経験で挿入する**詰めもの**に似ています．

昔は果物を段ボール箱に入れて郵送するときには果物の周囲に新聞紙や籾殻（もみがら）などを入れていました．箱の中のものが適切に宛て先に届けられるように入れるこの新聞紙や籾殻を**詰めもの**と呼びます．LaTeX においてもある領域の中での空きを制御するためとか，適切な表示をするための詰めものが使われます．LaTeX で使われる詰めものは**グルー**と呼ばれています．グルーは伸縮自在でバネのように要素と要素をくっ付けるので伸縮する**糊**とも呼ばれています．

グルーは特別な単位 ‘fil’ と ‘fill’ によって定義されています．‘fil’ は良い按排（あんばい）で 0 からかなり大きい空きまで挿入する働きをします．‘fill’ は ‘fil’ よりも大きく無限に近い空きを挿入する働きをします．‘fil’ は ‘fill’ よりも弱いので ‘fill’ のほうが優先されます．

表 9.9　LaTeX で使用できるグルー

命令	意味
\hfil	水平方向にかなり大きく伸びる空き
\hfill	水平方向に無限に大きく伸びる空き
\hss	水平方向にかなり大きく伸び縮みする空き
\vfil	垂直方向にかなり大きい空き
\vfill	垂直方向に無限に大きい空き
\vss	垂直方向にかなり大きく伸び縮みする空き

```
この空き{\hfil}は{\hfil}かき消されません．\par
この空きは{\hfil}恐らく{\hfill}かき消されます．
```

この空き　は　かき消されません．
この空きは恐らく　かき消されます．

‘fil’ と ‘fill’ の二つのグルーの両方を使っている場合は空きが挿入されない場合がありますから気を付けてください．

もう少し相対的なグルーを挿入するには\strecth を使います．

```
\stretch{⟨整数⟩}
```

これは ‘fill’ の何倍かのグルーを挿入しても良いようにするために使用できます．

```
hoge\hspace{\stretch{3}}hoge%
\hspace{\stretch{1}}hoge.\par
hoge\hspace{\fill}hoge\hspace{\fill}%
hoge.\par
```

hoge　　hoge　　hoge.
hoge　　hoge　　hoge.

\fill は\hspace や\vspace の引数使うことができるグルーです．例えばページの最上部に無限に近いグルーを挿入するときは以下のようにします．

```
\vspace*{\fill}を使ってもこの出力例で
は伸びません．
```

を使ってもこの出力例では伸びません．

9.11.1　空白ではないグルー

\hfil や\hfill は空白を挿入するグルーでした．そうではなく，与えられた領域を程よく文字列や要素で埋めてくれる詰めものもあると便利です．目次でも使われているのがお分かりになるでしょう．これらを**リーダ**と呼びます．LaTeX であらかじめ定義されているリーダは表 9.10 となります．‘fill’ ですから無限に伸びます．\dotfill と \hrulefill は ltplain.dtx で，それ以外は fontdef.dtx にて定義されています．段落中で使うと段の終わりまで伸びます．

```
\makebox[5zw][l]{ほ\dotfill げ}\par
\hrulefill\par
ほげ\leftarrowfill ほげ\par
ほげほげ\rightarrowfill\par
```

ほ……げ
ほげ ←―――――――――――――――― ほげ
ほげほげ ――――――――――――――――→

表 9.10　主なリーダ

命令	出力例
\dotfill	.....
\hrulefill	———
\leftarrowfill	←———
\rightarrowfill	———→
\downbracefill	⏟
\upbracefill	⏞

これらのリーダは\leaders という命令によって定義されています．自分でこのような命令を定義したいときは\leaders，\cleaders，\xleaders の三つが使えます．例えば領域を「ほげ」で埋めるような命令ならば次のようになります．

```
\newcommand{\hogefill}{\leavevmode\leaders%
  \hbox{ほげ}\hfill\kern0pt}
\hogefill\par
あら\hogefill そう．\par
```

ほげほげほげほげほげほげほげほげほげほげほげほげほげ
あらほげほげほげほげほげほげほげほげほげほげほげそう．

9.12　原稿を複数のファイルに分ける

大規模な文書になるとそれを一つのファイルにまとめるのは効率が悪い場合があります．第 3 章は田中さんが編集し第 5 章は斉藤さんにお任せする，という状況では第 3 章と第 5 章の原稿は別々に存在させたいものです．この場合は原稿を複数のファイルに分けます．

```
\include{⟨ファイル名,...⟩}
\input{⟨ファイル名,...⟩}
\includeonly{⟨ファイル名,...⟩}
```

\include 命令はファイルを読み込むときに必ず新しいページから始めます．大規模な文書で章の区切りや節の区切りなどで使用します．この命令で取り込むときはファイルを章ごとに（\chapter ごと）に分けることが考えられます．\input はそのままの意味で指定されたファイルをそのまま親の LaTeX のソースファイルに取り込みます．取り込むファイルの拡張子が.tex ならば拡張子を省略しても構いません．

9.13　付録の追加

文書の最後に付録としてプログラムリストを載せるとか，本文とは直接的に関係のない資料を載せるときは\appendix 命令を使うか，appendix 環境を使うかの 2 通りの方法があります．appendix 環境を使う場合は

```
\begin{appendix}
```

追加する内容

```
\end{appendix}
```

という記述が可能ですから付録の範囲を指定できます．あえて\appendix 命令を使う必要もないでしょう．この命令を付けた後の文章は付録として扱われ，見出しの番号付けが自動的に大文字のアルファベットに変更され，‘A’ からカウントされるようになります．あとは通常通り見出しの定義をして文章を記述するだけです．

9.14 人名の敬称の統一

一般的に一つの文書において人名の敬称は統一します．これはけっこう面倒な作業で，登場する人名を索引に追加する場合などの手間もあるかも知れません．その場合，新たに何か便利な命令を作るのが良いでしょう．始めに索引に追加しない簡単な例を示します．この場合，敬称だけを付け足す簡単なマクロです．

```
\newcommand{\hito}[1]{#1 氏}
```

これだけです．後は

```
\hito{渡辺徹}のおかげで 1 輪車に乗れるようになった．
```

のように使うだけです．

9.15 翻訳作業

しばしば日本語ではない言語で書かれた文書を訳す作業があります．運良く原書の LaTeX の原稿が手に入ったとすると，作業は幾分楽になります．例えば以下のような原稿があったとします．

```
Hello, everyone! I'm a student at Future University Hakodate.
Today, please let me talk about my future plan.
First, ...
```

これを普通に翻訳すると

皆さん，こんにちは．私は公立はこだて未来大学の生徒です．
今日は私の未来計画についてお話したいと思います．まず，

となりますが，どうせなら原書の英文も削除したくありませんので

```
%Hello, everyone! I'm a student at Future University Hakodate.
皆さん，こんにちは．私は公立はこだて未来大学の生徒です．
%Today, please let me talk about my future plan.
今日は私の未来計画についてお話したいと思います．
%First, ...
まず，
```

のように入力すると英文と和文の対応が取れて分かりやすいでしょう．Word などではマネのできない芸当ですよね．1 行ずつに分ける必要はなく，非常に長い文章の場合は 1 段落ごとに対応させるのも良いでしょう．

9.16　用語の統一

9.15 節のような場合には用語の統一というが必要になってきます．一つの書籍を複数の訳者で共同翻訳するときに専門用語の場合や新語の場合は語句を統一しなければ，読者を混乱させます．統一されていない事態を避けるためにはマクロを作成しておきます．

```
Hello, everyone! I'm a student at Future University-Hakodate.
```

という文章があったとして ‘Future University-Hakodate’ という用語が新語であったとしましょう．この用語をどんな単語に訳すのかをまだ決められない段階では次のようなマクロを作成します．

```
\newcommand{\FUN}{Future University-Hakodate}
```

訳者のあいだで用語の訳が決まったならば

```
\newcommand{\FUN}{公立はこだて未来大学}
```

とします．他にも人名や専門用語，難解な単語などに遭遇したときはこのようなマクロを全ての訳者が参照できるファイルに収めておきます．

9.17　色

TeX/LaTeX では白と黒の 2 色しか理解できません．他の色を表現しようと思えば color パッケージなどの力を借りてデバイスドライバに全てを任せることになります．ですから色はドライバに依存しますし，プリンタがモノクロプリンタならばどうがんばってもグレースケールのページしか出力できません．

色には**色相・明度・彩度**の三つの要素があります．色相とは青や緑などの波長の種類，明度はその波長の明るさ，彩度は黒色の少なさを表します．

9.17.1　要素に色を付ける

LaTeX は白と黒の 2 色しか理解できません．ですからそのほかの色を付ける場合はデバイスドライバに全てを任せます．グレースケースでも同様です．色を付けるためには graphics パッケージと同封されている David Carlisle 氏が作成した color パッケージを使います．color を使うにはまず graphicx と同じように使用するデバイスドライバを指定します．Dvipdfm*x* などを使っているときは

```
\usepackage[dvipdfm]{color}
```

のようにします．パッケージオプションとして次のようなものがあります．

usenames　color パッケージで定義されている色の名前を全て使えるようにします．
usenames で使用できる色名は口絵 I の通りです．

dvipsnames　dvips で使用できる色を名前を指定して使えるようにする．

色の指定には四つあります．

`rgb`　Red，Green，Blue の 3 色を混ぜ合わせて**加法混色**で色を指定します．それぞれの色は 0 から 1 のあいだで指定します．

`cmyk`　Cyan，Magenta，Yellow，Black の 4 色を混ぜ合わせて**減法混色**で色を指定します．これも 0 から 1 のあいだで指定します．

`gray`　グレースケースで 0 から 1 のあいだで濃さを決めます．

`named`　あらかじめ定義された名前を使うことを意味します．

新規に色の名前を定義するときは`\definecolor` を使います．

```
\definecolor{⟨名前⟩}{⟨種類⟩}{⟨値⟩}
```

⟨種類⟩ には上記の四つが選択できます．例えば以下のように定義します．

```
\definecolor{MyGray}{gray}{0.85}
\definecolor{MyRed}{rgb}{0.3,0,0}
\definecolor{MyYellow}{cmyk}{}
```

文字列の色は`\textcolor` を使って色を指定します．

```
\textcolor{⟨色⟩}
\textcolor[⟨種類⟩]{⟨値⟩}{⟨文字列⟩}
```

ある範囲中の要素には`\color` 命令で指定します．

```
\color{⟨色⟩}
\color[⟨種類⟩]{⟨値⟩}
```

```
\textcolor{Gray}{この文字は灰色}\\
\textcolor[rgb]{1,0,0}{赤になりたい}\\
{\color{Gray}{\fbox{枠も文字も灰色です．}}}
```

この文字は灰色
赤になりたい
枠も文字も灰色です．

基本色を`\color` 命令で指定せずにそのまま色の名前を使えるようにするには

```
\newcommand{\black}{\color{black}}
\newcommand{\gray}{\color{Gray}}
\newcommand{\white}{\color{white}}
\newcommand{\red}{\color{red}}
\newcommand{\green}{\color{green}}
\newcommand{\blue}{\color{blue}}
\newcommand{\cyan}{\color{cyan}}
\newcommand{\magenta}{\color{magenta}}
\newcommand{\yellow}{\color{yellow}}
```

のように定義するだけです.

```
\newcommand{\gray}{\color{Gray}}
{\gray ここは灰色になります}
```

ここは灰色になります

ページの背景色を指定するときは\pagecolor 命令を使います.

\pagecolor{⟨色⟩}
\pagecolor[⟨種類⟩]{⟨値⟩}

これは命令を使ったそのページからずっと色の指定が有効ですので，1 ページだけの色を変更したいときはページの切り替わるだろう部分で色を元々の色に戻します.

枠付きの箱の背景を指定できる\colorbox 命令があります.

\colorbox{⟨色⟩}{⟨文字列⟩}
\colorbox[⟨種類⟩]{⟨値⟩}{⟨文字列⟩}

```
\colorbox{red}{背景が赤になる}\\
\colorbox[gray]{.8}{背景は灰色}\\
\colorbox{black}{\color{white}%
  背景が黒で文字は白}
```

背景が赤になる

背景は灰色

背景が黒で文字は白

さらに枠の色とその背景の色を指定できる\fcolorbox もあります.

\fcolorbox{⟨枠の色⟩}{⟨背景色⟩}{⟨文字列⟩}
\fcolorbox[⟨種類⟩]{⟨枠の色の値⟩}{⟨背景色の値⟩}{⟨文字列⟩}

\fcolorbox では\fbox と同じように\fboxrule と\fboxsep を調整できます.

```
\fcolorbox{red}{blue}{枠が赤で背景が青}\\
\fcolorbox[gray]{.5}{.8}{枠も背景も灰色}\\
{\fboxrule=2.4pt\fcolorbox{blue}{red}%
  {枠の線幅の調節}}
```

枠が赤で背景が青

枠も背景も灰色

枠の線幅の調節

9.18　プログラムソースの挿入

プログラムなどのソースコードを載せるときに，verbatim だけでは寂しいと感じる方も多いと思います．変数やコメントなどをうまく整形するプログラムやマクロパッケージがあります．アルゴリズムの行数を文中で指定したいときには行数も表示されるとうれしいものです．Texinfo のように変数名や関数名なども半自動的に索引に追加されるとうれしいものです．ソースコードの整形プログラムの中で私が良いと思うのが Carsten Heinz 氏が作成した listings です．これは正式には日本語が通りませんが，吉永徹美氏が作成した自称強引なマクロで切り抜けることができます．これに関しては奥村晴彦氏の掲示板の『汎用的な浮動体』

http://oku.edu.mie-u.ac.jp/~okumura/texfaq/qa/21172.html

という一連の書き込みを参照してください．結果としてこれをまとめたファイルを jlisting.sty として置いておきます．おそらく私のサイトの

http://tex.dante.jp/

にあると思われます．他にも lgrind，morevrb，progc などの類似品がありますがそれらの包括的なパッケージが listings です．この listings だけを使用していれば特に困ることもないと思います．

listings を使用するときにはプリアンブルに

```
\usepackage[オプション]{listings}
```

のようにします．このときに指定するパッケージオプションは

draft　ソースコードを出力しないようにします．ドキュメントクラスオプションですでに *draft* を指定している場合はそれが反映されるので，さらにオプションを記述する必要はありません．

final　*draft* オプションを無効にし，実際にソースコードを整形します．

savemem　TeX/LaTeX のメモリをなるべく消費しないようにします．

などがあります．listings は TeX のメモリをたくさん使いますので原稿執筆の段階では *draft* オプションを付けたほうが良いでしょう．

listings の使い方は大きく分けて二つです．一つは lstlisting 環境にソースコードを記述する方法，二つ目はプログラムのソースコードをファイルから入力する方法です．一つ目の LaTeX の原稿に直接記述する方法は次のような文法になります．

```
begin{lstlisting}[⟨設定 1, 設定 2,...⟩]
ソースコード
end{lstlisting}
```

例として C のソースを記述するならば

```
\begin{lstlisting}[language=c]
int main( void ){
   printf("Hello, World!!\n");
}
\end{lstlisting}
```

とになります．

二つ目の方法としては\lstinputlisting 命令を使ってファイルからソースコードを読み込みます．

```
\lstinputlisting[⟨設定 1, 設定 2,...⟩]{⟨ファイル名⟩}
```

⟨設定⟩ を毎回指定していたのでは疲れますので\lstset 命令で

```
\usepackage[dvips]{color}
\lstset{%listings の表示設定
    frame=tbrl,% 枠を上下左右に表示する
    backgroundcolor={\color[gray]{0.85}},% 背景を灰色に
    numbers=left,% 行番号を左に
    numberstyle=\scriptsize,%
    stepnumber=1,%1 行おきに行番号を
    numbersep=1zw}% ソースと行番号の間隔
```

のように設定しておけば全てのソースコードに共通の設定をすることができます．この設定の場合は上下左右に枠を表示させ，背景色を薄いグレーにし，行番号を左側の 1 行おきに表示するようにします．

いくつかの設定をひとまとめにしたいときは\lstdefinestyle 命令を使ってスタイルを定義します．

\lstdefinestyle{⟨スタイル名⟩}{⟨設定⟩}

例えば行番号に関しては左側に 1 行おきに表示させたいので

```
\lstdefinestyle{number1}{numbers=left, numberstyle=\scriptsize,%
   stepnumber=1, numbersep=1zw}
```

という定義をして

```
\begin{lstlisting}[language=c,style=number1]
```

のように指定したり，または特定の言語に共通のスタイルを用いるときは

```
\lstdefinstyle{C}{language=c,style=number1}
```

として

```
\begin{lstlisting}[style=c]
```

と使うこともできます．

ある言語のソースコード用に新しい環境を定義できます．

\lstnewenvironment{⟨環境名⟩}{⟨始めの設定⟩}{⟨終わりの設定⟩}

例えば C 言語用の環境を新たに定義するときは

```
\lstnewenvironment{C}{%
\lstset{language=c,style=number1}}{}
```

のとしても良いでしょう．

ソースコードを浮動体（float）として出力することもできます．具体的には

```
\begin{lstlisting}[language=c,float,caption={listings の使用例}]
int main( void ){printf("Hello, World!!\n");}
\end{lstlisting}
```

のようになります．

9.18.1　言語の設定

listings では非常に多くの言語用の設定が定義されています．あらかじめ定義されていない言語は自分で新たに定義することもできます．あらかじめ定義されている言語は表 9.11 の通りです．角括弧を含む言語の指定は

表 9.11　listings で使用できる主な言語

Assembler	Basic	C
C++	Cobol	csh
[Sharp]C	Delphi	Fortran
HTML	IDL	Java
ksh	[LaTeX]TeX	Lisp
Logo	make	Mathematica
Matlab	MetaPost	MuPAD
Octave	Pascal	[plain]TeX
Perl	PHP	Prolog
Python	Ruby	Scilab
SQL	tcl	[tk]tcl
TeX	VBScript	[Visual]Basic
[Visual]C++	VRML	XML

```
\begin{lstlisting}[\language={[LaTeX]TeX}]
```

のように波括弧の中に入れて指定するのが安全です．

言語があらかじめ定義されていなくても，\lstdefinelanguage 命令で自分で言語の定義をすることができます．

```
\lstdefinelanguage{〈言語名〉}{〈設定〉}
```

具体的には

```
\lstdefinelanguage{CCC}{
   morekeywords={short, int, long, float, double},% 予約語
   sensitive=false,
   morecomment=[l]{//},% 行末コメント
   morecomment=[s]{/*}{*/},% 範囲コメント
   morestring=[b]",% 文字列
   morestring=[d]',% 文字}
```

のようにします．

標準の listings の字詰めや行送りはローマン体でなるべく等幅になるように字詰めをしています．そちらのほうが正しい設定なのでしょうが，見栄えを考えると

```
columns=[l]{fullflexible}
```

としたほうが良いかもしれません．タイプライタ体で出力しない場合は等幅になりませんので，ソースコードとしては不適切な設定かもしれません．

9.18.2　主な設定値

空白などに関する設定です．

lineskip=⟨長さ⟩　行間にさらに加える空きを調節します．標準は 0 pt.

firstline=⟨整数⟩　入力されているソースコードを，どの行から読み込むを設定します．標準は 1.

lastline=⟨整数⟩　ソースコードのどの行まで読み込むかを設定します．標準は 9999999.

tabsize=⟨整数⟩　ソースコード中のタブ文字を何文字分にするかを設定します．標準は 8 です．

showtabs=⟨true|false⟩　タブ文字を可視・不可視にします．

tab=⟨文字列⟩　タブ文字を ⟨文字列⟩ に置き換えて表示します．$\Longrightarrow$（⟹）などを使う方法もあります．

showspace=⟨true|false⟩　スペースを可視（␣）・不可視（ ）にします．

linewidth=⟨長さ⟩　ソースコードの文章幅を指定します．標準は\linewidth です．

xleftmargin=⟨長さ⟩　左側の余白を指定します．標準は 0 pt です．

xrightmargin=⟨長さ⟩　右側の余白を指定します．標準は 0 pt です．

breaklines=⟨true|false⟩　ソースコードの文章幅よりも長い文字列を自動的に折り返すかを指定します．標準は false です．

prebreak=⟨文字列⟩　行頭に挿入する文字列を指定します．

postbreak=⟨文字列⟩　行末の挿入する文字列を指定します．例えば行末の改行の位置を特別に示したいときは ‘postbreak=\return’ のようにすると良いでしょう．

言語の設定です．

language={⟨言語⟩}　あらかじめ定義されている主な言語については表 9.11 をご覧ください．

文字の表示の仕方の設定です．

basicstyle=⟨スタイル⟩　通常のソースのスタイルを設定します．⟨スタイル⟩ には \small や\ttfamily などのフォントの宣言をする命令が使えます．

commentstyle=⟨スタイル⟩　コメントのスタイルを設定します．

stringstyle=⟨スタイル⟩　ソースコード中の文字列のスタイルを設定します．

keywordstyle=⟨スタイル⟩　キーワード，プログラミング言語で言えば予約語のスタイルの設定です．

行番号の表示の設定です.

numbers=⟨none|left|right⟩　行番号をどのように表示するかの設定です.
stepnumber=⟨数字⟩　何行おきに行番号を表示するかを設定します.
numberstyle=⟨スタイル⟩　行番号のスタイルを指定します.
firstnumber=⟨auto|last|⟨数字⟩⟩　行番号の開始の数字を指定します.

浮動体などに関する設定です.

float=⟨htbp の部分集合⟩　ソースコードを浮動体として出力します.
caption=⟨文字列⟩　見出しの文字列を指定します.
label=⟨文字列⟩　\ref 命令で参照できるラベルを作成します.
captionpos=⟨t|b⟩　見出しの位置を指定します.

以下の命令はソースコードを目次として出力するときなどの命令です. 適宜再定義してください.

\lstlistoflistings　ソースコード目次を出力します. これは caption を付けたソースコードが出力されます.
\lstlistlistingname　ソースコード目次の見出しです. 標準は 'Listings' です.
\lstlistingname　見出しの前の文字列を指定します. 標準は 'listing' です.

\lstlistlistingname 命令と\lstlistingname は

```
\renewcommand{\lstlistlistingname}{ソースコード目次}
\renewcommand{\lstlistingname}{ソースコード}
```

のように定義しておくと日本語の文書でも違和感がないと思います.

罫線枠に関する設定です.

frame=⟨none|single|shadowbox⟩　ソースコードの周りに表示する罫線枠の設定です.
frame=⟨trblTRBL の部分集合⟩　上（t）・下（b）・左（l）・右（r）の罫線の表示を指定します.

枠を付けて表示する例として

```
\begin{lstlisting}[frame=TBlr,]
main( void ){
   int a=3; int b=5;
   printf("a+b=%d.\n", a+b);
}
\end{lstlisting}
```

と入力するとソースコード 9.1 のようになります.

ソースコード 9.1　枠の調整

```
1 main( void ){
```

```
2     int a=3; int b=5;
3     printf("a+b=%d.\n", a+b);
4 }
```

色に関する設定です.

`backgroundcolor=`⟨色⟩　color パッケージで使われる宣言型の命令`\color` などが使えます.

`rulecolor=`⟨色⟩　罫線枠の色を指定します.

色と付ける例として

```
[frame=single,backgroundcolor={\color[gray]{.85}},
rulecolor={\color[gray]{.5}}]
```

という入力はソースコード 9.2 となります.

ソースコード 9.2　色の調整

```
1 main( void ){
2     int a=3; int b=5;
3     printf("a+b=%d.\n", a+b);
4 }
```

ソースコードは浮動体として宣言できますので

```
\lstinputlisting[language={Prolog},caption=Prolog のソースコードを %
\textsf{Listings}を使って挿入する例,label=src:prolog]{filename.pl}
```

とすれば相互参照も可能になります（ソースコード 9.3）.

ソースコード 9.3　Prolog のソースコードを Listings を使って挿入する例

```
1  % File: prolog.pl
2  % Author: Toru Watanabe
3  % Date: Saturday, Aug. 9th, 2003
4  % Original Source Information:
5  % File: Micro Expert System.
6  % Ian Frank, July 2001, updated May 2002
7  % (Based on original by Alison Cawsey)
8  % Dynamic userfact/1
9  :– dynamic(userfact/1).
10 % Clear userfact/1
11 clear_facts :–
12    retract(userfact(_)),
13    clear_facts.
14 clear_facts :–
```

```
15   write('OK, removed user facts').
16 % NEW OPERATORS
17 :- op(975, fx, if).
18 :- op(950, xfy, then).
19 :- op(925, xfy, and).
20 %% If All people come to X's party, thne X is happy.
21 rule(if all_come_the_party(X) then happy(X) ).
22 qtext(good(X,wed_class) , ['Is ',X,'\'s Wednesday class good?'] ).
23 %% If X is happy, this problem would be succeed.
24 atext(happy(X),['Everyone comes to the party and ',X,' is happy!!']).
25 %
26 prove(Goal) :-
27   bchain(Goal),!, % bchain to check if true.
28   atext(Goal, Text), % get hold of appropriate text.
29   write_list(Text). % write out the recommendation
30 %
31 prove(_) :-
32   write_list(['The goal does not seem to be true.']).
33 %
34 yesno(Text) :-
35   write_list(Text),
36   write_list(['(y/n)']),
37   get(X),
38   X =:= 121.
```

9.19 URL の記述

最近ではウェブ上への参照先を示すために URL と呼ばれるアドレスを書く場合があります．これを LaTeX でやろうと思えば\verb 命令が使えると思うのですが脚注の中で使えないとか，引数の中で使えないという事態に陥ります．このようなときは Donald Arseneau 氏による url を使うと良いでしょう．使い方は\verb 命令とほぼ同じで '%' や '#' などの特殊記号に対して特別な対処をしなくともそのまま記述できます．URL に対しては\url を，パスやファイルを示す場合は\path を使います．e-mail などを表記する場合は新規に\email 命令をを定義します．

```
\newcommand{\email}{\begingroup \urlstyle{rm}\Url}
```

使われるフォントは\urlstyle で指定します．スラッシュやピリオドの位置などで自動的に改行されます．

```
\newcommand\email{\begingroup
   \urlstyle{rm}\Url}
\newcommand\directory{\begingroup
   \urlstyle{tt}\Url}
\url{http://www.server.com/dir/file.html}にアクセスしたら\email{name@server.ac.jp}というメールアドレスがあったから\directory{/usr/local/bin/}のファイルを消した.
```

http://www.server.com/dir/file.html にアクセスしたら name@server.ac.jp というメールアドレスがあったから /usr/local/bin/ のファイルを消した.

9.20　ハイパーリンクの実現

LaTeX でも HTML のように相互参照にハイパーリンクを実現できます．Sebastian Rahtz 氏が作成した hyperref [87] を使うことになると思います．これは\special 命令によってハイパーリンクを実現する HyperTeX を使いやすくしたものと考えて良いです．

LaTeX と HTML を足して 2 で割ったような文法で，次のようなマクロが提供されています．⟨URL⟩ の書き方は HTML とまったく同じで，拡張子や#なども忘れないでください．

\hypertarget{⟨name⟩}{⟨文字列⟩}

\hypertarget は ⟨name⟩ を名札として ⟨文字列⟩ にラベルを付けます．ラベルをつけたものは\href などにより参照できます．

\href{⟨URL⟩}{⟨文字列⟩}

\href はファイルやラベルなどの ⟨URL⟩ を参照します．⟨URL⟩ には

```
\href{http://www.server.co.jp/~user/index.htm}{ここをクリック}
```

のように絶対パスでファイルやサイトを指定することができますし，相対パスで指定することもできます．URL にチルダや特殊な記号があるからといって，特別なこともしなくてもよいようです．アドレス中にピリオドが二つあると拡張子を誤認するかもしれないので，一番最後のピリオド以外をアンダーバーに変えるとどうにかなるかもしれません．

\hyperimage{⟨URL⟩}

\hyperimage は画像ファイルの ⟨URL⟩ を指定し，該当するファイルがあればそれを張り込みます．

9.19 節で紹介した url と同等の\url 命令も使えます．hyperref の\url を使うと自動的にハイパーリンクが作成されます．

9.20.1　パッケージオプション

hyperef はほとんど全てのパッケージオプションを keyval を使って ‘⟨項目⟩=⟨値⟩’ のような指定で設定します．ブール値ならば ⟨値⟩ を省略できます．\usepackage 命令の任意引数に対して ⟨オプション⟩ を渡せばそれが hyperref のパッケージオプションになります．例を示すと，

```
\usepackage[pdfpagemode=FullScreen,colorlinks=true,dvipdfm]{hyperref}
```

のようにすると PDF 文書を開いたときにフルスクリーンで表示し，リンクのある文字の色が変更され，DVI ドライバには dvipdfm を使うという設定になります．hyperref で使用できる主なパッケージオプションを紹介しておきます．特に断りがない限り以下のオプションはブール値であって

```
breaklinks=false
breaklinks=true
```

のように有効（‘true’）か無効（‘false’） で指定できるようになっています．ただ単にそのオプションを指定するとそのオプションが有効になります．

デバイスドライバ　デバイスドライバとしては *dvips*, *ps2pdf*, *latex2html*, *tex4ht*, *pdftex*, *dvipdfm* などが使えます．

用紙サイズ　用紙サイズを *a4paper*，*a5paper*，*b5paper* などで指定してデバイスドライバに応じた設定をします．

ハイパーリンクに関する設定です．

breaklinks　ハイパーリンクが複数行ににまたがっても良いようにします．使用したほうが良いでしょう．

colorlinks　ハイパーリンクがあることを枠ではなく，文字列に色を付けて示します．可読性を考えて使用すると良いでしょう．

PDF しおりに関する設定です．

bookmarks　PDF しおりを作成するようにします．

bookmarksopen　PDF しおりを展開するかどうかの設定です．dvipdfm などはパッケージ側で指定してもプログラム側で対応していないかもしれません．角藤亮氏の配布している dvipdfm と Dvipdfm*x* にはコマンドラインオプション -E によって展開できます．

boookmarksopenlevel=⟨数値⟩　PDF しおりを展開する深さを指定します．これは LaTeX における見出しと同じ ⟨数値⟩ を使いますから表 3.2 を参照してください．

bookmarksnumbered　PDF しおりにおいて見出し番号を含めます．

bookmarkstype=⟨種類⟩　⟨種類⟩ には PDF しおりに使うべき目次情報を指定します．PDF しおりに特別なしおりをすでに別ファイル ⟨*file*⟩.foo に作成しているならば ⟨種類⟩ には ‘foo’ を指定します．

PDF しおりに特殊なコマンドが含まれているときはうまくしおりが作成されませんので\texorpdfstring 命令を使います．

\texorpdfstring{⟨LaTeX での表記⟩}{⟨PDF での表記⟩}

これを使うのは例えば見出し用のコマンドに数式などが含まれるときです．以下のように

```
\section{この\hoge について}
```

とするとうまく行かないので

```
\section{この\textorpdfstring{\hoge}{hoge}について}
```

とするのが良いことになります.

デバイスドライバとしては dvipdfm ではなく Dvipdfm*x* を使うと良いでしょう．その場合はオプションとして ‘dvipdfmx’ ではなく ‘dvipdfm’ を渡します.

```
\usepackage[dvipdfm,bookmarks=true,%
bookmarksnumbered=true,bookmarkstype=toc]{hyperref}
```

上記のようにすると ⟨*file*⟩.toc から PDF しおりが見だし番号付きで PDF に追加できます．この場合ソースコードの先頭で PDF に対して渡すべき文字コードを指定します.

```
\AtBeginDvi{\special{pdf:tounicode 90ms-RKSJ-UCS2}}
%\AtBeginDvi{\special{pdf:tounicode EUC-UCS2}}%NTT jLaTeX
```

私の場合は以下のように設定しています.

```
\AtBeginDvi{\special{pdf:tounicode 90ms-RKSJ-UCS2}}
\usepackage[dvipdfm,bookmarks=true,%
   bookmarkstype=toc,bookmarksnumbered=false,%
   bookmarksopen=true,colorlinks=true,%
   linkcolor=blue,citecolor=blue,filecolor=blue,%
   menucolor=magenta,pagecolor=blue,urlcolor=blue,%
   backref=page]{hyperref}
```

PDF ファイルに対して「文書情報」というものを追加したければ hyperref を読み込んでから

```
%\AtBeginDvi{\special{pdf:tounicode 90ms-RKSJ-UCS2}}
\special{pdf:docinfo <<
  /Author   ( 著者を書く )
  /Title    ( 主題を書く )
  /Subject  ( 副題を書く )
  /Creator  ( どのプログラムを使って PDF を作成したのか )
  /Keywords ( キーワード，複数個，指定，可能)
>>}
```

のような設定をします．丸括弧の中には適当な情報を追加します．これを Dvipdfm*x* で PDF に変換し pdfinfo でその PDF 文書情報を見ることもできます．パッケージオプションに

```
%\AtBeginDvi{\special{pdf:tounicode 90ms-RKSJ-UCS2}}
\usepackage[dvipdfm,pdftitle={主題},%
pdfsubject={副題},pdfauthor={著者},%
pdfkeywords={キーワード}]{hyperref}
```

としても同じことになります.

9.21 原稿の執筆支援

LaTeX の原稿を入力するときにそれを支援する環境がいくつかあります.

9.21.1 入力支援統合環境——YaTeX

有名なプログラムとして GNU Emacs 上で動作する広瀬雄二氏の作成された YaTeX が便利です. 詳しいことは YaTeX の公式ページ

http://www.yatex.org/

から情報を集めてください. YaTeX の主な機能を広瀬氏のマニュアルから転載すると

- タイプセッタやプレヴューアなどの編集画面からの起動.
- カーソル位置によらない固定リジョンの部分タイプセット.
- \includeonly のワンタッチ更新.
- エラー箇所への自動ジャンプ.
- LaTeX コマンドの補完入力.
- 既に入力したテキストを環境やコマンド引数の中に取り込む括り補完.
- セクション区切り入力ときの文書構造アウトライン表示.
- セクションコマンドの一括シフト.
- 補完辞書の学習.
- LaTeX の環境やコマンドに応じたガイド付き引数入力.
- 野鳥にないガイド付き引数入力関数の自動生成.
- LaTeX コマンドの削除/変更.
- ファイル間, \begin–\end 間, \ref–\label 間, \cite——\bibitem ジャンプ.
- 一括コメントアウト/アンコメントアウト.
- アクセント記号/数式環境用コマンド/ギリシャ文字の入力支援.
- tabular/array 環境のカラム位置ガイド.
- 標準的 LaTeX コマンドのオンラインヘルプ.
- ドキュメントのインクルード構造の視覚的表示とバッファ切り替え.

となっております. 使いこなせるともう 2 度と他の環境では LaTeX の原稿を入力する気がおきなくなると思います. 詳細に関しては広瀬氏の書かれたマニュアルがありますのでそちらを参照してください.

同じように GNU Emacs 上で動作する AUCTeX というのもあります. 詳細はご自分で調べてみてください.

GNU Emacs には BibTeX モードという参考文献データベースを編集するための便利なモードも備えています. 原貴弘氏のウェブページ

http://www.eis.t.u-tokyo.ac.jp/~hara/tex/tex.html

などを参考にしてください.

9.21.2　入力支援統合環境——WinShell

Ingo H. de Boer 氏らが作成した WinShell は Windows 上で動作する LaTeX の統合支援環境です.

http://www.winshell.de/

鈴見咲君高氏によって日本語化されました.

http://suzumizaki.at.infoseek.co.jp/winshell/

ただし，更新が中断されていますので旧バージョンの日本語化パッチのみとなります. WinShell は日本語が通るように適切な設定します．その辺は直感でなんとかやってください．最近の WinShell のバージョン 2.5 を使うと OS によっては文字化けを起こしますから，古いものを使ってください.

9.21.3　原稿のコンパイル支援——Make

LaTeX はコンパイル型の言語です．ソースファイルを LaTeX プログラムに渡せば人間が読める媒体に変換します．この LaTeX のソースファイルに対して Make と呼ばれるプログラムを使うことで原稿の再コンパイルを支援してくれます．これは一つのグループになったソースファイルを依存関係をはっきりとさせて，再コンパイルを自動化するプログラムです.

Make は Unix 系 OS 用のプログラムですから，Unix 系 OS でなければ実行させるのが難しいかもしれません．あなたの OS が GNU Linux ならば簡単にインストールできるはずです．Windows の方は Cygwin と呼ばれる Windows 上で動作させることができる擬似 UNIX 環境がありますのでこれを導入してみてください.

端末などから次のように

```
▉ make
```

とすれば

```
make: *** No targets specified and no makefile found. Stop.
```

の表示が出るでしょう．ゲームを始める準備は整っているようです．表示されないならば，近くの詳しい人に聞いてみましょう.

まずは簡単な例を示してから，Make の基本を見てみましょう.

```
all: 結果的に作りたいもの
作りたいもの: それに依存するもの
└──────処理内容
```

上記のソースにおいて '└──────┘' はタブ文字をあらわします．これは非常に重要な点です．Make での変数の定義は単純に

⟨変数⟩=⟨値⟩

の書き方になります．同じ記述が何度も繰り返される場合はそれを変数として定義できます．

```
MENDEX=mendex -d jisyo.dic -g
TEX =platex
DVIPS =dvipsk -Pdl -t a4
DVIPDFM=dvipdfmx -f dlbase14.map -p a4 -V 4 -z 5
```

LaTeX では主となる原稿に対して章ごとに分割されたファイルを\include 命令で読み込んでいることがあるでしょう．この場合 LaTeX でタイプセットした後に生成される DVI ファイル ⟨*file*⟩.dvi は主となる原稿 ⟨*file*⟩.tex と分割されたファイルに依存することになります．LaTeX には中途ファイルである ⟨*file*⟩.aux が使われています．LaTeX 自身もタイプセットを完全なものにするために ⟨*file*⟩.aux が変更されたかどうかを検査します．この ⟨*file*⟩.aux が前回のタイプセットから変更されていない場合はタイプセットが完了したとみなされます．ですから，この依存関係は次のようになります．

⟨*file*⟩.dvi: ⟨*file*⟩.aux ⟨*file*⟩.aux: ⟨*file*⟩.tex └──────┘platex ⟨file⟩

具体的には次のように記述します．

```
FILE=hoge
TEX=platex
all: $(FILE).dvi
$(FILE).dvi: $(FILE).aux
             platex $(FILE)
$(FILE).aux: $(FILE).tex
             platex $(FILE)
```

変数を使う場合は '$' を先頭にし変数名を丸括弧で括ります．上記のような例はちょっとうまく行きません．⟨*file*⟩.dvi を作成するのに毎回タイプセットが行われます．これは ⟨*file*⟩.log に相互参照に関する変更があるかどうかを調べるだけにするほうが良いでしょう．次のような Makefile を作成するともう少しうまく行きます．Makefile 中でシャープ '#' 以降はコメントになります．

```
#主となる原稿
FILE=hoge
TEX=platex
REFGREP=grep "^LaTeX Warning: Label(s) may have changed."
all: $(FILE).dvi
$(FILE).dvi: $(FILE).aux
             (while $(REFGREP)$ $(FILE).log; \
```

```
            do $(TEX) $(FILE); done)
$(FILE).aux: $(FILE).tex
            $(TEX) $(FILE)
```

Make で使われるファイルには ‘Makefile’ という決まった名前を付けます．この ‘Makefile’ をタイプセットしたい ⟨*file*⟩.tex と同じ場所におきます．後は端末などから

```
▌make
```

とすると ⟨*file*⟩.dvi が作成されるので 1 度やってみると良いでしょう．

原稿を作成しているときは ⟨*file*⟩.aux とか ⟨*file*⟩.log や ⟨*file*⟩.toc などが中途ファイルなのでタイプセット後には必要ありませんから，これを一括で削除できると良いでしょう．

```
clean:
       rm -f $(FILE).aux $(FILE).log $(FILE).toc $(FILE).tex~
```

Emacs などを使っているときは ⟨*file*⟩.tex~ も削除してくれると有り難いでしょう．上記の 2 行を先程の Makefile に追加しましょう．

原稿が複数に分割されている場合は ⟨*file1*⟩.tex，⟨*file2*⟩.tex，⟨*file3*⟩.tex，⟨*file4*⟩.tex など複数ありますからこれを次のように定義しておきます．

```
SRC=file1.tex file2.tex file3.tex file4.tex
```

これらのファイルが ⟨*file*⟩.aux に依存することは

```
SRC=file1.tex file2.tex file3.tex file4.tex
$(FILE).aux: $(FILE).tex $(SRC)
            $(TEX) $(FILE)
```

と書くことができます．参考文献データベースが複数ある場合も同じように行いますので

```
REF=biblio1.bib biblio2.bib biblio3.bib
$(FILE).aux: $(FILE).tex $(SRC) $(REF)
            $(TEX) $(FILE)
```

のようにします．

最終的に成形したいファイルが DVI ファイル ⟨*file*⟩.dvi だけではなく PDF ファイル ⟨*file*⟩.pdf や PostScript ファイル ⟨*file*⟩.ps も作成するときは ⟨*file*⟩.dvi に依存させて，成形手順を書きます．普通 ⟨*file*⟩.pdf や ⟨*file*⟩.ps も ⟨*file*⟩.dvi を成形してから作成します．pdfLaTeX などを使っているならば ⟨*file*⟩.tex から ⟨*file*⟩.pdf にいきなり変換されますから，この場合は ⟨*file*⟩.pdf を ⟨*file*⟩.tex に依存させて適切な処理内容を記述します．最終的に ⟨*file*⟩.ps と ⟨*file*⟩.pdf を作成するならば以下のように記述できます．

```
TEX =platex
DVIPS =dvipsk -Pdl -t a4 -o $(FILE).ps
DVIPDFM=dvipdfmx -f dlbase14.map -p a4 -V 4 -z 5 -o $(FILE).pdf
all: $(FILE).pdf
```

```
$(FILE).pdf: $(FILE).dvi
             $(DVIPDFM) $(FILE)
$(FILE).ps:  $(FILE).dvi
             $(DVIPS) $(FILE)
```

おまけとして関係するファイルを一つの TAR 書庫にまとめて gzip で圧縮することを考えましょう．これには書庫を作成するための作業用のディレクトリ$(FILE)src/を作成すると簡単です．マクロパッケージや索引用のスタイルなどの他のファイルは ‘OTHERS’ に忘れずにまとめておきましょう．

```
#SRC=file1.tex file2.tex file3.tex file4.tex
#OTHERS=mymacro.sty
#REF=biblio1.bib biblio2.bib biblio3.bib
#FILE=main
tar: $(FILE).tex $(SRC) $(OTHERS) $(REF) Makefile
        mkdir -p $(FILE)src/
        cp  $(FILE).tex $(SRC) $(OHTERS) $(REF) Makefile $(FILE)src/
        tar czf $(FILE)src.tgz $(FILE)src/
        rm -fr $(FILE)src
```

関係する全てのファイルを$(file)src ディレクトリにコピーします．次のそのディレクトリに対して tar を使って$(file)src.tgz を作成します．処理が終わったならば作業用のディレクトリの中を全て削除して完了です．こうすると$(file)src.tgz がディレクトリ付きの書庫として作成されます．

以上のようなことをレポート・論文作成時にも活用します．少々長いのですが以下の記述を Makefile にまとめます．このファイルは Makefile.gz として私のウェブページに置くことにします．

```
# Title: Makefile
# Date:  2004/03/28
# Name:  Thor Watanabe
# Mail:  thor@tex.dante.jp
# 主となる原稿
FILE =sample
# 分割され，インクルードされているファイル
SRC =#chap1.tex chap2.tex,..., chap<n>.tex
#スタイルファイルやクラスファイルなど
OHTERS =#funthesis.cls mymacro.sty
# 画像などのバイナリファイル
IMG =#hoge.eps capture1.jpg
#文献データベース，あるならば．
REF =biblio.bib
#走らせる TeX プログラム
TEX=platex
# Red Hat の場合
#DVIPS =pdvips -Ppdf
#XDVI =pxdvi
```

```
# GNU Linux の場合
#DVIPS =dvips -Ppdf
#XDVI =xdvik
# Windows の場合
DVIPS =dvipsk -Pdl -t a4
XDVI =dviout
# dvipdfmx
DVIPDF  =dvipdfmx -p a4 -f dlbase14.map -o $(FILE).pdf
# 相互参照の解消のため
REFGREP=grep "^LaTeX Warning: Label(s) may have changed."
# プリンタの設定
PRINTER=//server/printername
# 標準のターゲット
all: $(FILE).ps $(FILE).pdf
printps:  $(FILE).ps
        lpr -P$(PRINTER) $(FILE).ps
printpdf:  $(PSFILE)
        lpr -P$(PRINTER) $(FILE).ps
$(FILE).pdf: $(FILE).dvi
        $(DVIPDF) $(FILE)
$(FILE).ps: $(FILE).dvi
        $(DVIPS) -o $(FILE).ps $(FILE)
$(FILE).dvi: $(FILE).aux $(FILE).bbl
        (while $(REFGREP) $(FILE).log; do $(TEX) $(FILE); done)
$(FILE).bbl: $(REFFILE)
        $(BIBTEX) $(FILE)
$(FILE).aux: $(FILE).tex
        $(TEX) $(FILE)
clean:
        rm -f $(FILE).aux $(FILE).log $(FILE).toc $(FILE).dvi
        rm -f $(FILE).pdf $(FILE).tex~ $(FILE).lof $(FILE).lot
tar:
        mkdir -p $(FILE)
        cp $(SRC) $(OHTERS) $(REF) $(IMG) $(FILE).tex Makefile $(FILE)/
        tar czf $(FILE)src.tgz $(FILE)/
        rm -fr $(FILE)/
```

適宜設定などに関してはご自分の環境やファイル状況に合わせてください.

レポートなどのそれ程大規模ではない文書の場合は次のようにします.

```
FILE=report
#SRC=file1.eps file2.eps
XDVI=xdvi
TEX=platex
all: $(FILE).dvi
$(FILE).dvi: $(FILE).tex
        platex $(FILE) && platex $(FILE) && platx $(FILE)
$(FILE).pdf: $(FILE).dvi
```

```
        dvipdfm -p a4 -o $(FILE).pdf $(FILE)
clean:
        rm $(FILE).aux $(FILE).log $(FILE).tex~
view: $(FILE).dvi
        $(XDVI) $(FILE) &
tar:
        mkdir -p $(FILE)src/
        cp $(FILE).tex Makefile $(SRC) $(FILE)src/
        tar czf $(FILE)src.tgz $(FILE)src/
        rm -fr $(FILE)src/
```

上記のファイルを作成しておけば

```
% make view
```

とすると report.tex の変更に併せてタイプセットをしてから hoge.dvi を表示し，

```
% make tar
```

とするとそのときのレポートに関連するファイルを reportsrc.tgz として書庫化してくれます．そのためにはあらかじめ

```
SRC=file1.eps file2.eps
```

のように関係ファイルを SRC に指定します．何度も何度も

```
% platex report.tex
% dvipdfm reprot.dvi
```

と打ち込むのは今日でグッバイです．昨日までの自分とは違う自分に出会えます，多分，多分です．

9.21.4 latexmk

Make はちょっと大げさなのでもう少し簡単に実行できる John Collins 氏による latexmk を紹介しておきます．Perl スクリプトで書かれていますので，導入する前に Perl をインストールします．まず latexmk を動かすために latexmk.pl に変更を加えます．ただ単に欧文用のプログラムを和文対応にするくらいです．

```
$latex = 'latex';  → $latex = 'platex';
$bibtex = 'bibtex'; → $bibtex = 'jbibtex';
$makeindex = 'makeindex'; → $makeindex = 'mendex';
$dvips = 'dvips'; → $dvips = 'dvipsk';
```

あとは端末などから

```
% latexmk file.tex
```

とすると標準では DVI ファイルが作成されます．BibTeX や MakeIndex を使っていても自動的にそれらの処理が行われます．ただし

```
\bibliography{biblio.bib}
\include{file1.tex}
```

とするよりは拡張子を省略して

```
\bibliography{biblio}
\include{file1}
```

としたほうがうまくいきます.

9.21.5　原稿の版管理——CVS

例えばごく普通にテキストエディッタで文書を作成していたならば,「過去の状態の復元」などのような作業は非常に困難でしょう．これを手動で行うにはテキストの文書であれば過去のファイルを更新したファイルとは別に保存しておかなければなりません．この作業は面倒なのでついつい上書きしがちです．しかし，過去に削除してしまった項目が実は後で必要になるという事態はときたま起こるものです．このような問題を防ぐために版管理（バージョンコントロール）というシステムが必要になります．現在広く使われているのは Unix 系 OS でも Windows でも動作するような，プラットフォームに依存しない版管理プログラムが使われています．Brian Berliner 氏，david d zuhn 氏，Jim Kingdon 氏，Jeff Polk 氏らによる CVS や，GNU Emacs の VC などが有名です．CVS に関してはマニュアルの日本語化がされいますので

http://www.radiofly.to/nishi/cvs/

などを参照してください.

第 10 章
論文のサンプル

今まで様々な情報を提供してきましたが，実際に自分で論文の書式を書き起こすのは大変かもしれません．そこでこの章では卒業研究などで提出する概要レポート，いわゆる中間報告と卒業研究で最終的に提出する卒業論文の例を示します．

学位論文などの書式である文書クラスは大学や学会などから指定されます．当大学の場合は funthesis.cls というファイル名で卒業論文のウェブページにて配布されているものと思います．学会なども同様に独自のクラスファイルを配布していますので，その書式に合わせて書きます．

10.1 中間報告のサンプル

中間報告は当大学の 2003 年度の規定で，2 ページ程度にまとめることになっています．この場合，題名，概要，参考文献，図表などを要領よく整理することが重要になります．そのため中間報告では 2 段組にするにが良いでしょう．2 段組にすると以下のような利点があります．

- 1 段組よりも適切な文字数で改行される．
- 図を取り込むときに\columnwidth を使える．

中間報告のサンプルソースファイルと出力結果をご覧ください．このサンプルに使っている文書クラスは奥村晴彦氏の jsarticle です．サンプルのソースファイル中には注意事項なども書いていますので参考にしてください．

jsarticle を使わずに article や jarticle を使わなければならないならば，概要については表題の下に 1 段組で出力するでしょうから abstract パッケージを使ってみてください．abstract では\twocolumn 命令の任意引数の中で\onecolabstract 命令を使います．

```
\twocolumn[{\maketitle
\begin{onecolabstract}
概要部分
\end{onecolabstract}}]
```

jsarticle を使った例のソースファイルです．

```
\documentclass[twocolumn]{jsarticle}
\columnseprule 0.5pt% 段間の罫線
```

```
\usepackage{epic,eepic,amssymb,amsmath,graphicx,url}
\title{2 段組での中間報告のサンプル}
\author{{\small システム情報科学部 情報アーキテクチャ学科}\\
m1202147 渡辺 徹 \\ 指導教官 未来 太郎}
\date{\today} % \today 命令は文書を作成した日付が代入される
% 本文開始
\begin{document}
\begin{abstract}
論文作成においては\LaTeX{}を使用するのが望ましいが，近年で
は事務処理用の Word がその代わりとなっているように見受けられ
る．今回は，はこだて未来大学においてどの程度 Word や\LaTeX{}
が浸透しているのかを 2003 年度の卒業研究から出てくる中間レポ
ートを参考に統計を取ってみた．結果は予想通り Word 人口が圧倒
的に多かった．また，この中間報告のサンプルの内容は出たら目
であるので，内容的な部分を参考にしてはいけない．
\end{abstract}
\maketitle% 表題
\section{目的}
当大学では卒業研究の中間報告として中間レポートを提出するようになってい
る．各自がどのようなアプリケーションを使っているのかを調査することが今
回の目的である．
\section{方法}
直接研究生にアンケートをとったわけではなく，ウェブページ上で 2003 年 9 月
10 日までに提出されているレポートを調査対象とした．
\section{結果}
提出されているレポートを大まかに調査した結果が表\ref{2bansenji}となる．
これは研究生がどのようなアプリケーションで中間レポートを作成したのかを
調べた結果である．どうしても判別できないものは\yo{その他}の項目に入れてあ
る．レポートの最終形態ではなく，原稿を作成する段階で使ったアプリケーシ
ョンを示している．
\begin{table}[htbp]
\begin{center}
\caption{データの分析結果}\label{2bansenji}
\begin{tabular}{|l|r|r|}
\hline
項目          & 人数（人）& 割合（\%）  \\\hline
Word          &  75 & 45.2  \\
\LaTeX{}      &  26 & 15.6  \\
HTML          &  54 & 32.5  \\
Illustrator &   4 &  2.4  \\
OpenOffice  &   1 &  0.6  \\
その他        &   6 &  3.0  \\\hline
合計          & 166 &  100  \\\hline
\end{tabular}
\end{center}
\end{table}
これらの結果は二次的に入手した情報のため，データに若干の誤りがある．直
接アンケートをとって調べればもっと正確な情報が収集できるが，今回は簡易
```

```
的な形をとった．
\section{考察}
以上の結果から，現在 HTML で作成している人物は Word を使う事になるだろう．
結果があくまで中間報告である事を考えれば，Word 人口がこれから増えること
は明白である．今度の働きかけ次第で当大学の\LaTeX{}人口を増加させること
も可能である．

この現象を天下り的にフーリエ変換で解析する．まず，フーリエ変換で関数
$f(x)$を定義する．この関数$f(x)$は変換のための区間を必要とするので，
区間を$[-L,L]$とする．すると以下の式が定義から導出される．
\begin{eqnarray*}
f(x)& = & \frac{a_0}{2} + \sum^{\infty}_{n=1} \left( a_n \cos
          \frac{n\pi x}{L} + b_n \sin \frac{n\pi x}{L} \right) \\
a_n & = & \frac{1}{L} \int^{L}_{-L} f(u) \cos \frac{n\pi u}{L} du\\
b_n & = & \frac{1}{L} \int^{L}_{-L} f(u) \sin \frac{n\pi u}{L} du
\end{eqnarray*}
よって，次式 (\ref{eq:fourier1}) が新たに得られる．
\begin{eqnarray}
f(x) & = & \frac{1}{2L} \int^{L}_{-L} f(u) du \nonumber\\
     & + & \sum^{\infty}_{n=1} \left[ \frac{1}{L} \int^{L}_{-L}
           f(u) \cos \frac{n\pi x}{L} du \cdot \cos \frac{n\pi x}{L}
           \right. \nonumber \\
     & + & \left. \frac{1}{L} \int^{L}_{-L} f(u) \sin
           \frac{n\pi u }{L}du \cdot \sin \frac{n\pi x}{L} \right]
           \label{eq:fourier1}
\end{eqnarray}
式 (\ref{eq:fourier1}) を\(L\rightarrow\infty\) にしたりしてフーリエ変
換は一般に式 (\ref{eq:fourier2}) のように書き表すことができる．
\begin{equation}
F(\alpha )= \frac{1}{\sqrt{2\pi}} \int^{\infty}_{-\infty}
            f(u) e^{-t\alpha u}du \label{eq:fourier2}
\end{equation}
式 (\ref{eq:fourier2}) を使って今回の結果を解析することは，現段階では非
常に困難であると容易に考察できる．
\section{今後の展望}
今回得られた調査結果を下に Gnuplot でデータをプロットする作業が続くもの
と思われる．また，グラフは主に Gnuplot から挿入するのが望ましいとされる．
Gnuplot から挿入したグラフは図~\ref{fig:sample}となる．
\begin{figure}[htbp]
\begin{center}
%\input{abstgnu.tex} %  ファイルからの読み込み
\fbox{\rule{0pt}{3zw}\rule{3zw}{0pt}}
\caption{picture 環境で描画した図形}\label{fig:sample}
\end{center}
\end{figure}
\nocite{*}
\begin{thebibliography}{10}% 参考文献
\bibitem{latexcompanion}
```

```
   Michel Goossens, Frank Mittelbach, and Alexander Samarin.
   The \LaTeX コンパニオン.
   東京アスキー, 1998.
\bibitem{latexgraphics}
   Michel Goossens, Sebastian Rahtz, and Frank Mittelbach.
   \LaTeX グラフィックスコンパニオン.
   株式会社アスキー, 2000.
\bibitem{bibunsyo}
   奥村晴彦.
   [改訂第 3 版] {\LaTeXe} 美文書作成入門.
   技術評論社, 2004.
\bibitem{platex2e}
   乙部厳己, 江口庄英.
   {\em {p\LaTeXe} for Windows Another Manual Vol.1 Basic Kit 1999}.
   ソフトバンク, 1998.
\bibitem{linuxthesis}
   臼田昭司, 伊藤敏, 井上祥史.
   Linux 論文作成術.
   オーム社, 1999.
\bibitem{metafont}
   Donald~E. Knuth.
   \textsf{METAFONT} ブック.
   アスキー, 1994.
\bibitem{jtexbook}
   Donald~E. Knuth.
   改訂新版{\TeX}ブック.
   アスキー出版局, 1992.
\end{thebibliography}
\end{document}
```

2 段組での中間報告のサンプル

システム情報科学部 情報アーキテクチャ学科
m1202147 渡辺 徹
指導教官 未来 太郎

2004 年 10 月 14 日

概要

論文作成においては LaTeX を使用するのが望ましいが、近年では事務処理用の Word がその代わりとなっているように見受けられる。今回は, はこだて未来大学においてどの程度 Word や LaTeX が浸透しているのかを 2003 年度の卒業研究から出てくる中間レポートを参考に統計を取ってみた。結果は予想通り Word 人口が圧倒的に多かった。また、この中間報告のサンプルの内容は出たら目であるので、内容的な部分を参考にしてはいけない。

1　目的

当大学では卒業研究の中間報告として中間レポートを提出するようになっている。各自がどのようなアプリケーションを使っているのかを調査することが今回の目的である。

2　方法

直接研究生にアンケートをとったわけではなく、ウェブページ上で 2003 年 9 月 10 日までに提出されているレポートを調査対象とした。

3　結果

提出されているレポートを大まかに調査した結果が表 1 となる。これは研究生がどのようなアプリケーションで中間レポートを作成したのかを調べた結果である。どうしても判別できないものは ‘その他’ の項目に入れてある。レポートの最終形態ではなく、原稿を作成する段階で使ったアプリケーションを示している。これらの結果は二次的に入手した情報のため、データに若干の誤りがある。直接アンケートをとって調べればもっと正確な情報が収集できるが、今回は簡易的な形をとった。

表 1　データの分析結果

項目	人数 (人)	割合 (%)
Word	75	45.2
LaTeX	26	15.6
HTML	54	32.5
Illustrator	4	2.4
OpenOffice	1	0.6
その他	6	3.0
合計	166	100

4　考察

以上の結果から、現在 HTML で作成している人物は Word を使う事になるだろう。結果があくまで中間報告である事を考えれば、Word 人口がこれから増えることは明白である。今度の働きかけ次第で当大学の LaTeX 人口を増加させることも可能である。

この現象を天下り的にフーリエ変換で解析する。まず、フーリエ変換で関数 $f(x)$ を定義する。この関数 $f(x)$ は変換のための区間を必要とするので、区間を $[-L, L]$ とする。すると以下の式が定義から

1

導出される。

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty}\left(a_n \cos\frac{n\pi x}{L} + b_n \sin\frac{n\pi x}{L}\right)$$
$$a_n = \frac{1}{L}\int_{-L}^{L} f(u)\cos\frac{n\pi u}{L}du$$
$$b_n = \frac{1}{L}\int_{-L}^{L} f(u)\sin\frac{n\pi u}{L}du$$

よって、次式 (1) が新たに得られる。

$$f(x) = \frac{1}{2L}\int_{-L}^{L} f(u)du + \sum_{n=1}^{\infty}\left[\frac{1}{L}\int_{-L}^{L} f(u)\cos\frac{n\pi x}{L}du \cdot \cos\frac{n\pi x}{L} + \frac{1}{L}\int_{-L}^{L} f(u)\sin\frac{n\pi u}{L}du \cdot \sin\frac{n\pi x}{L}\right] \quad (1)$$

式 (1) を $L \to \infty$ にしたりしてフーリエ変換は一般に式 (2) のように書き表すことができる。

$$F(\alpha) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} f(u)e^{-t\alpha u}du \quad (2)$$

式 (2) を使って今回の結果を解析することは、現段階では非常に困難であると容易に考察できる。

5　今後の展望

今回得られた調査結果を下に Gnuplot でデータをプロットする作業が続くものと思われる。また、グラフは主に Gnuplot から挿入するのが望ましいとされる。Gnuplot から挿入したグラフは図 1 となる。

図 1　picture 環境で描画した図形

参考文献

[1] Michel Goossens, Frank Mittelbach, and Alexander Samarin. The LaTeX コンパニオン. 東京アスキー, 1998.

[2] Michel Goossens, Sebastian Rahtz, and Frank Mittelbach. LaTeX グラフィックスコンパニオン. 株式会社アスキー, 2000.

[3] 奥村晴彦. [改訂第 3 版] LaTeX 2ε 美文書作成入門. 技術評論社, 2004.

[4] 乙部厳己, 江口庄英. *pLaTeX 2ε for Windows Another Manual Vol.1 Basic Kit 1999.* ソフトバンク, 1998.

[5] 臼田昭司, 伊藤敏, 井上祥史. Linux 論文作成術. オーム社, 1999.

[6] Donald E. Knuth. METAFONT ブック. アスキー, 1994.

[7] Donald E. Knuth. 改訂新版 TeX ブック. アスキー出版局, 1992.

10.2　学位論文のサンプル

学位論文などは規模として大きくなるので文書クラスは jreport か jsbook を使うことになります．jsbook の場合にクラスオプションは

```
\documentclass[openany,oneside,11pt]{jsbook}
```

とすると左右起こしをせずに片面印刷で出力されます.

jreport や jsbook で使用できる見出しは

- \chapter
- \section
- \subsection

の三つです. \subsubsection 命令はなるべく使わないほうが良いでしょう. jsarticle 文書クラスで使用できた abstract 環境は使えなくなりますので

```
\chapter*{概要}\addcontentsline{toc}{chapter}{概要}
ここに簡潔に概要を書く.
```

として章立てします.

10.2.1 クラスファイルが提供されている

学位論文などは大学側から文書クラスが提供されることがあります. 当大学の卒業論文の場合は funthesis というクラスファイルが配布されていますのでこれを使うことになります. クラスファイルとして funthesis を使った例を示します. 出力は省略させていただきます.

```
%\documentclass[english]{funthesis}% 本文が英語のとき
\documentclass{funthesis}
\usepackage[dvipdfm]{graphicx}%dvips の場合は'dvips' にする
% 日本語の題名
% 長いときは'\\' で改行
\jtitle{公立はこだて未来大学における卒業論文の
{\LaTeX}クラスファイルの設計に関する考察}
% 論文の英文タイトル
\etitle{Title in English}
% 氏名 (日本語)
\jauthor{未来 太郎}
% 氏名 (英語)
\eauthor{Taro MIRAI}
% 所属学科名
\affiliciation{複雑系アーキテクチャ学科}
% 学籍番号
\studentnumber{1300000}
% 正指導教員名
\advisor{正指導 教員}
% 副指導教員がいる場合はコメントアウトし名前を書く
% 副指導教員がいない場合は, ここは削除しても可
%\coadvisor{副指導 教員}
% 論文提出日
\date{2004/01/31}
% ここから本文の始まり
```

```
\begin{document}
% 表紙
\maketitle
% 英語の概要
\begin{eabstract} Abstract in English. (about 500 words)
\fake{you should write your English abstract in one page. }
% 英文キーワード
\begin{ekeyword}
Keyrods1, Keyword2, Keyword3, Keyword4, Keyword5
\end{ekeyword}
\end{eabstract}
% 和文概要 (2000 字程度)
\begin{jabstract} 日本語の概要を書く．(約 200 字)
\fake{ここに日本語の概要を書きます．}
% 和文キーワード
\begin{jkeyword}
キーワード 1，キーワード 2，キーワード 3，キーワード 4，キーワード 5
\end{jkeyword}
\end{jabstract}
% 目次
\tableofcontents
% 図目次
\listoffigures
% 表目次
\listoftables
\chapter{序論} % 章 (chapter) のタイトル
ここに序論を書きます．
\section{背景} % 節 (section) のタイトル
以下に背景，関連する環境，状況，技術に関する概要を記述．
\chapter{考察}
考察しました．
\section{評価結果}
どっこいどっこいです．
\chapter{結論と今後の展開}
どれもだめでした．
% 以降，付録 (付属資料) であることを示す
\begin{appendix}
\chapter{アルゴリズム}
% 付録その 1(関連資料など) を必要があれば載せる
\section{あるアルゴリズム}
% 付録その 2(関連資料など) を必要があれば載せる
\chapter{ソースコード}
プログラムのソースコードなどを掲載します．
\section{あるソースコード}
何かを処理するあるプログラム\texttt{hoge.cpp}のプログラムを示す．
\begin{verbatim}
int main( void ){ return 0; }
\end{verbatim}
```

```
\fake[40]{{\thehoge}行\par}
% 付録の終わり
\end{appendix}
\chapter*{謝辞}
謝辞を書く．
% 参考文献
\begin{thebibliography}{9}
 \bibitem{ラベル} 著者名．書籍名．出版社，年号．
 \bibitem{MT1999} 未来太郎．未来の未来．どこかの出版，1999．
\end{thebibliography}
\end{document}
```

10.2.2 クラスファイルが提供されていない

もし大学側からクラスが提供されていない場合は自前で作成することになります．しかも大抵の大学は Times 系のフォントを使ってフォントサイズは何々でという細かい指定をしてくるのが普通のようです．親切な教員が作成してくれている場合もあります．とりあえず子供だましですが jsbook を用いた例を紹介します．jreport を使っても良いですが jsbook の方が個人的には良いと感じています．まずはご自分の大学の規定に合わせて jsbook に定義のいくつかに変更を加えます．jsbook そのものに変更を加えるとどこにどのような変更を加えたのかが分からなくなる問題などがありますので，別ファイル mygs.sty に変更したマクロなどをまとめておきます．ファイルの先頭に

```
%% File: mygs.sty
%% Copying : Your Name
%% E-mail  : name@univ.ac.jp
%% Date    : 2004/02/20
\ProvidesPackage{mygs}[2004/03/31 First Family]
```

のようなファイル情報を書き込んでおくとよいでしょう．大抵の機関で Times 系のフォントを指定すると思いますので

```
\RequirePackage{txfonts}
```

の 1 行も必要でしょう．マクロパッケージの中で他のパッケージを必要とする場合は \Requirepacakge 命令を使います．

次にページレイアウトについてです．マージンについても細かい指定をしてくるかもしれませんが，一定の設定方法を紹介しましょう．ページレイアウトで設定できる項目については図 9.1 を見てください．まずは 1 行の字数です．1 行 40 文字であったとすると長さ\textwidth に全角 40 文字の幅（40zw）を指定します．

```
\setlength\textwidth{40zw}
\setlength\fullwidth{\textwidth}%jsbook で必要
```

行数は 40 行と指定されている場合\textheight に 40 行送り分（40\baselineskip）を指定します．

```
\setlength\textheight{40\baselineskip}
```

この程度でも良いと思うのですが,

```
\setlength\hoffset{13\p@}
\setlength\voffset{0\p@}
\setlength\evensidemargin{0\p@}
\setlength\oddsidemargin{\evensidemargin}
\setlength\topmargin{0\p@}
\setlength\headheight{0\p@}
\setlength\headsep{0\p@}
\setlength\marginparwidth{0\p@}
\setlength\marginparpush{0\p@}
\setlength\marginparsep{0\p@}
```

のように設定しても良いでしょう．ここでの\p@は単位 'pt' のことです．マクロの中ではこのような命令を使うと良いそうです．ここでは傍注やヘッダーを出力しないと仮定してほとんどの項目に '0pt' を代入しています.

ヘッダーやフッターは割とシンプルなもので良いと思うので jsbook の場合は

```
\pagestyle{plainfoot}
```

としてフッターの中央にページ番号を出力するようにします．jreport は最初からシンプルな plain というページスタイルになっています．ただし，'– 13 –' のようにダッシュも入れるときは

```
\let\@mkboth\@gobbletwo
\let\@oddhead\@empty
\let\@evenhead\@empty
\def\@oddfoot{\normalfont\hfil-- \thepage\ --\hfil}%
\let\@evenfoot\@oddfoot
\setlength\footskip{2\baselineskip}% 必要に応じて
```

とします．ページ番号を太字にするときは\normalfont に後に\bfseries を追加します.

見出しのフォントの場合は和文はゴシック，欧文は Times Bold としたい場合は jsbook の場合は

```
\renewcommand{\headfont}{\gtfamily\rmfamily\bfseries}
```

のようにしておけば良いでしょう．jsbook は標準では欧文がサンセリフ体になっています．jreport の場合は最初から欧文がボールド体に設定されています.

おまけに目次の深さを決めるカウンタ tocdepth も

```
\setcounter{tocdepth}{2}
```

とすると\subsection まで出力されます.

jreport の場合は見出しの後の字下げが行われないことがありますので

```
\RequirePackage{indentfirst}
```

として indentfirst パッケージを読み込みます.

これらをまとめると自分のマクロパッケージ mygs.sty が出来上がりです.

```
%% File: mygs.sty
%% Copying : Thor Watanabe
%% E-mail  : m1202147@fun.ac.jp
%% Date    : 2004/02/20
\ProvidesPackage{mygs}[2004/02/20 First Family]
\RequirePackage{txfonts}% Times 系のフォントを使う
%\RequirePackage{indentfirst}% jreport は必要
\setlength\textwidth{40zw}%1 行 40 文字
\setlength\fullwidth{\textwidth}%jsbook では必要
\setlength\textheight{40\baselineskip}%1 ページ 40 行
\setlength\hoffset{13\p@}%\p@は 0pt のこと
\setlength\voffset{0\p@}
\setlength\evensidemargin{0\p@}
\setlength\oddsidemargin{\evensidemargin}
\setlength\topmargin{0\p@}
\setlength\headheight{0\p@}
\setlength\headsep{0\p@}
\setlength\marginparwidth{0\p@}
\setlength\marginparpush{0\p@}
\setlength\marginparsep{0\p@}
\setlength\footskip{2\baselineskip}% 必要に応じて
\def\ps@foot{% フッターに `-- ページ番号 --' としたいとき
 \let\@mkboth\@gobbletwo
 \let\@oddhead\@empty
 \let\@evenhead\@empty
 \def\@oddfoot{\normalfont\hfil-- \thepage\ --\hfil}%
 \let\@evenfoot\@oddfoot
}
\pagestyle{plainfoot}%jsbook ならば
%\pagestyle{plain}%jreport ならば
\renewcommand{\headfont}{\normalfont\bfseries}
\setcounter{tocdepth}{2}
```

そのような作業が終わったら自分の論文の主となるソースファイルを書き上げます．用紙は A4 で，フォントサイズは 11 pt，左右起こしはせずに片面印刷というのが一般的だと思いますから

```
\documentclass[a4j,11pt,openany,oneside]{jsbook}
```

のようにします．そして先程作成した mygs.sty を

```
\usepackage{mygs}
```

として読み込みます.

この程度でも良いのですが，表紙もまた細かい指定をされる場合があります．1 から\maketitle を作っても良いのですが，一刻も早く論文を仕上げなければならないときに，命令を定義しては間に合わないかも知れません．そのようなときは断腸の思いで\titlepage 環境を借用して表紙を作ることもできます．例として\maketitle 命令の変更例を紹介します．

```
\renewcommand{\maketitle}{%
\begin{titlepage}
    \let\footnotesize\small
    \let\footnoterule\relax
    \let\footnote\thanks
    \null\vskip2em% ページ上部の空白
    \begin{center}\thispagestyle{empty}%
      {\LARGE\headfont ここに表題を書きます}\par\vskip1.5em
      {\Large\normalfont 未来太郎}\par\vskip2em
      {\small 未来研究学科 \qquad 学籍番号}\par\vskip1em
      {\small 指導教員 \qquad 北海太郎}\par\vskip2em
      {提出日 2004/02/30}\par\vskip1em
      {\Large\headfont English Title}\par\vskip1em
      {\large\rmfamily Your Name}\par\vskip1em
    \end{center}%
    \vfill\null
\end{titlepage}}
```

\vskip とは垂直方向に空きを挿入する命令です．

以上は例ですので先方に規定された通りのレイアウトに適宜変更してください．

付録 A
参考資料

A.1 LaTeX と直接関係のない参考資料

LaTeX を使うには Unix 系 OS のツールや Unix 系 OS の基本的な操作ができることが望ましいと思います．まずは Unix 系 OS の使い方に関する参考書です．

LaTeX を動かすための環境を構築するための技術を解説した資料です．Windows を手放せない方は Cgywin を，フリーにこだわる方は GNU Linux を，飽き足らない人は Unix の書籍を参照ください．

[1] 佐藤竜一, いけだやすし, 野村直. Cygwin+CygwinJE-Windows で動かす UNIX——Cygwin is a UNIX environment for Windows. アスキー, 2003.

▶ 日本語環境でも Cygwin がある程度動作するように追加がなされていますし，大変参考になります．

[2] 市川順一. デスクトップ Linux——Vine Linux 2.6 & OpenOffice.org. ローカス, 2004.

▶ GNU Linux の中で日本語環境に配慮されたディストリビューションである Vine Linux の解説書です．新しいです．事務系ソフトの OpenOffice.org に関する情報もあるそうです．

[3] Jerry Peek, Grace Todino, and John Strang. 入門 Unix オペレーティングシステム　第 5 版. オライリージャパン, 2002. 羽山博訳.

▶ Unix システムの解説書です．

シェル関連の書籍です．本書で「端末」とか「ターミナル」と呼ばれるカーネルを突付くためのプログラムです．コンソールなどとも呼ばれます．Unix 系 OS ユーザは参照したほうが良いでしょう．

[4] Cameron Newham and Bill Rosenblatt. 入門 bash 第 2 版. オライリージャパン, 1998. QUIPU LLC 遠藤美代子訳.

▶ bash に関する良書です．中身はちょっと初心者向けではないかもしれません．

[5] Paul DuBois. 入門 csh & tcsh. オライリージャパン, 2002. 鈴鹿倫之, 福澤康裕訳.

▶ bash ではなく csh の本です．

大規模な文書を作成している方や複数人数で文書を作成している方にお勧めの書籍です．

[6] Andrew Oram and Steve Talbott. make 改訂版. オライリージャパン, 1997. 矢吹道郎監訳. 菊池彰訳.

▶ 再コンパイル支援プログラム make の解説書です．これも初心者向けではないかもしれません．

[7] Jennifer Vesperman. 実用 CVS. オライリージャパン, 2003. 滝沢徹, 牧野祐子訳.
▶ 版管理プログラム CVS の解説書です．

[8] Karl Fogel. CVS――バージョン管理システム――. オーム社, 2000.
▶ 版管理プログラム CVS の入門書です．これはウェブ上に日本語訳のマニュアルが公開されていると思います．

コンピュータで文章を打ち込むためのプログラムをテキストエディッタと呼びます．本冊子では GNU Emacs を推奨します．これはフリーウェアで Windows（Meadow が開発されています）など多くの OS で動作します．Mac にもそれらしきエディタがあります．

[9] Debra Cameron, Bill Rosenblatt, and Eric Raymond. 入門 GNU Emacs 第 2 版. オライリージャパン, 1999. 福崎俊博訳.
▶ テキストエディッタというか何でも屋の GNU Emacs の解説書です．GNU Emacs に関してもウェブ上に日本語化されたマニュアルがあると思われます．ただし数 100 ページに及ぶと思うので，紙に印刷されたものもあると便利でしょう．

[10] 小関吉則. 入門 Meadow/Emacs. オーム社, 2003.
▶ GNU Emacs を Windows 上に移植した Mule の後継の Meadow の入門書です．Emacs の内容も含まれるそうです．Emacs と Meadow も基本的には同じ機能を有していると思われます．

コンピュータがどのように文章を理解するのか，それが分かるとコンピュータに仕事をさせ易くなります．文字列処理やプログラミング言語を使うとさまざまな文章の加工ができるようになります．

[11] Jeffrey E. F. Friedl. 詳説 正規表現 第 2 版. オライリージャパン, 2003. 田和勝訳.
▶ 正規表現は多くのプログラミング言語で使われている言語表現形式です．正規表現は言語学に近い内容かもしれないです．

[12] Dale Dougherty and Arnold Robbins. sed & awk プログラミング 改訂版. オライリージャパン, 1997. 福崎俊博訳.
▶ 古くから使われているテキスト加工プログラム，ストリームエディッタです．ほとんどの Unix 系 OS に導入されている標準的なプログラムです．*The Art Of Awk Programming* という本もあるとかないとか．

[13] Randal L. Schwartz and Tom Phoenix. 初めての Perl 第 3 版. オライリージャパン, 2003. 近藤嘉雪訳.
▶ スクリプト系プログラミング言語 Perl の入門書です．『プログラミング Perl』[14, 15] の 2 冊よりは初心者向けの本です．Perl は CGI やテキスト処理など幅広い分野に応用されている優れた言語です．

[14] Larry Wall, Tom Christiansen, and Jon Orwant. プログラミング Perl 第 3 版 VOLUME 1. オライリージャパン, 2002. 近藤嘉雪訳.

▶ Perl の開発者による公式なマニュアルの 1 巻目です.

[15] Larry Wall, Tom Christiansen, and Jon Orwant. プログラミング Perl 第 3 版 VOLUME 2. オライリージャパン, 2002. 近藤嘉雪訳.

▶ Perl の開発者による公式なマニュアルの 2 巻目です.

[16] 原信一郎. Ruby プログラミング入門. オーム社, まつもとゆきひろ監修.

▶ 国産のスクリプト系オブジェクト指向プログラミング言語 Ruby の入門書です. もちろん日本語の扱いが丁寧だし，Perl に取って代わるかもしれない言語です.

[17] まつもとゆきひろ, 石塚圭樹. オブジェクト指向スクリプト言語 Ruby. アスキー, 1999.

▶ プログラミング初心者向けとは言い難い ruby の本です.

[18] Ken Lunde. CJKV 日中韓越情報処理. オライリージャパン, 2002. 小松章 逆井克己訳.

▶『日本語情報処理』の改訂版として出版された非常に分厚い本です．CJKV 情報処理に関する数少ない良書だと思います.

ウェブに関する技術もあると最新の情報を即座に入手できるなどの利点があります. ウェブは情報の検索にも役立ちます.

[19] Chuck Musciano and Bill Kennedy. HTML & XHTML 第 5 版. オライリージャパン, 2003. 原隆文訳.

▶ HTML を使いこなすというよりは現在の HTML の詳細な仕様書と言った感じです.

[20] Tara Calishain and Rael Dornfest. Google Hacks——プロが使うテクニック & ツール 100 選. オライリージャパン, 2003. 山名早人監訳, 田中裕子訳.

▶ 検索エンジン Google の活用術を紹介した書籍です.

Adobe 社の開発した PostScript と PDF について知ると，ページ記述言語の特性や機能などが分かると思います.

[21] Adobe Systems. PDF リファレンス第 2 版——Adobe Portable Document Format Version 1.3. ピアソンエディケーション, 2001.

▶ 数少ない PDF に関する日本語の技術資料です．Acrobat に関する書籍は沢山ありますが，PDF の規格そのものに言及したものはこれ以外にないと思われます.

[22] ———. PostScript リファレンスマニュアル第 3 版——ASCII 電子出版シリーズ. アスキー, 2001. 桑沢清志訳.

▶ Adobe 社の開発したページ記述言語 PostScript の入門書です.

[23] ———. ページ記述言語 PostScript プログラム・デザイン——電子出版シリーズ. アスキー, 1990. 松村邦仁 アスキー出版技術部訳.

▶ Adobe 社の開発したページ記述言語 PostScript の入門書の次に読むべき書籍です.

描画やプロットなどのグラフィックに関わる書籍です[*1].

[*1] MATLAB とか Octave なんかの情報も載せたほうが良いでしょうか？

[24] 川原稔. gnuplot パーフェクト・マニュアル. ソフトバンク・パブリッシング, 1999.
▶ プロットソフト Gnuplot の解説書です.

[25] 皆本晃弥, 坂上貴之. GIMP/GNUPLOT/Tgif で学ぶグラフィック処理――UNIX グラフィックツール入門――. サイエンス者, 1999.
▶ 実際に読んだことがないので詳細は分かりませんが GIMP，Gnuplot，Tgif を取り扱った書籍です.

[26] 向井領治, 古川泰弘. GIMP エッセンシャルテクニック. オーム社, 2000.
▶ 描画プログラム GIMP の解説書です.

A.2　LaTeX の書籍

LaTeX に関する書籍は多く存在します．その中でも特に読むべきもの，手に入りやすいものを紹介します.

A.2.1　入門書その 1

まずは入門として読むべき良書を紹介します.

[27] 奥村晴彦. [改訂第 3 版] LaTeX 2ε 美文書作成入門. 技術評論社, 2004 .
▶ 通称『奥村本』は定評があります．定期的に改訂がされているので，そのときの最新の情報も入手できます.

[28] Leslie Lamport. 文書処理システム LaTeX 2ε. ピアソン・エデュケーション, 1999. 阿瀬はる美翻訳.
▶ LaTeX の産みの親が執筆した良書．通称 *LaTeX manual* とも呼ばれている．上記の奥村晴彦氏の文献を購入していれば特に困ることもないのだが，LaTeX の基本をがっちりと押さえたい人向け．ただし，日本語環境の情報がないので別の情報が必要になると思います.

[29] Michel Goossens, Frank Mittelbach, and Alexander Samarin. The LaTeX コンパニオン. アスキー, 1998. アスキー書籍編集部監修.
▶ *LaTeX manual* [28] で解説できなかったことの補足説明と LaTeX 2ε で使用できるマクロパッケージの紹介がされています.

[30] Michel Goossens, Sebastian Rahtz, and Frank Mittelbach. LaTeX グラフィックスコンパニオン――TeX と PostScript による図形表現テクニック. アスキー, 2000. 鷺谷好輝翻訳.
▶ 『The LaTeX コンパニオン』[29] で解説できなかったことの補足説明と PostScript 周辺の技術を解説したものです.

[31] Michel Goossens and Sebastian Rahtz. LaTeX Web コンパニオン――TeX と HTML/XML の統合. アスキー, 2001. 鷺谷好輝翻訳.
▶ PDF フォーマットの解説から HTML や XML と TeX をどのように統合するかが解説されています.

[32] 臼田昭司, 伊藤敏, 井上祥史. Linux 論文作成術. オーム社, 1999.
▶ Linux 環境を前提としていますが LaTeX や Gnuplot，Tgif などの解説を含んでいますので重宝すると思います.
[33] 生田誠三. LaTeX 2ε 文典. 朝倉書店, 2000.
▶ LaTeX 2ε ではこんなこともできるのかと感心してしまう 1 冊です．入力と出力が対になった辞典に近いと思います.
[34] 藤田眞作. LaTeX 2ε コマンドブック. ソフトバンクパブリッシング, 2003.
▶ その名の通りコマンドブックです．例題が多すぎる気もしますが，その分学習しながら読み進めることもできます.

A.2.2 入門書その 2

少し古くなったのですが，良書ですので紹介します．これらの書籍を購入しても最近の情報をインターネットなどで収集するのが良いと思います.

[35] 乙部厳己, 江口庄英. *pLaTeX 2ε for Windows Another Manual Vol.1 Basic Kit 1999*. ソフトバンク, 1998.
▶ pLaTeX 2ε の丁寧な解説書です．持っていて損のない本です.
[36] ———. *pLaTeX 2ε for Windows Another Manual Vol.2 Extended Kit*. ソフトバンク, 1997.
▶ 上記で取り上げられなかった内容を紹介しています．Gnuplot や $\mathcal{AMS}$-LaTeX に関する情報もあります.
[37] 江口庄英. *Ghostscript Another Manual*. ソフトバンク, 1997.
▶ Ghostscript に関する数少ない解説書で，貴重な書籍です.
[38] 藤田眞作. LaTeX 2ε 階梯 第 2 版. ピアソン・エデュケーション, 2000.
▶ LaTeX 2ε の基本的な部分から XeTeX やマクロの使用方法，カウンタの使い方にいたるまで，洗練された一冊です．持っていて損のない一冊です.
[39] ———. pLaTeX 2ε 入門・縦横文書術. ピアソン・エデュケーション, 2000.
▶ 上記の書籍を購入した後に縦組みにも興味のある人は購入すると良いでしょう.
[40] 中野賢. 日本語 LaTeX 2ε ブック. アスキー, 1996.
▶ 日本語 TeX の開発者による解説書です．これは良書なのですが，入手は難しいようです.

A.2.3 数学系

入門の書籍を持っていれば困ることはないと思いますが，数式を多用する方は $\mathcal{AMS}$-LaTeX に関する書籍があったほうが便利だと思います.

[41] 嶋田隆司. LaTeX 2ε 数式環境. シイエム・シイ出版部, 2001.

▶ $\mathcal{AMS}$-LaTeX に関する基本的な情報はこの 1 冊あれば良いでしょう．少々間違いがあるのが残念ですが，論文作成には役に立つと思います．

[42] George Gratzer. Math into LaTeX. Springer, 2000.

▶ 英語ですがとても参考になる 1 冊です．CTAN/info/mil/にサンプルがあります．

A.2.4　化学・生化学

化学系では化学構造式や反応式などを書く必要があると思います．そのような図を作成するには藤田眞作氏が作成された XyMTeX を使うのが良いと思います．

[43] 藤田眞作. *XyMTeX: typesetting chemical structural formulas*. 星雲社, 1997.

▶ 英語と日本語の 2 ヶ国語で書かれた書籍で，海外でも広く使われていると思われます．バージョンアップされているので藤田眞作氏のホームページを確認したほうが良いでしょう．

[44] ———. 化学者・生化学者のための LaTeX——パソコンによる論文作成の手引き. 東京化学同人, 1993.

▶ 少し古いので今の事情に従わない部分もあるかもしれません．

A.2.5　マクロやクラスの作成

LaTeX をしばらくやっていると，その内部の機構について知りたくなるときがあるかもしれません．しかし book.cls などの中を見て溜め息をつきたくなる人も多いでしょう．そのようなときはマクロ作成やクラス作成を解説した書籍を参考にすると良いと思います．

[45] ページ・エンタープライゼズ株式会社. LaTeX 2ε【マクロ&クラス】プログラミング基礎編. ページエンタープライゼズ, 2002.

▶ かなりお勧めの書籍です．マクロ&クラス作成に関しては，この本があれば今まで悩んでいたことが解決すると思います．

[46] 吉永徹美. LaTeX 2ε【マクロ&クラス】プログラミング 実践編. ページエンタープライゼズ, 2003.

▶ 上記の続編です．

A.2.6　TeX についての本

LaTeX を追求しているとやはり TeX についても知りたくなります．Donald Knuth 氏や plain TeX に関する書籍を参考にすると良いでしょう．いずれの本も入手が難しくなってきていますので，早めに購入するか古本屋や探してください．

[47] Donald Knuth. 改訂新版 TeX ブック. アスキー, 1992. 斎藤信男監修. 鷺谷好輝翻訳.

▶ TeX の作者が書いた本です．アスキーでは取り扱いを停止している模様です．非常に重要な 1 冊です．

[48] ———. METAFONT ブック. アスキー, 1994. 鷺谷好輝翻訳.

▶ TeX システムに必要だったフォントの作成をするためのプログラムの解説です．これも重要な 1 冊です．

A.2.7 組版全般

組版に関する知識がなければまともな本は作れませんので，以下の書籍を読んでみることをお勧めします．

[49] 日本エディタースクール編集. 文字の組方ルールブック——横組編. 日本エディタースクール出版部, 2001.

▶ 横組における組版規則を要領よくまとめた手軽な良書です．

[50] ———. 文字の組方ルールブック——縦組編. 日本エディタースクール出版部, 2001.

▶ 縦組における組版規則を要領よくまとめた手軽な良書です．

[51] ———. 校正記号の使い方——タテ組・ヨコ組・欧文組. 日本エディタースクール出版部, 1999.

▶ 校正記号の使い方をコンパクトにまとめた良書です．以上の 3 冊は値段も手ごろです．

[52] ———. 新編 出版編集技術 上巻. 日本エディタースクール出版部, 1997.

▶ 内容が少し古いのですが（写植時代の名残が強いので）実技的な部分も多く含んだ出版編集に 関わる人間は持っていて損のない 1 冊と言えます．これはその上巻です．

[53] ———. 新編 出版編集技術 下巻. 日本エディタースクール出版部, 1997.

▶ これは上記の下巻です．

[54] Robert Ritter. *The Oxford Style Manual*. Oxford University Press, 2002.

▶ 欧文組版に関する優れた資料です．組版関係のことに興味が沸いてきたら参照してみてください．こちらは英語圏での標準の組版規則についての解説です．

[55] University of Chicago Press Staff. *The Chicago Manual of Style 15th edition*. Chicago University Press, 2003.

▶ こちらは米語圏での標準の組版規則についての解説です．

A.2.8 入手がちょっと難しい書籍

10 年も経つと入手が難しくなるのが専門書というもので，あるときに買わなければ 2 度と手に入らないかもしれません．

[56] 藤田眞作. LaTeX まくろの八衢. アジソン・ウェスレイ, 1995.

▶ 繰り返し処理やカウンタの使い方などを習得するために是非読んでおきたい本です．

[57] ———. LaTeX 本づくりの八衢. アジソン・ウェスレイ, 1996.
- 和文組版に配慮した本づくりに関した技術を取り扱った本です.

A.2.9　無料の冊子

とにかく無料で済ませたい方はオンラインで公開されている無料の冊子をご自分で印刷して勉強されるのが良いでしょう.

[58] Tobias Oetiker（Hubert partl, Irene Hyna and Elisabeth Schlegl）. *The Not So Short Introduction to LaTeX 2ε—Or LaTeX 2ε in 129 minutes*.
- CTAN/info/lshort/english/
- 通称 lshort と呼ばれる LaTeX の入門書です. これだけ読めば一通り LaTeX が使えるようになると思います.

[59] Tobias Oetiker. LaTeX 2ε への道——83 分 LaTeX 2ε 入門. 野村昌孝訳.
- CTAN/info/lshort/japanese/
- 上記の lshort の日本語訳で通称 jlshort と呼ばれています.

[60] 岩熊哲夫, 古川徹生. LaTeX のマクロやスタイルファイルの利用. 1994.
- http://ea3pch.yz.yamagata-u.ac.jp/member/sumio/tex/tex.htm
- LaTeX209 でのマクロの利用についてですが, 今でも十分参考になると思います.

A.3　ウェブ

近年はウェブ（World Wide Web）というネットワークが広く活用されています. 情報の共有や多様なメディアの共存などができるウェブ上では実に様々な情報が提供されています. しかし, あまりに沢山散在するために, どこをどう探せば良いのかが分かりづらいのも事実です. そのような場合は検索エンジンと呼ばれる無料の検索機構やディレクトリと呼ばれる検索対象を項目ごとに階層的に分類したウェブページなども存在します. これらの中で現在最も有力なサービスを提供しているのが Google という検索エンジンです. Google の URL は

http://www.google.co.jp/

ですからウェブブラウザの標準のページはこのサイトにするのも良いでしょう.

この冊子でも取り上げているマクロやプログラムなどの情報源の多くはウェブから得ています. ‘CTAN’ と書かれているのは例えば http://www.ring.gr.jp/pub/text/CTAN/ などのアドレスに変わります. CTAN とは Comprehensive TeX Arhcive Network の略で TeX に関連するマクロパッケージやプログラムなどを収集しそれを提供するサイトです. 英語版ですが CTAN にはカタログがありますので, そこから自分がしたいことを実現できるパッケージなどを見つけると良いでしょう.

A.3.1 LaTeX

LaTeX や LaTeX 基本マクロ，LaTeX 拡張マクロの情報源です．

[61] Johannes Braams, David Carlisle, Alan Jeffrey, Leslie Lamport, Frank Mittelbach, Chris Rowley, and Rainer Schöpf.*LaTeX 2ε Sources*, 2001.

- `$texmf/tex/latex/base/souce2e.tex` をコンパイルすると作成されます．LaTeX 2ε のソースコード部分です．大変勉強になります．

[62] David Carlisle. *The enumerate package*, 1999.

- `enumerate.dtx` から作成されます．
- LaTeX tools に含まれるパッケージです．番号付きリスト環境の拡張です．

[63] ———. *The longtable package*, 2000.

- `longtable.dtx` から作成されます．
- LaTeX tools に含まれるパッケージです．ページをまたぐ程の大きな表を作成するために使います．

[64] ———. *The ifthen package*, 2001.

- `ifthen.dtx` から作成されます．
- LaTeX tools に含まれるパッケージです．条件分岐などに使えます．

[65] Kent McPherson. *Displaying page layout variables*, 2000.

- `layout.dtx` から作成されます．
- 現在使用中のクラスでのページレイアウトについての情報を出力するためのマクロです．

[66] Frank Mittelbach. *Producing slides with LaTeX 2ε*, 1997.

- `slides.dtx` から作成されます．
- slides クラスについての情報です．

[67] ———. *An environment for multicolumn output*, 2003.

- `multicol.dtx` から作成されます．
- 他段組他段組を実現する multicol パッケージについてです．商用利用にはライセンスが必要という癖のあるマクロです．

[68] ———. *An Extension of the LaTeX theorem environment*, 2003.

- `theorem.dtx` から作成されます．
- LaTeX の `theorem` 環境を拡張した theorem パッケージです．$\mathcal{AMS}$-LaTeX のものよりも高性能かもしれません．

[69] Frank Mittelbach, Denys Duchier, Johannes Braams, Marcin Wolinski, and Mark Wooding. *The docStrip program*, 1999.

- `docstrip.dtx` から作成されます．
- クラス作成者は必読の文書です．DocStrip ユーティリティーについての解説です．

[70] Rainer Schöpf, Bernd Raichle and Chris Rowley. *A New Implementation of LaTeX's `verbatim` and `verbatim*` Environments*, 2001.

- verbatim.dtx から作成されます.
- verbatim 環境の拡張です.

[71] LaTeX3 Project Team. *LaTeX 2ε font selection*, 2000.
- fntguide.tex から作成されます.
- LaTeX でのフォントの選択方法（NFSS）についての文書です．LaTeX で使われているフォントの定義の仕方について知りたいならば読むべきでしょう.

[72] ———. *LaTeX 2ε for authors*, 2001.
- usrguide.tex から作成されます.
- LaTeX を使い始める人は読むべき情報です．LaTeX の更新履歴などが含まれているので重要な文書です.

[73] ———. *LaTeX 2ε for class and package writers*, 1999.
- clsguide.tex から作成されます.

[74] Kresten Thorup, Frank Jensen, and Chris Rowley. *The calc package*, 1998.
- calc.dtx から作成されます.
- LaTeX での四則演算を楽にするパッケージです.

[75] Vlandimir Volovich, Werner Lemberg and LaTeX3 Project Team. *Cyrillic languages support in LaTeX*, 1999.
- cyrguide.tex から作成されます.

A.3.2　LaTeX 周辺の資料

周辺に関する資料です.

[76] Mark Wicks. *Dvipdfm User's Manual*, 1999.
- デバイスドライバ Dvipdfm のマニュアルです.

[77] Cho Jin Hwan. *DVIPDFMx, an eXtension of DVIPDFM*, 2003.
- http://project.ktug.or.kr/dvipdfmx/
- Dvipdfm の拡張版である Dvipdfm*x* についての情報です.

[78] Oren Patashnik. BibTeXing: BibTeX の使い方, 1991. 松井正一訳.
- jBibTeX と共に配布される文書です.

[79] 松井正一. 日本語 BibTeX: jBibTeX, 1991.
- jBibTeX と共に配布される文書です.

[80] Scott Pakin. *How to Package Your LaTeX Package*, 2003.
- 自分が作成した LaTeX のパッケージを配布するためのパッケージの作成方法が書かれた資料です.

[81] Philipp Lehman. *The Font Installation Guide*, 2003.
- CTAN/info/Type1fonts/.
- LaTeX で PostScript フォントを使うための技術資料です.

[82] Karl Berry. *Fontname*, 2003.

- CTAN/info/fontname/.
- LaTeX におけるフォント名についての技術資料です.

[83] TeX Users Group. *A Directory Structure for TeX Files*, 2003.

- CTAN/tds/.
- TeX に関連するファイルが乱雑に分類されていたので，ある基準できちんと整理するように定めた資料です.

A.3.3 マクロパッケージ

[84] Various Artists. *The Lgrind Package*, 2002.

- CTAN/support/lgrind/
- ソースコードを整形する Lgrind パッケージについてです．最近はあまり使われなくなったのでしょうか.

[85] 古川正恵. *jlgrind—grind nice program listings using LaTeX*, 1997.

- http://www.vector.co.jp/authors/tfuruka1/
- 上記の Lgrind を日本語化された古川正恵氏による文書です．日本語化されたのが 1997 年ですから，少々古くなっています.

[86] Carsten Heinz. *The Listings Package*, 2003.

- http://www.atscire.de/.
- 同じくソースコードを整形する listings パッケージについてです．日本でもよく使われているでしょうか.

[87] Sebastian Rahtz. *Hypertext marks in LaTeX the hyperref package*, 1998.

- LaTeX でハイパーリンクを実現するためのマクロ hyperref に関しての資料です.

[88] Keith Reckdahl. *Using Imported Graphics in LaTeX 2ε*, 1997.

- CTAN/info/epslatex.pdf
- 画像を取り込むための graphics（graphicx）パッケージの使い方を丁寧に解説した文書です.

[89] Timothy Zandt. *PSTricks: PostScript macros for Generic TeX*, 1993.

- http://www.tug.org/applications/PSTricks/
- PostScript 命令を使って図形を描く PSTricks についての文書です．情報量が豊富です.

[90] Piet Oostrum. *Page layout in LaTeX*.

- CTAN/macros/latex/contrib/supported/fancyhdr/
- ヘッダーやフッターを調整する fancyhdr についてのマニュアルです.

[91] American Mathematical Society. *User's Guide for amsmath Package*.

- http://www.ams.org/tex/amslatex.html 米国数学会が提供する $\mathcal{AMS}$-LaTeX に含まれる amsmath パッケージにに関する資料です.

[92] Michael Downes. *Short Math Guide for LaTeX*.

- http://www.ams.org/tex/short-math-guide.html

- Michael Downes 氏による数式の書き方と $\mathcal{AMS}$-LaTeX の簡単な紹介をした資料です.

A.4　ウェブページ

インターネットは広大で LaTeX に関する情報がウェブページで公開されています．それらの情報を追いかけるのも良いでしょう.

[A] CTAN: the Comprehensive TeX Archive Network
- http://ctan.org
- TeX とその周辺のツールを納めたサイトです．各国にミラーがあります．日本では RING サーバの http://www.ring.gr.jp/pub/text/CTAN/ などにあります.

[B] 日本語 TeX 情報
- http://oku.edu.mie-u.ac.jp/~okumura/
- 奥村晴彦氏が管理しているページです. 最新情報が手に入ります.

[C] The Publishing TeX
- http://www.ascii.co.jp/pb/ptex/
- 日本語 TeX を開発したアスキー社のウェブページです．日本語 TeX についての情報があります.

[D] W32TeX
- http://www.fsci.fuk.kindai.ac.jp/~kakuto/win32-ptex/
- 角藤亮氏が Windows 用に移植された pTeX をダウンロードできます.

[E] dviout/dviprt 情報
- http://akagi.ms.u-tokyo.ac.jp/index-j.html
- Windows 用の DVI プレビューアを開発している大島利雄氏のホームページ.

[F] MacpTeX とその周辺
- http://macptex.appi.keio.ac.jp/~uchiyama/macptex.html
- 内山孝憲氏が管理されている Macintosh 上で動作する MacpTeX を取り扱ったホームページ.

[G] pLaTeX for Windows
- http://www.grn.mmtr.or.jp/~ohishi/tex/index.html
- 大石守氏のホームページ．LaTeX の周辺の情報も豊富です.

[H] DTP と印刷フォーラム
- http://forum.nifty.com/fdtp/
- @nifty の DTP に関するウェブページで TeX についても解説している．更新が早く，情報も豊富で非常に良いウェブページ.

[I] 藤田眞作 個人ページ
- http://imt.chem.kit.ac.jp/fujita/fujitas/fujita.html
- 藤田眞作氏のホームページ．いわゆる『藤田本』と呼ばれる良書の著者です.

XeTeX の開発もしています.

[J] 乙部厳己（おとべよしき）個人ページ

- http://argent.shinshu-u.ac.jp/otobe/
- 乙部厳己氏のホームページ．いわゆる『乙部本』と呼ばれる良書の著者です.

[K] yama-Ga.com

- http://www.yama-ga.com/
- 山賀正人氏のホームページ．Gnuplot の日本語化など.

[L] 野鳥（YaTeX）

- http://www.yatex.org/
- 野鳥（やちょう）は広瀬雄二氏が開発した GNU Emacs 上で使える拡張 Lisp です．LaTeX 原稿の編集が楽になります.

[M] Excel2LaTeX

- http://plaza19.mbn.or.jp/~Butcher_Bird/
- Excel で作成された表を LaTeX のソースに変換してくれます.

[N] TeX「超」入門

- http://www.nsknet.or.jp/~tony/TeX/texindex.html
- 刀祢宏三郎（とねこうざぶろう）氏のホームページ．@nifty の「DTP と印刷フォーラム」のスタッフも兼ねています.

[O] 初等数学プリント作成マクロ emath

- http://emath.s40.xrea.com/
- 数学のプリントを作成するだけでなく，数学の多方面にも活用できそうなマクロを収めた emath パッケージの配布元です.

[P] ワープロユーザーのための LaTeX 入門

- http://www.klavis.info/texindex.html
- 大友康寛氏が開設している初心者向けの LaTeX の情報．インストールなどはかなり丁寧に画像付きで詳しいです.

[Q] SMALL LaTeX LAB

- http://www.h4.dion.ne.jp/~latexcat/
- マクロ作成の良書 [45, 46] を手がけられた吉永徹美氏のホームページです.

[R] LaTeX 2ε 的

- http://psitau.at.infoseek.co.jp/
- さまざまなマクロを公開している齋藤修三郎氏のホームページです．特に utf/otf はとても便利です.

[S] 熊澤吉紀（くまざわよしき）氏のホームページ

- http://www.biwako.shiga-u.ac.jp/sensei/kumazawa/
- ソースと画像でマクロの使用例がある熊澤吉紀氏のホームページです.

[T] 竹野研究室 Home Page

- http://takeno.iee.niit.ac.jp/~foo/index.html

- latex2html の日本語化や Gnuplot のマニュアルの日本語化などをされている竹野茂治氏のホームページです.

[U] Ghostscript 8.14 + GSview 4.6 の日本語版

- http://auemath.aichi-edu.ac.jp/member/khotta/ghost/
- 題名にとらわれずに pLaTeX 2ε についての情報を提供している堀田耕作氏のホームページです.

[V] LaTeX によるドイツ語・日本語処理

- http://www.lg.fukuoka-u.ac.jp/~ynagata/latex.html
- 福岡大学の永田善久氏が管理されているホームページです．本冊子では多言語処理についてはほとんど扱っておりません．申し訳ないのですがウェブからそれらの情報を集めてみてください．私がせいぜい理解できるのは日本語と英語とアイヌ語の挨拶くらいですから．あとバウムクーヘンとかドッペルゲンガーなどのドイツ語も知ってますけれど，ぜんぜん分からないので.

[W] 日本語 LaTeX による多言語処理

- http://www2.tba.t-com.ne.jp/ing/
- 上記の永田氏のページに比べてこちらの稲垣徹氏のホームページでは日本語 LaTeX 環境での多言語処理を念頭に置かれた解説があります.

付録 B

GNU Free Documentation License

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The **"Document"**, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as **"you"**. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A **"Modified Version"** of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A **"Secondary Section"** is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The **"Invariant Sections"** are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The **"Cover Texts"** are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A **"Transparent"** copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called **"Opaque"**.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The **"Title Page"** means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section **"Entitled XYZ"** means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as **"Acknowledgements"**, **"Dedications"**, **"Endorsements"**, or **"History"**.) To **"Preserve the Title"** of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or non-commercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
D. Preserve all the copyright notices of the Document.
E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form

shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
H. Include an unaltered copy of this License.
I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties–for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections

with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright ©YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

命令索引

括弧内の記号はその命令で出力することができる記号を意味します．括弧書きのない命令でも，記号を出力するものがあります．

数字/記号

A

B

C

D

E

F

G

H

I

J

Q

R

S

T

U

V

W

X

Y

Z

か

環境

せ

わ

索引

数字/記号

A

B

C

D

う

え

お

か

き

く

け

こ

さ

し

す

せ

そ

ひ

ふ

好き好き LaTeX 2ε 初級編

発行日　2004 年 4 月 2 日 第 0.1 版 配布
2004 年 4 月 16 日 第 0.2 版 配布
2004 年 4 月 30 日 第 0.2a 版 配布
2004 年 8 月 5 日 第 0.3 版 配布
2004 年 10 月 14 日 第 0.3b 版 配布
2004 年 12 月 28 日 第 0.3c 版 配布
2005 年 3 月 20 日 第 0.3d 版 配布
編集　渡辺徹