

# はじめに

## 本マニュアルについて

これは、 $\text{T}_{\text{E}}\text{X}$ （テフ or テック）を初めてやる人，また，ちょっとやったことがあるけど軽い数式しか打てない人など，初心者～中級者くらいを対象として書かれた  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X} 2_{\epsilon}$ （ラテフ or ラテック）のざっくりしたマニュアルです。細かいところは飛ばしてとりあえずすぐ使えるようになりたいという人を想定して書いているので，ざっくりしています。また，細かいことは僕もわかっていないことが多いのであしからず。

## $\text{T}_{\text{E}}\text{X}$ について

ざっくり説明すると， $\text{T}_{\text{E}}\text{X}$  はキレイな文書を PDF で作るためのソフトです。文書を作成するなら word でいいじゃないと思いますが， $\text{T}_{\text{E}}\text{X}$  は word よりも細かいところまでキレイに書けますし（特に数式），オープンソースなのでお金の色々を気にしなくて済んだりします。読み方は前述のとおりテフかテックで，普通のアルファベットで書くときは  $\text{TeX}(\text{LaTeX}2_{\epsilon})$  と書きます。

$\text{T}_{\text{E}}\text{X}$  は，普通のテキストエディタ（メモ帳）とプログラミングのハイブリッドみたいなものです。目的は文書作成（テキストエディタ）なのですが，書き方がプログラミングっぽいです。普通の数式を打つだけなら大して難しくないで，プログラミングと聞いてびびらないでくださいね。

以上で前書きを終わりたいと思います。目次の後から具体的な書き方について説明していきます。細かいところは端折ったりもしてるので，わからないところがあったら普通にググってくださいね～。

# 目次

<b>第 1 章 文書を書き始める前に</b>	<b>7</b>
1.1 インストール	7
1.2 使ってみよう	7
1.3 ファイルの保存	7
<b>第 2 章 文書の書き方</b>	<b>9</b>
2.1 基礎知識	9
2.2 改行と半角スペースについて	9
2.3 コマンドについて	10
2.4 文書の構造	11
2.4.1 タイトル	11
2.4.2 本文の構造	11
2.5 文字サイズ	12
2.6 フォント	12
2.7 文字について	13
2.7.1 そのまま出力されない記号	13
2.7.2 そのまま出力する方法	14
2.7.3 特殊文字	14
2.8 コメント	14
2.9 脚注・欄外への書き込み	15
2.10 長さの単位	15
2.11 空白などの調整	16
2.12 罫線と枠	17
2.13 ルビ	18
2.14 圏点	19
<b>第 3 章 環境</b>	<b>20</b>
3.1 環境	20
3.2 quote 環境と文字寄せ	20
3.3 箇条書き	20
3.3.1 itemize 環境	21
3.3.2 enumerate 環境	21
<b>第 4 章 数式</b>	<b>24</b>
4.1 数式の基礎	24
4.2 上付き文字と下付き文字	24

4.3	演算記号	25
4.4	分数	26
4.5	平方根	26
4.6	括弧	26
4.7	ギリシャ文字	27
4.8	矢印	27
4.9	式の上下に付けるもの	28
4.10	数式の書体とフォント	29
4.11	数学関数	30
4.12	数式番号と参照	30
4.13	数式の頭揃え	32
4.14	場合分け	33
4.15	行列	33
<b>第 5 章</b>	<b>図と色</b>	<b>35</b>
5.1	図	35
5.1.1	図の挿入	35
5.1.2	図の自動配置	36
5.1.3	図を左右に配置する	37
5.2	図の回り込み配置	38
5.3	色	39
<b>第 6 章</b>	<b>表</b>	<b>41</b>
6.1	表の書き方	41
6.2	罫線	42
6.3	表のレイアウト	42
6.4	左右のセルをまとめる	43
6.5	上下のセルをまとめる	44
6.6	横幅を指定する	45
6.7	セルの色を変える	46
6.8	ページまたぎのできる表	47
6.9	表に図を挿入する	47
6.10	表の配置	47
<b>第 7 章</b>	<b>相互参照・目次・リンク</b>	<b>49</b>
7.1	相互参照	49
7.2	目次	49
7.3	リンク	50
7.3.1	PDF 内リンク	50
7.3.2	WEB ページへのリンク	50
7.3.3	その他	51
7.3.4	hyperref のオプションと PDF の文書情報	51

<b>第 8 章</b>	<b>ページレイアウトについて</b>	<b>53</b>
8.1	ドキュメントクラス	53
8.2	ドキュメントクラスのオプション	53
8.3	紙面上下の余白	54
8.4	インデント	56
8.5	行送り	56
8.6	字間	57
8.7	ヘッダーとフッター	57
8.7.1	$\text{\TeX}$ 標準のヘッダーとフッター	57
8.7.2	fancyhdr を使う	58
8.8	横置き	59
8.8.1	PDF 全体を横置きにする	59
8.8.2	PDF の一部だけ横置きにする	59
8.8.3	PDF の一部を紙ごと横置きにする	62
<b>第 9 章</b>	<b>欧文フォントについて</b>	<b>64</b>
9.1	欧文フォントの 5 要素	64
9.1.1	エンコーディング	64
9.1.2	ファミリ (書体)	64
9.1.3	シリーズ	65
9.1.4	シェープ	65
9.1.5	サイズ	65
9.2	フォントの変更について	66
<b>第 10 章</b>	<b>和文フォントについて</b>	<b>67</b>
10.1	和文フォントについて	67
10.2	書体の切り替え	67
10.3	和文フォントを変える	67
10.4	pxchfon パッケージを使う	68
10.4.1	基本的な使い方 (単ウェイトの場合)	68
10.4.2	多ウェイトの場合	68
10.4.3	様々なオプション	69
10.4.4	注意点	70
10.5	色々なフォント	71
<b>第 11 章</b>	<b>ファイルの分割と統合</b>	<b>72</b>
11.1	ディレクトリとパス	72
11.1.1	ディレクトリ	72
11.1.2	パス	72
11.2	$\text{\TeX}$ 標準のコマンド	73
11.3	具体例	73
11.4	import パッケージの利用	74
11.5	分割したファイルを実行する	75

11.5.1 docmute パッケージ . . . . .	76
11.5.2 subfiles パッケージ . . . . .	76
11.5.3 standalone パッケージ . . . . .	77
<b>第 12 章 正しくきれいな文書の書き方</b>	<b>79</b>
12.1 出版物の常識 . . . . .	79
12.2 数式中の文字の字体 . . . . .	79
12.2.1 物理量 . . . . .	79
12.2.2 非物理量 . . . . .	80
12.3 単位 . . . . .	80
12.3.1 文字式の単位 . . . . .	80
12.3.2 数値につける単位 . . . . .	80
12.3.3 siunitx パッケージを使う . . . . .	81
12.4 文章中の数式 . . . . .	81
<b>第 13 章 各種パッケージの使い方について</b>	<b>83</b>
13.1 パッケージのインストールとファイルの置き場所 . . . . .	83
13.2 siunitx . . . . .	83
13.2.1 単位自体について . . . . .	83
13.2.2 単位の区切り . . . . .	84
13.2.3 数値の演算 . . . . .	85
13.2.4 その他 . . . . .	85
13.3 文字を回す . . . . .	85
<b>第 14 章 その他の話題</b>	<b>87</b>

# 第1章 文書を書き始める前に

この章では文章を書き始める前の準備について書きたいと思います。

## 1.1 インストール

インターネット経由でインストールする場合は、「TeX インストール」とでも検索すればいい感じのサイトが上の方に出てきます。そのサイトに従ってやってください（僕もこれでやりました）。もし奥村晴彦さんの「 $\LaTeX$  2 $\epsilon$  美文書作成入門」を購入していれば、付属の CD からできるのでその場合はそちらでやってください（それ持ってたらかれじゃなくてそっち読んだ方がいいと思います）。

## 1.2 使ってみよう

TeX のインストールが終わったら、さっそく簡単な文書を作ってみようと思います。定番のあれを出力してみましょう。前節のどちらかの方法で TeX をインストールした場合、TeXworks というデスクトップアプリがインストールされていると思います。これを開いて、以下の表の左側、入力の方を打ってみてください。もちろん半角でお願いします。なお、 $\_$  は半角スペースを表しています。

```
\documentclass{jsarticle}
```

```
\begin{document}
```


Hello, TeX!

```
Hello, \_TeX\_!
```

$$\int dx = x + C$$

```
\[_\int\_dx\_=_x\_+_C\_]
```

```
\end{document}
```

入力しただけでは何も起こりませんね。まだ実行していないからです。実行するには、TeXworks の左上あたりにある「」のボタン（タイプセットボタン）を押してください。上の表の右側のような出力がされたでしょうか？ されなかったときは、スペルミスなどを見直してみてください。ちゃんと出力されたら文書作成成功です。あとはこれがちょっと複雑になるだけなので、あまり身構えなくても大丈夫です。次章でこの例の入力で何故この出力になるのか説明します。

## 1.3 ファイルの保存

TeX ファイルを保存するときはファイル名の最後に .tex をつけてください（拡張子を .tex にしてください）。テキストファイル（～.txt）で作成した文書を TeX のファイルにするときは、「名前を付けて保存」を

し、ファイル指定を「すべてのファイル」にしてから.tex をつけて保存してください。また、ファイル名はアルファベット・数字・アンダーバー（\_）のみでつけるのがよいでしょう。

以上でとりあえず日本語や英語のみの文章を打てるようになりました。次章から実際に文章を書いていきましょう。



## 第2章 文書の書き方

この章からしっかり文書を作成していきます。この章では、数式、表、図を使わない文書を扱います。まずは普通の文章を打てるようになりましょう！

### 2.1 基礎知識

ここではかる〜く必要知識を説明しておこうと思います。細かいことは追々説明しますので、かる〜く読み流してもらえれば十分です。

先ほどの例について、以下で説明を加えていこうと思います。

```
\documentclass{jsarticle}
```

```
\begin{document}
```

```
Hello, \TeX!
```

```
\[\int dx = x + C\]
```

```
\end{document}
```

Hello,  $\text{\TeX}$ !

$$\int dx = x + C$$

まず、コードの書き方についてです。 $\text{\TeX}$ の命令は、キーボードの右上と右下にある $\backslash$ で始めます（windowsでは円マーク¥になります）。 $\backslash$ から始めた部分は、何かの処理（例えば $\text{\TeX}$ という合成文字を出力するなど）をするためのものであり、そのまま出力されるわけではありません。

次に、ドキュメントクラスについてです。入力之初めに $\backslash\text{documentclass}\{\text{jsarticle}\}$ とあります。これは、ドキュメント（文書）のクラス（種類）を指定するものです。基本 $\text{jsarticle}$ で大丈夫なので、おまじないみたいなものだと思ってもらって大丈夫です。最初の行に書くようにしてください。

また、 $\backslash\text{begin}\{\text{document}\}$ と $\backslash\text{end}\{\text{document}\}$ についてですが、これらで挟まれた部分が本文になります。書きたいことはここに書くようにしてください。

最後に、 $\backslash\text{documentclass}\{\text{jsarticle}\}$ と $\backslash\text{begin}\{\text{document}\}$ の間ですが、このことを「プリアンブル」といいます。今回の例では何も書いてありませんが、ページレイアウトなどのオプションはここで設定することが多いです。

### 2.2 改行と半角スペースについて

先の例で1行の文書は書けるようになりました。しかし、改行や改段落の仕方はまだ知りません。改行するときは、改行前の文の最後に $\backslashbackslash$ とつければOKです。また、空白行を作ると、それは改段落とみなさ

れて、次の文章の始めが頭下げ（インデント）されます。改段落はしたいけどインデントはしたくないというときは、その段落の前に`\noindent` とつければ OK です。以下に例を示します。

<code>\documentclass{jsarticle}</code>	
 <code>\begin{document}</code>	
あいうえお	あいうえおかきくけこ
かきくけこ <code>\</code>	さしすせそ
さしすせそ	たちつてと
たちつてと <code>\</code>	なにぬねの
<code>\</code>	
なにぬねの	
 <code>\end{document}</code>	

$\text{\TeX}$  ファイル中では改行や半角スペース、タブは普通には出力されません。普通に改行をしても出力に反映されないので、`\` という命令で改行するのです。

なお、文章中に命令（`\` で始まるもの）を用いるときは後ろに半角スペースを入れるのが安全です。これ以降、プリアンブルに何も書かないときは、`\documentclass{jsarticle}`、`\begin{document}`、`\end{document}` を省略します。また、`\begin{document}` と `\end{document}` の間を「ドキュメント」と呼ぶことにします。

## 2.3 コマンドについて

$\text{\TeX}$  で作る文書には、地の文と組版命令（コマンド）の両方が含まれています。地の文は他のワープロソフト（word など）と変わらないですが、このコマンドというものは  $\text{\TeX}$  に特徴的です。コマンドには色々な種類があり、自分で命令を作ることも可能です。例えば、 $\text{\LaTeX}$  という文字は `\LaTeX` というコマンドによって出力されます。

先ほど書いたように、コマンドの後ろには半角スペースを入れて使うのが安全です。半角スペースを入れずに打ってしまうと、続く文まで命令の一章とみなされエラーしてしまうので注意してください。 $\text{\TeX}$  ではコマンドの後の半角スペースは区切りのために使われ、出力はされません。この仕様により、英語の文章中などに  $\text{\LaTeX}$  と打ちたければ、コマンドの後に半角スペースを出力するコマンド `\_` を使わなければいけません。これを入れずにただスペースを入れただけでは、コマンドの出力と単語の間が詰まってしまうので気を付けて下さい。

さあ `\LaTeX` を使おう！`\`  
 Let's enjoy `\LaTeX` \ command ! `\`  
 Let's enjoy `\LaTeX` command ! `\`

さあ  $\text{\LaTeX}$  を使おう！  
 Let's enjoy  $\text{\LaTeX}$  command !  
 Let's enjoy  $\text{\LaTeX}$ command !

## 2.4 文書の構造

文書には、文書名・著者名・見出し・段落...といった構造があります。この節ではこれらの表示の仕方を扱おうと思います。

### 2.4.1 タイトル

タイトルには、文書名・著者名・日付が含まれます。これらの出力は以下のようにして行います。

```
\title{タイトル}
\author{著者}
\date{\today}
\maketitle
```

タイトル  
著者名  
平成 29 年 10 月 14 日

`\title`, `\author`, `\date` の3つは、`\maketitle` の前にあれば（プリアンブルでも）大丈夫です。また、特にこの順番である必要もありません。実際にこれらの出力をするのは`\maketitle` であるので、これはちゃんとドキュメント中（`\begin{document}` と `\end{document}` の間）に書いてください。また、何もしなければタイトルは1ページ目の上章に出力されますが、タイトルで1ページ使いたいのなら、

```
\documentclass[titlepage]{jsarticle}
```

と、ドキュメントクラスにオプションをつけてください。以下でそれぞれの命令について細かく説明します。

まずタイトルについてですが、長くなり改行が必要な場合、途中で`\\`を入れれば改行できます。

また、著者についてですが、複数名いる場合は`\and`で区切ります。`\and`の後に必ず半角スペースを入れてください。

日付についてですが、`{}`の中に手動で打ち込めばその日付が、`\date`を書かなかつたり`{}`の中を`\today`とすればその文書ファイルを $\text{\LaTeX}$ が処理した日付が出力されます。また、`\date{}`のように、`{}`の中に何も書かなければ日付の出力はなくなります。不要なときはそうしてください。

### 2.4.2 本文の構造

タイトルの書き方がわかったので、今度は文書中の章や節などの書き方を見ていきましょう。

```
\part{これは章}
\section{これは節}
\subsection{これは小節}
\subsubsection{小々節}
\paragraph{段落}
\subparagraph{小段落}
```

## 第I章

## これは章

## 1 これは節

## 1.1 これは小節

## 1.1.1 小々節

タイトルや節の細かい設定（例えば節番号を §4 にしたいなど）は後々ページレイアウトの章で扱うと思います。

## 2.5 文字サイズ

文字サイズを変えることを考えます。文字のサイズは、pt（ポイント）という単位を用いて表すことが多いです。標準を 10pt として色々なサイズがあるので、以下に列挙します。

<code>\tiny</code>	5pt	わーい
<code>\scriptsize</code>	7pt	わーい
<code>\footnotesize</code>	8pt	わーい
<code>\small</code>	9pt	わーい
<code>\normalsize</code>	10pt	わーい
<code>\large</code>	12pt	わーい
<code>\Large</code>	14.4pt	わーい
<code>\LARGE</code>	17.28pt	わーい
<code>\huge</code>	20.74pt	わーい
<code>\Huge</code>	24.88pt	わーい

文書全体の文字サイズを変えたいときは、ドキュメントクラスにオプションをつけて指定します。

```
\documentclass[12pt]{jsarticle}
```

## 2.6 フォント

文字サイズをいじったら今度はフォントをいじりたくなりますね。とりあえず押さえておきたいものだけ下にまとめます。

<code>\textrm{Roman}</code>	<b>Roman</b>	本文（デフォルト）
<code>\textbf{Boldface}</code>	<b>Boldface</b>	太字
<code>\textit{Italic}</code>	<i>Italic</i>	強調（斜体）
<code>\textgt{ゴシック}</code>	<b>ゴシック</b>	見出し

なお、文字サイズは指定したところだけ変更したい場合、その指定する範囲を`{}`で囲みます。そうしないと、それ以降の文字のサイズが変更されたままになってしまうからです。また、これらは併用もできます。

```
{\large 大きい文字} \\
\textgt{太い文字} \\
{\large \textgt{大きい太字}} \\
\textit{Italic 体} \\
\huge こうすると \\
それ以降にも影響が出ます
```

大きい文字

太い文字

大きい太字

*Italic 体*

こうすると

それ以降にも影響が  
出ます

## 2.7 文字について

今まではアルファベットや平仮名、漢字などの文字を扱ってきました。これらの文字は打った通りに出力されるので、特に困ったことはありませんでした。しかし、 $\text{\LaTeX}$ には、そのまま打っても出力できない記号がいくつかあります。また、今までは英語と日本語しか扱っていなかったので問題はありませんでしたが、他の言語（例えばドイツ語やフランス語、スペイン語など）にあってキーボードでは打てない文字はどうしたらよいのでしょうか。この節ではそういった  $\text{\LaTeX}$  における様々な文字について扱いたいと思います。

### 2.7.1 そのまま出力されない記号

先ほど説明したように、 $\text{\LaTeX}$ にはそのまま打つだけでは出力されない文字がいくつかあります。それは、

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

これらの記号です。最後の3つは数式（次章で扱います）中ならそのまま打てますが、地の文では使えません。これらの記号には特殊な役割があり、そのまま打つとその役割を果たそうとしてエラーや出力ミスが起きてしまいます。これらの記号を出力したいときは、記号の前に`\`をつけて、`\#`などとすればOKです。また、半角スペースやタブもそのまま打つだけでは無視されて出力されません。半角スペースを打ちたいときも、同様に`\`をつけて「`\_`」のようにすれば出力されます。なお、全角スペースは1つの文字としてみなされているので、出力されます。

### 2.7.2 そのまま出力する方法

前小節でどの文字が打てないのかはわかりましたので、今度はそれらを文章中に入りたいときにどうしたらいいかを説明しようと思います（もちろんいちいち\をつけても出力はできますが、面倒です）。

打ちたい記号が数文字程度なら、

```
\verb|...|
```

のように、\verb という命令の後の || の間に打ちたい記号を入れれば OK です（同じ文字で挟めばよいので、!や\$などでも大丈夫です）。

打ちたいものが複数行に渡る場合は、それらを\begin{verbatim}と\end{verbatim}で囲えば OK です。

<code>\verb #! </code>	<code>#!</code>
<code>\begin{verbatim}</code>	
<code>\documentclass{jsarticle}</code>	<code>\documentclass{jsarticle}</code>
Hello <code>\TeX</code> !	Hello <code>\TeX</code> !
<code>\end{verbatim}</code>	

なお、半角スペースを示す「`\_`」を出力したいときは、\verb\* や \begin{verbatim\*}, \end{verbatim\*} のように、「\*」をつけてください。

### 2.7.3 特殊文字

TeX では様々な特殊文字をコマンドによって出力できるようにしています。例えば © や §, æ や £ などです。ただ、あんまり使わないという人が多いと思うのでとりあえず省略します。普通にググってください。まとめてあるサイトがあります。

## 2.8 コメント

文書を作成していると、入力した文の中で無視してほしい部分が出てきます。例えば書きかけで出力してほしくない部分とか、その部分を書いた日付とか。TeX では、%を書いた行の%以降の部分は無視され、出力されません。この部分をコメント（注釈）といいます。プログラミングでもよく使いますね。これを書くと改行も無視されますので、それも頭に入れて使ってください。必要に応じてコメントを入れて、編集し直ししやすいように文書を作るのをおすすめします。

あいうえお^^I^^I%ここはコメント かきくけこ	
%2017/5/14 たちつと	あいうえお かきくけこ たちつと
%なにぬねの	

また、例えばよくわからないエラーが出るなどの理由で複数行をコメントアウトしたいときもあるかと思いますが、そのときは、`comment` パッケージの `comment` 環境を使ってみましょう。まずはプリアンブル (`\documentclass` と `\begin{document}` の間) に

```
\usepackage{comment}
```

と書いてください。そして、コメントアウトしたい部分を `comment` 環境に入れて

<pre> ここは出力される。\\ ここも。 \begin{comment} ここは出力されない。 ここも。 なんでも書ける \aaaaaaaaaaaaa エラーもしない。 \end{comment} </pre>	<pre> ここは出力される。 ここも。 </pre>
---	-----------------------------

のようにすれば OK です。ただ、`\end{comment}` の直後に `%` を入れるとエラーするようなので、そこだけは気を付けて下さい（また、`listings` 系とも相性が悪いようです）。

## 2.9 脚注・欄外への書き込み

文書を書いていたら、(文書中に) 注釈を入れたいときもありますよね。この節では脚注と欄外への書き込みの方法を説明します。

まず、脚注ですが、`\footnote{...}` というコマンドを使って、このように<sup>1</sup> つけたいところの直後に書きます (このように `\footnote{わーい}` )。 `\footnote{...}` の後の半角スペースを忘れないようにしてください (もしくは `{}` でくくってください)。

次に欄外の書き込みについてですが、これは `\marginpar{欄外だよ〜}` のように書けば出力できます (ちょっとエラーしてるので省いておきます)。しかし、ドキュメントクラスが `jsarticle` だと欄外部分はあまり広くないので、これを使いたいならページレイアウトをいじるか、ドキュメントクラスを変えるかしてください (どちらも後の章で説明します)。

## 2.10 長さの単位

$\text{\TeX}$  で使える長さの単位は色々ありますが、あまり使わないものは省いてよく使うものだけ列挙しておこうと思います。換算値も一応載せておきますね。

---

<sup>1</sup>わーい

<b>cm</b>	センチメートル	$1\text{ cm} = 10\text{ mm} = 0.3937\text{ in} = 28.452699\text{ pt}$
<b>mm</b>	ミリメートル	
<b>in</b>	インチ	$1\text{ in} = 2.54\text{ cm}$
<b>pt</b>	ポイント	$1\text{ pt} = \frac{1}{72.27}\text{ in} = 0.03515898\text{ cm}$
<b>zw</b>	全角 width	$1\text{ zw} = 9.2469\text{ pt} = 0.32499\text{ cm}$
<b>Q</b>	級	$1\text{ Q} = 0.25\text{ mm}$
<b>H</b>	齒	$1\text{ H} = 0.25\text{ mm}$

また、他に、長さを表すコマンドがいくつかあるのでそれもまとめておきます。真ん中の列に書いているのはデフォルト値です。なお、これらの値は変更可能です。変更の仕方については後々ページレイアウトの章で扱おうと思います。

<code>\baselineskip</code>	12 pt	1 行の縦の長さ
<code>\hsize</code>	453 pt = 49 zw	1 行の横の長さ

ちょっとした補足を加えます。読み飛ばしてもらっても構いません。

まず、pt（ポイント）についてですが、Word などでは 1/72 インチになっているので少しだけサイズが違います。このポイントという単位は、文字の大きさを表す単位として広く使われており、10pt が標準となっています。

また、zw についてですが、これは p $\text{\TeX}$  と up $\text{\TeX}$  でしか使えません（初期設定で pdf $\text{\LaTeX}$  になっていると思うので何もいじっていなければ使えます）。また、上の値は jsarticle でのものであり、jarticle では 9.62216pt となります。

Q と H についてですが、これらは和製の書籍の組版でよく使われる単位です。文字 14Q、行送り 30H というのが大体の標準らしいです。

## 2.11 空白などの調整

$\text{\TeX}$  において入力の際の空白や空白行は無視されてしまいます。しかし、全角スペース（そのまま出力される）を必要数打ち込むのは美しくありません。この節では空白の入れ方を説明します。

なお、下で長さとした部分は、前節で挙げた単位の長さか長さを表すコマンドを入力してください。



<code>\_</code>	半角スペース
<code>~</code>	改行されないスペース
<code>\quad</code>	本文の文字と同じ大きさのスペース (本文 10pt なら 10pt)
<code>\qquad</code>	<code>\quad</code> の 2 倍
<code>\,</code>	<code>\quad</code> の $3/18$ ほど
<code>\&gt;</code>	<code>\quad</code> の $4/18$ ほど (数式モード内のみ)
<code>\;</code>	<code>\quad</code> の $5/18$ ほど (数式モード内のみ)
<code>\!</code>	<code>\quad</code> の $-3/18$ ほど (数式モード内のみ)
<code>\hspace{長さ}</code>	行頭・行末では出力されない水平方向の空白
<code>\hspace*{長さ}</code>	行頭・行末でも出力される水平方向の空白
<code>\vspace{長さ}</code>	行頭・行末では出力されない垂直方向の空白
<code>\vspace*{長さ}</code>	行頭・行末でも出力される垂直方向の空白
<code>\hfill</code>	いくらでも伸びる空白 (弱め)
<code>\hfill</code>	いくらでも伸びる空白 (強め)

適宜これらのスペースをいれて空白の量を調整してください。

また、改行や改段落、改ページの仕方についてですが、一般に改行は`\\`、改段落は空行で行うことが多いです。ただ、明示的にコマンドで行いたいときは、以下の表のようにしてください。なお、インデントとは、段落の最初の頭下げのことです。

<code>\\</code>	改行
<code>\newline</code>	改行
<code>\\*</code>	改行 (改行位置で改ページしない)
<code>\newline*</code>	改行 (改行位置で改ページしない)
空行	改段落 (インデントあり)
<code>\par</code>	改段落 (インデントあり)
<code>\newpage</code>	改ページ
<code>\clearpage</code>	改ページ (図と表を出力してから)
<code>\samepage</code>	改ページしない

特に、環境 (後で説明します) の後の空白行で`\\`を使うとエラーすることがありますので、そのときは`\vspace*`等を使ってください。

また、この「長さ」にはマイナスの値を指定することもできるので、スペースが空きすぎていると思ったらマイナス値を指定してください。

## 2.12 罫線と枠

下線や枠線などの引き方を列挙します。

- 下線

`\underline{...}` で下線が引けます。例えばこんな風に (`\underline{例えばこんな風に}`)

- 罫線

`\hrulefill` で罫線が引けます。

- 点線  
`\dotfill` で罫線が引けます。  
.....
- 枠  
`\fbox{...}` で枠が描けます。例えば `\fbox{こんな風に}`
- 幅指定した枠  
`\framebox[幅][位置]{...}` で幅と文字の位置を指定した枠を描けます。位置は, l,c,r (left, center, right) の頭文字) から選べます。位置を指定しなければデフォルトの center になります。  

わーい
- 長方形  
`\rule[d]{w}{h}` でベースラインからの高さ  $d$ , 幅  $w$ , 縦の長さ  $h$  の長方形が描けます。 $d, w, h$  を調整することで様々な罫線が引けます。この長方形は幅  $w = 0$  にして挿入し、行送りを調整するのによく使われます。

## 2.13 ルビ

$\text{\TeX}$  には様々なパッケージというものがあります。パッケージとは、様々なマクロ（コマンドの定義）をまとめて書いてあるファイルのことです。これを読み込むことで、 $\text{\TeX}$  のコマンドを拡張することができます。この節で扱うルビや、次章で扱う数式を（拡張して）使うときにもパッケージを読み込みます。

ルビを振るために読み込むパッケージは、「pxrubrica」です。これを読み込むには、プリアンブルに

```
\usepackage{pxrubrica}
```

と書けば OK です。

これで準備は終わりました。ここから具体的に使い方を見ていきます。ルビの振り方は、

```
\ruby{親文字}{ルビ文字}
```

です。ここで、ルビ文字には、

```
\ruby{漢字}{かん|じ}
```

のように、親文字の区切りの部分に | を入れます（親文字が 1 文字ならいりません）。

また、漢字全体として読みが与えられている場合は、次のようにオプションに「g」を指定します。

```
\ruby[g]{雲雀}{ひばり}
```

以下に例を示します。

<code>\ruby{漢字}{かん じ} \\\</code>	かん じ 漢字
<code>\ruby{獺}{かわうそ} \\\</code>	かわうそ 獺
<code>\ruby{獺祭}{だっ さい} \\\</code>	だっさい 獺祭
<code>\ruby{獺祭}{かわうそ まつり} \\\</code>	かわうそまつり 獺 祭
<code>\ruby{航空宇宙}{こう くう う ちゅう} \\\</code>	こうくううちゅう 航空宇宙
<code>\ruby[g]{雲雀}{ひばり} \\\</code>	ひばり 雲雀
<code>\ruby[g]{超電磁砲}{レー ル ガン}</code>	レー ル ガン 超電磁砲

これで出力されるのは、デフォルトで設定された正しい日本語のルビの振り方によるルビです。ですが、出力が気に入らなかったときは、様々なオプションでその設定を変えることができます。それについては（不要だと思うので）ここでは扱わないので、必要を感じた人はググってみてください。

また、プログラム言語の `ruby` を読み込むパッケージを使っている場合、`\ruby` という命令はそちらの出力をするものとなってしまいます。この場合を考慮して、ルビを振る命令には、`\jruby` という別名も用意されているので、`ruby` を使っている場合はそちらで対応してください。

なお、この節は「[LaTeX 文書で“美しい日本の”ルビを使う ～pxrubrica パッケージ～](#)」を参考にしました。

## 2.14 圈点

圈点とは文字の上にある強調を示す点のことです。`\.{強}``\.{調}` のようにすれば強調出来ます。もっと点を大きくしたいときは、ルビを使って

```
\ruby{圈点}{\textbullet|\textbullet}
```

のようにすれば強調 できます。

ルビを使わない場合は、マクロを扱う章で扱います（そのうち書きます...）。

以上で数式・表・図などを扱わない平の文書を、文字サイズやフォントの変更も含めて書けるようになりました。次章から平ではない部分の書き方を説明していこうと思います。

## 第3章 環境

先の章までで平の文書は打てるようになりました。この章では、「環境」を使って箇条書きや中央揃え、右寄せなどをしていきます。

### 3.1 環境

まず、環境 (environment) というものについて説明しようと思います。環境とは、`\begin{...}`と`\end{...}`で対になった命令のことを言います。`\begin{○○}... \end{○○}`となっている環境のことを○○環境と言います。環境の内章は外章とは異なることが多く、例えばフォントや文字サイズを環境内で変えても、その環境の外のフォント・文字サイズは変わりません。

では、以下でいくつかの例を見ていきましょう。

### 3.2 `quote` 環境と文字寄せ

`quote` 環境は、文章を引用するときに使うためのものです。文頭にスペースが入ってそれっぽくなります。

```
普通の部分だよ～  
\begin{quote}  
ここは quote 環境の中。\\  
\tiny ここで色々変えても  
\end{quote}  
元に戻る～
```

```
普通の部分だよ～  
  
ここは quote 環境の中。  
ここで色々変えても  
  
元に戻る～
```

また、文字の位置を変え、中央揃えや右寄せにすることもできます。

<code>flushleft</code> 環境	左寄せ
<code>flushright</code> 環境	右寄せ
<code>center</code> 環境	中央揃え

なお、1行なら、これらはそれぞれ、`\raggedright`、`\centering`、`\raggedleft` というコマンドでも実現できます。

### 3.3 箇条書き

また別の環境の例として箇条書きを挙げたいと思います。

### 3.3.1 itemize 環境

普通に「•」などで始まる箇条書きです。

箇条書きといっても

```
\begin{itemize}
\item 記号
\item 番号
\item 見出し
\end{itemize}
```

といったパターンがあります。

箇条書きといっても

- 記号
- 番号
- 見出し

といったパターンがあります。

また、itemize 環境を入れ子にすると、記号が変わっていきます。

```
\begin{itemize}
\item 1 段階目
  \begin{itemize}
\item 2 段階目
  \begin{itemize}
\item 3 段階目
  \begin{itemize}
\item 4 段階目
\end{itemize}
\end{itemize}
\end{itemize}
\end{itemize}
```

- 1 段階目
  - 2 段階目
    - \* 3 段階目
      - 4 段階目

なお、頭の記号を変えることもできます。

この頭の記号を出力する命令は、第 1～4 段階目についてそれぞれ、

```
\labelitemi, \labelitemii, \labelitemiii, \labelitemiv
```

です。これを定義し直せば記号を変えることができます。例えば、1 段階目の記号 • (`\textbullet`) を和文の「・」(中黒)に変えるときは、

```
\renewcommand{\labelitemi}{・}
```

をプリアンブル (`\documentclass` と `\begin{document}` の間) に書けば OK です。

また、一部の記号だけ変えたいときは、その部分の `\item` 命令にオプションをつけて、

```
\item[☆]
```

などのようにしてください。

### 3.3.2 enumerate 環境

これは番号で始まる箇条書きです。

箇条書きといっても	箇条書きといっても
<code>\begin{enumerate}</code>	1. 記号
<code>\item 記号</code>	2. 番号
<code>\item 番号</code>	3. 見出し
<code>\item 見出し</code>	
<code>\end{enumerate}</code>	
といったパターンがあります。	といったパターンがあります。

また、`enumerate` 環境を入れ子にすると、数字の種類が変わっていきます。

<code>\begin{enumerate}</code>	
<code>\item 1 段階目</code>	
<code>\begin{enumerate}</code>	
<code>\item 2 段階目</code>	1. 1 段階目
<code>\begin{enumerate}</code>	
<code>\item 3 段階目</code>	(a) 2 段階目
<code>\begin{enumerate}</code>	
<code>\item 4 段階目</code>	i. 3 段階目
<code>\end{enumerate}</code>	A. 4 段階目
<code>\end{enumerate}</code>	
<code>\end{enumerate}</code>	

こちらでも頭の番号を変えることを考えます。第 1～4 段階目を出力する命令はそれぞれ、

```
\labelenumi, \labelenumii, \labelenumiii, \labelenumiv
```

であるので、これらを定義直せば OK です。具体的に事例を見ていきましょう。第 1 段階の番号を変えることを考えます。その他の段階については、`i` を `ii`, `iii`, `iv` に変えてください。

まず、数字の後のピリオドを取るには、

```
\renewcommand{\labelitemi}{\theenumi}
```

また、数字に `()` をつけるには、

```
\renewcommand{\labelitemi}{(\theenumi)}
```

ローマ数字（小文字）(`i.`, `ii.`, `iii.`, `iv.`) にするには、

```
\renewcommand{\theenumi}{\roman{enumi}}
```

をそれぞれプリアンブルに書いてください。

ローマ数字（小文字）の他にも、算用数字（デフォルト）、英小文字、英大文字、ローマ数字（大文字）といったものを使うこともできます。そのためにはそれぞれ `\arabic`, `\alph`, `\Alph`, `\Roman` を使えばよいです。

また、参考ですが、`enumitem` パッケージを読み込めば、この番号の設定を簡単にすることができます。例えば、

```
\begin{enumerate}[label=例 \arabic*]
```

とすれば、番号の付き方が「例 1, 例 2, 例 3...」となります（空白も反映されます）。このオプション部分の `\arabic*` は、他の `\roman*` などにもすることもできます。その際、`*` を忘れないようにしてください。

他の例ですが，例えば数字を丸で囲んで箇条書きにしたいときは，

```
\begin{enumerate}[label = \textcircled{\scriptsize \theenumi}]  
\item あ  
\item い  
\item う  
\end{enumerate}
```

① あ

② い

③ う

のようにしてください。なお，`\textcircled{\scriptsize 数字}`で，とりあえず簡単ですが丸囲い数字を表現できます。`emath`の`\maru`や`otf`パッケージの`\ajMaru`などもっとキレイな丸はいっぱいあるので，拘る人は是非調べてみて下さい。

## 第4章 数式

この章では数式を扱いたいと思います。数式を打つために  $\text{\TeX}$  を使い始める人が大半だと思いますので、頑張っていきましょう。

### 4.1 数式の基礎

数式は、 $\$$ または $\backslash[ \ ]$ で挟んだ部分に書きます。文章中に数式を入れたいときは、 $\$$ で挟めばよいです。例えば  $y = ax + b$  こんな風に ( $\$y = ax + b\$$ )。

重要な数式は文章とは別で1行使いたいですね。そんなときは $\backslash[ \ ]$ で挟んで、

```
\[ y = ax + b \]
```

のようにすれば、

$$y = ax + b$$

のようになります。

上の例を見ていただけるとわかると思うのですが、数式中に半角スペース（や `tab`）を入れても出力されません（つまり、入れないのと同じです）。見やすくするために適宜半角スペースを入れるのがよいと思います。

また、数式は、AMS（米国数学会）が開発した「`amsmath`, `amssymb`」というパッケージで拡張することができます。とりあえずプリアンブル（`\documentclass` と `\begin{document}` の間）に

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

と書いてください。

### 4.2 上付き文字と下付き文字

指数、例えば  $x^2$  を出力するには、「 $\wedge$ 」を用いて、

```
$ x^2 $
```

とすればよいです。この、「 $^2$ 」のような、文字などの右肩につく文字を「上付き文字」といいます。上付き文字が複数になるときは、`{}`でくくって

```
$ x^{\{n-1\}} $  $\rightarrow x^{n-1}$ 
```

とします。くくらないと変なことになったりするので気を付けて下さい。

$a_n$  のように文字の右下につく文字を「下付き文字」と言います。下付き文字を出力するには、「 $_$ 」を用いて、

```
$ a_n $  $\rightarrow a_n$ 
```

とすればよいです。上付き文字と同様、下付き文字が複数になるときは



`$ a_{n-1} $`  $\rightarrow a_{n-1}$

のようにすればよいです。

下にいくつか例を挙げておきます。なお、`\mathrm` は、デフォルトではイタリック体になってしまう数式中の文字をローマン体にするコマンドです。

<code>a_{ij}</code>	$\rightarrow a_{ij}$
<code>{v_0}^2</code>	$\rightarrow v_0^2$
<code>\mathrm{{}^1{}_1H}</code>	$\rightarrow {}^1_1\mathrm{H}$
<code>a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0</code>	$\rightarrow a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$

### 4.3 演算記号

次は演算記号についてです。`+`, `-`, `=` はキーボードから打ったものをそのまま使えますが、他の色々な記号はコマンド (`\` で始まるもの) を使って出力します。例えば、掛け算記号の  $\times$  は、`\times` というコマンドを使えば出力できます。また、掛け算やベクトルの内積を表す「 $\cdot$ 」は、`\cdot` で出せます。

こういったものを下にまとめておきます。ただ、あまり充実させてはいないので、ここに載っていないものは是非自分で調べてみて下さい。

入力	出力		入力	出力		入力	出力
<code>+</code>	$+$		<code>&lt;</code>	$<$			
<code>-</code>	$-$		<code>\leq</code>	$\leq$		<code>\prime</code>	$'$
<code>\pm</code>	$\pm$	入力	<code>\ll</code>	$\ll$		<code>\emptyset</code>	$\emptyset$
<code>\mp</code>	$\mp$	<code>=</code>	<code>&gt;</code>	$>$		<code>\nabla</code>	$\nabla$
<code>\times</code>	$\times$	<code>\equiv</code>	<code>\geq</code>	$\geq$		<code>\angle</code>	$\angle$
<code>\div</code>	$\div$	<code>\sim</code>	<code>\gg</code>	$\gg$		<code>\triangle</code>	$\triangle$
<code>*</code>	$*$	<code>\simeq</code>	<code>\subset</code>	$\subset$		<code>\forall</code>	$\forall$
<code>\circ</code>	$\circ$	<code>\approx</code>	<code>\subseteq</code>	$\subseteq$		<code>\exists</code>	$\exists$
<code>\bullet</code>	$\bullet$	<code>\propto</code>	<code>\supset</code>	$\supset$		<code>\neg</code>	$\neg$
<code>\cdot</code>	$\cdot$	<code>\perp</code>	<code>\supseteq</code>	$\supseteq$			
<code>\cap</code>	$\cap$	<code>\parallel</code>	<code>\in</code>	$\in$			
<code>\cup</code>	$\cup$		<code>\ni</code>	$\ni$			
<code>\vee</code>	$\vee$						
<code>\wedge</code>	$\wedge$						

`amsmath` パッケージを読み込むともっと多くの記号を使えるようになります。こちらもとりあえず使えそうなものだけ挙げておきます。

入力	出力
<code>\therefore</code>	$\therefore$
<code>\because</code>	$\because$
<code>\leqq</code>	$\leqq$
<code>\geqq</code>	$\geqq$
<code>\fallingdotseq</code>	$\fallingdotseq$

## 4.4 分数

分数は`\frac` というコマンドを使って出力します。使い方は、

```
\frac{分子}{分母}
```

です。

```
\[ \frac{1}{2} \]  
\[ \frac{13}{100} \]  
\[ \frac{1}{1 + x^2} \]
```

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

$$\frac{13}{100}$$

$$\frac{1}{1+x^2}$$

## 4.5 平方根

平方根は`\sqrt` というコマンドで出力できます。使い方は、

```
\sqrt{ルートの中身}
```

です。

```
\[ \sqrt{2} \]  
\[ \sqrt{10} \]  
\[ \sqrt{b^2 - 4ac} \]  
\[ \sqrt{\frac{1}{2}} \]
```

$$\sqrt{2}$$

$$\sqrt{10}$$

$$\sqrt{b^2 - 4ac}$$

$$\sqrt{\frac{1}{2}}$$

## 4.6 括弧

普通に使う括弧としては

```
() , {} , [] , ||
```

の 4 つくらいだと思います。() と [] と || はそのまま打てば出せます。ただ、{} は  $\text{T}_\text{E}\text{X}$  では特殊な働きをするので、そのまま打っても出力されません。{} を出すには、\をつけて\{ \}とします。

普通に () をつけただけでは、中に分数を入れたときなどにサイズが合わず不格好になってしまいます。こういうときのために`\bigl`などのコマンドが用意されていますが、基本は`\left`と`\right`を付けて

```
\left( \right)
```

などとしています。

```
\[ (x - 1)(x - 5) = x^2 - 6x + 5 \]
\[ n \left( \frac{3}{2} R \right) T \]
\[ \left\{ \left( x - \frac{3}{5} \right) + \frac{1}{2} \right\}
```

$$(x-1)(x-5) = x^2 - 6x + 5$$

$$n \left( \frac{3}{2} R \right) T$$

$$\left\{ \left( x - \frac{3}{5} \right) + \frac{1}{2} \right\}$$

## 4.7 ギリシャ文字

ギリシャ文字はコマンドを使って出力します。小文字は英語名 (alpha など) をそのままコマンドにしたもの (つまり \ をつけたもの) で出力できます。大文字は小文字のスペルの 1 文字目を大文字にするだけです。また, *o* (omicron) は英語の *o* (オー) のイタリック体 *o* と同じなので用意されていません。とりあえず下にまとめておきます。

入力	出力	入力	出力	入力	出力	入力	出力
\alpha	$\alpha$	\eta	$\eta$	\nu	$\nu$	\tau	$\tau$
\beta	$\beta$	\theta	$\theta$	\xi	$\xi$	\upsilon	$\upsilon$
\gamma	$\gamma$	\iota	$\iota$	\textit{o}	<i>o</i>	\phi	$\phi$
\delta	$\delta$	\kappa	$\kappa$	\pi	$\pi$	\chi	$\chi$
\epsilon	$\epsilon$	\lambda	$\lambda$	\rho	$\rho$	\psi	$\psi$
\zeta	$\zeta$	\mu	$\mu$	\sigma	$\sigma$	\omega	$\omega$

一部の文字には変体文字が用意されています。

入力	出力	入力	出力	入力	出力
\varepsilon	$\epsilon$	\varphi	$\phi$	\varsigma	$\varsigma$
\vartheta	$\vartheta$	\varrho	$\rho$	\varphi	$\phi$

大文字は以下の 11 個以外はアルファベットと同じです。

入力	出力	入力	出力	入力	出力
\Gamma	$\Gamma$	\Lambda	$\Lambda$	\Psi	$\Psi$
\Delta	$\Delta$	\Xi	$\Xi$	\Omega	$\Omega$
\Theta	$\Theta$	\Pi	$\Pi$		

## 4.8 矢印

矢印も色々出せます。

入力	出力	入力	出力
<code>\uparrow</code>	$\uparrow$	<code>\rightarrow(\to)</code>	$\rightarrow$
<code>\Uparrow</code>	$\Uparrow$	<code>\Rightarrow</code>	$\Rightarrow$
<code>\downarrow</code>	$\downarrow$	<code>\leftarrow(\gets)</code>	$\leftarrow$
<code>\Downarrow</code>	$\Downarrow$	<code>\Leftarrow</code>	$\Leftarrow$
<code>\updownarrow</code>	$\updownarrow$	<code>\leftrightarrow</code>	$\leftrightarrow$
<code>\Updownarrow</code>	$\Updownarrow$	<code>\Leftrightarrow</code>	$\Leftrightarrow$
入力	出力	入力	出力
<code>\longrightarrow</code>	$\longrightarrow$	<code>\mapsto</code>	$\mapsto$
<code>\Longrightarrow</code>	$\Longrightarrow$	<code>\longmapsto</code>	$\longmapsto$
<code>\longleftarrow</code>	$\longleftarrow$	<code>\rightharpoonup</code>	$\rightharpoonup$
<code>\Longleftarrow</code>	$\Longleftarrow$	<code>\rightharpoondown</code>	$\rightharpoondown$
<code>\longlefttrightarrow</code>	$\longleftrightarrow$	<code>\nearrow</code>	$\nearrow$
<code>\Longlefttrightarrow</code>	$\Longleftrightarrow$	<code>\searrow</code>	$\searrow$

## 4.9 式の上下に付けるもの

数式の上下には、矢印や線、点といった色々なものを付けることができます。

入力	出力
<code>\bar{x}</code>	$\bar{x}$
<code>\tilde{x}</code>	$\tilde{x}$
<code>\vec{x}</code>	$\vec{x}$
<code>\dot{x}</code>	$\dot{x}$
<code>\ddot{x}</code>	$\ddot{x}$

また、伸縮するものもあります。

入力	出力	入力	出力
<code>\overline{a + b}</code>	$\overline{a + b}$	<code>\overrightarrow{a + b}</code>	$\overrightarrow{a + b}$
<code>\underline{a + b}</code>	$\underline{a + b}$	<code>\overleftarrow{a + b}</code>	$\overleftarrow{a + b}$
<code>\widehat{abc}</code>	$\widehat{abc}$	<code>\overbrace{abc}</code>	$\overbrace{abc}$
<code>\widetilde{abc}</code>	$\widetilde{abc}$	<code>\underbrace{abc}</code>	$\underbrace{abc}$

`\overbrace` と `\underbrace` については、以下のような使い方もできます。

```
\[
{}_m \mathrm{P}_n =
\overbrace{m \cdot (m-1) \cdot \dots (m-n+1)}^{n \text{ コ}}
\]
\[
{}_m \mathrm{P}_n =
\underbrace{m \cdot (m-1) \cdot \dots (m-n+1)}_{n \text{ コ}}
\]
```

$${}_m P_n = \overbrace{m \cdot (m-1) \cdot \dots (m-n+1)}^{n \text{ コ}}$$

$${}_m P_n = \underbrace{m \cdot (m-1) \cdot \dots (m-n+1)}_{n \text{ コ}}$$

また、矢印の上に文字を書きたいときもあると思います。そのときは、

```
\stackrel{数式}{\rightarrow}
```

を使えばよいです。

```
\[
x \stackrel{f}{\rightarrow} y
\]
U \stackrel{\mathrm{def}}{=} n C_V T
\]
```

$$x \xrightarrow{f} y$$

$$U \stackrel{\mathrm{def}}{=} n C_V T$$

## 4.10 数式の書体とフォント

第 12 章 12.2 節で説明していますが、数式の中にはイタリック体にしないものもあります。しかし、数式モード中では自動でイタリック体になってしまいます。これをアップライト体（立体）に戻すために、`\mathrm` というコマンドが用意されています。また、`amsmath` パッケージの `\text` コマンドを使えば数式中にテキストを入れることができます。

```
\[
x_{\mathrm{max}}
\]
\[
x + \mathrm{Const.}
\]
\[
x \text{ , } \mathrm{cm}^2
\]
\[
v_{\text{水平}}
\]
\[
f(x) = x^2 \text{ (二次関数)}
\]
```

$$x_{\mathrm{max}}$$

$$x + \mathrm{Const.}$$

$$x \text{ , } \mathrm{cm}^2$$

$$v_{\text{水平}}$$

$$f(x) = x^2 \text{ (二次関数)}$$

他にも色々なフォント・字体があるので、一気に下の表にまとめてしまいます。

- 立体 : `\mathrm`
- 筆記体 : `\mathcal` `eucal` パッケージの使用で形状変化
- 太字 : `\bm` `bm` パッケージの使用が必要
- RSFS : `\mathsrc` `mathrsfs` パッケージの使用が必要
- RSFS : `\mathcal` `rsrso` パッケージの使用で、筆記体がマイルドな RSFS 体に
- ドイツ文字 : `\mathfrak`
- 黒板文字 : `\mathbb`

下に例を示します。筆記体ははみ出しました。ごめんなさい。

```

\begin{align*}
&\&
\mathrm{ABCDEFGHIJKLMNOPQRSTUVWXYZ} \\
&\&
ABCDEFGHIJKLMNOPQRSTUVWXYZ \\
&\mathcal{ABCDEFGHIJKLMNOPQRSTUVWXYZ} \\
&\&
\mathcal{ABCDEFGHIJKLMNOPQRSTUVWXYZ} \\
&\&
\bm{ABCDEFGHIJKLMNOPQRSTUVWXYZ} \\
&\&
\bm{ABCDEFGHIJKLMNOPQRSTUVWXYZ} \\
&\mathfrak{ABCDEFGHIJKLMNOPQRSTUVWXYZ} \\
&\&
\mathfrak{ABCDEFGHIJKLMNOPQRSTUVWXYZ} \\
&\mathbb{ABCDEFGHIJKLMNOPQRSTUVWXYZ} \\
&\&
\mathbb{ABCDEFGHIJKLMNOPQRSTUVWXYZ} \\
&\end{align*}

```

`bm` パッケージについてですが、`nextmath` や `mathpazo` 等の数式フォントを設定するパッケージを使用している場合はそれらの後に読み込むようにしてください。

## 4.11 数学関数

`sin` や `cos`, `log` といった関数は、イタリック体ではなく立体で書くことになっています (第 12 章 12.2 節)。これを書くために、それぞれの関数に `\` をつけた命令が用意されています。気を付けてみて下さい。

$$\times \sin x \quad \bigcirc \sin x$$

## 4.12 数式番号と参照

いままで、別行立ての数式は `\[ \]` で出力する、と言っていましたが、他にも色々方法はあります。例えば、`equation` 環境というものがあります。これは、`\[ \]` のように別行立ての数式を出力し、更に数式番号を自動で付けてくれます。下の例では、第 4 章の 1 つめの数式ということで、(4.1) という番号が付けられています。

```

\begin{equation}
y = ax + b
\end{equation}

$$y = ax + b \quad (4.1)$$


\begin{equation*}
y = cx + d
\end{equation*}

$$y = cx + d$$


```

もし数式番号を付けたくないのために、`equation*`環境というものが用意されています。これはほぼ`\[ \]`と同じものになります。

数字でない、例えば`*`のような番号を付けたいときは、`\tag`というコマンドを使って、

```

\begin{equation*}
v = v_0 + at \tag{$*$}
\end{equation*}

$$v = v_0 + at \quad (*)$$


```

のようにします。`()`が不要なときは`\tag*`というコマンドを使えばOKです。

$\text{\TeX}$ では、数式の参照を自動で行うことができます。数式の参照とは、「式 (4.1) において...」などのようなものです。これは、数式にラベルを張り、それを参照する、という形で行います。使い方ですが、

```

\begin{equation}
y = ax^2 + bx + c
\label{eq:parabola}
\end{equation}

$$y = ax^2 + bx + c \quad (4.2)$$


\pageref{eq:parabola}ページの
式~(\ref{eq:parabola}) におい
て,$a=0$のとき...
31 ページの式 (4.2) において,  $a=0$  のとき...

```

のようにします。参照したい式に`\label` コマンドでラベルを貼り、後に使いたいところで`\ref` を用いてその式番号を出力することができます。また、その式のあるページ数を知りたいのために、`\pageref` というコマンドも用意されているので、必要に応じて使ってみてください。また、式の参照には`\eqref` というコマンドが用意されています。これはキレイに参照番号等を出してくれるので、数式の参照においてはこちらを使ってもいいと思います。

なお、第7章7.1節で説明している通りですが、2回実行しないと正しく参照が表示されないなので、そこは注意してください。

## 4.13 数式の頭揃え

式変形を書くときは、数行に渡って = などの位置で揃えて数式を書きたいときもあります。例えば下の様なものです。

$$\begin{aligned} I &= \int_0^{\infty} \frac{1}{1+x^2} dx \\ &= \int_0^{\frac{\pi}{2}} \frac{1}{1+\tan^2 \theta} \cdot \frac{1}{\cos^2 \theta} d\theta \\ &= \int_0^{\frac{\pi}{2}} 1 d\theta = [\theta]_0^{\frac{\pi}{2}} \\ &= \frac{\pi}{2} \end{aligned}$$

これを出力するには、`align` 環境を使います。この環境は `amsmath` パッケージで定義されているので、まずはプリアンプル (`\documentclass` と `\begin{document}` の間) に

```
\usepackage{amsmath}
```

と書いてください。

使い方ですが、揃える位置に `&` を書き、改行する位置に `\\` を書けば OK です。デフォルトでは各行に式番号がついてしましますが、不要なところには `\notag` を付ければその行にはつきません。

```
\begin{align}
I \quad & \& = \int_0^{\infty} \frac{1}{1+x^2} dx \\
& \& = \int_0^{\frac{\pi}{2}} \frac{1}{1+\tan^2 \theta} \cdot \frac{1}{\cos^2 \theta} d\theta \\
& \& = \int_0^{\frac{\pi}{2}} 1 d\theta = [\theta]_0^{\frac{\pi}{2}} \\
& \& = \frac{\pi}{2} \\
\end{align}
```

$$I = \int_0^{\infty} \frac{1}{1+x^2} dx \tag{4.3}$$

$$= \int_0^{\frac{\pi}{2}} \frac{1}{1+\tan^2 \theta} \cdot \frac{1}{\cos^2 \theta} d\theta \tag{4.4}$$

$$= \int_0^{\frac{\pi}{2}} 1 d\theta = [\theta]_0^{\frac{\pi}{2}} \\ = \frac{\pi}{2} \tag{4.5}$$

数式番号が不要ならば `align*` 環境を用いて下さい。また、`align` 環境は表と同じように何列でも揃えられるようになっています。例えば、次のように複数の数式を揃えられるということです。



```
\begin{align*}
x &= r \cos\theta \cos\varphi, & y &= r \cos\theta \sin\varphi, & z &= r \\
&\sin\theta \\
r &= a, & \theta &= \frac{\pi}{2}, & \varphi &= \frac{\pi}{3} \\
\end{align*}
```

$$\begin{aligned} x &= r \cos \theta \cos \varphi, & y &= r \cos \theta \sin \varphi, & z &= r \sin \theta \\ r &= a, & \theta &= \frac{\pi}{2}, & \varphi &= \frac{\pi}{3} \end{aligned}$$

&で揃え、\\で改行するのは、第 6 章で扱う表と全く同じなので、 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  に慣れてくれば自然と行えるようになると思います。

align 環境では数式各行に式番号が振られますが、複数行の数式全体に 1 つの番号を振りたいときは、equation 環境と split 環境を使って、

```
\begin{equation}
\begin{split}
e^x &= \sum_{n=0}^{\infty} \frac{1}{n!} x^n \\
&= 1 + x + \frac{1}{2!} x^2 \\
&+ \frac{1}{3!} x^3 + \dots
\end{split}
\end{equation}
```

$$\begin{aligned} e^x &= \sum_{n=0}^{\infty} \frac{1}{n!} x^n \\ &= 1 + x + \frac{1}{2!} x^2 + \frac{1}{3!} x^3 + \dots \end{aligned} \quad (4.6)$$

他にも、\fbox を使うときに便利な aligned 環境、数式間の空白を自分で制御できる alignat 環境、1 つの数式が複数行に渡るときに使える multiline 環境などがあります。また、align 環境の途中で一旦テキスト（すなわち、一般に、など）を入れてからもう一度数式に戻りたいときには\intertext{テキスト}というコマンドが便利です。ここに書いてあることで足りないときは是非調べてみてください。

## 4.14 場合分け

場合分けは cases 環境を用いて出力できます。

```
\[
| x | =
\begin{cases}
x & (x \leq 0) \\
-x & (x < 0)
\end{cases}
\]
```

$$|x| = \begin{cases} x & (x \leq 0) \\ -x & (x < 0) \end{cases}$$

## 4.15 行列

amsmath パッケージで、数種類の行列用の環境が定義されています。

```

\[
\begin{matrix}
a & b \\
c & d
\end{matrix}
\\
\begin{pmatrix}
a & b \\
c & d
\end{pmatrix}
\\
\begin{bmatrix}
a & b \\
c & d
\end{bmatrix}
\\
\begin{Bmatrix}
a & b \\
c & d
\end{Bmatrix}
\\
\begin{vmatrix}
a & b \\
c & d
\end{vmatrix}
\\
\begin{Vmatrix}
a & b \\
c & d
\end{Vmatrix}

```

$$\begin{matrix} a & b \\ c & d \end{matrix}$$

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

$$\begin{Bmatrix} a & b \\ c & d \end{Bmatrix}$$

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix}$$

$$\begin{Vmatrix} a & b \\ c & d \end{Vmatrix}$$

表や `align` 環境と同じで、列の区切りを`&`、行の区切りを`\\`で行います。  
また、

```
\hdotsfor{列数}
```

というコマンドで、複数列に渡る点々を書くことができます。

```

\[
A =
\begin{pmatrix}
a_{11} & \dots & a_{1n} \\
\hdotsfor{3} \\
a_{m1} & \dots & a_{mn}
\end{pmatrix}

```

$$A = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \dots & \dots & \dots \\ a_{m1} & \dots & a_{mn} \end{pmatrix}$$

## 第5章 図と色

この章では図の挿入と自動配置の仕方、そして色について説明したいと思います。

### 5.1 図

他のソフトを使わず、つまり  $\text{\LaTeX}$  だけでも、例えば `TikZ` というパッケージを使えば図を描くことができます。しかし、これは全てコマンドで描かねばならず、面倒でかつ技術や知識も必要です。

そこで、Excel や PowerPoint、または手書きなどで図を用意することが多いと思います。そうすると、ほとんどの画像が PDF か JPEG で用意されると思うので、それを想定して書いています（おそらく他の形式の図にも通用すると思います）。

#### 5.1.1 図の挿入

図の挿入には `graphicx` パッケージを使います。この `graphicx` パッケージは、グラフィック関係全般で使えるパッケージですので、読み込んでおいて損はないです。図を挿入したいときは以下のようにします。

```
\documentclass[dvipdfmx]{jsarticle}
\usepackage{graphicx}
\begin{document}

\includegraphics[オプション]{図のファイル名}

\end{document}
```

`\documentclass` のオプションに `dvipdfmx` というものがあります。この `dvipdfmx` は、ドライバ名で、 $\text{p}\text{\TeX}$ 、 $\text{up}\text{\TeX}$  で標準的に使われるドライバです。PDF、JPEG、PNG、EPS 形式の図を扱うことができます。他に `pdftex`、`xetex`、`luatex` などがありますが、`dvipdfmx` にしておいて大丈夫です。

図の挿入自体には、

```
\includegraphics[オプション]{図のファイル名}
```

というコマンドを使います。

ファイル名には全角文字は使わず、半角アルファベットと数字とアンダーバーだけにした方が安全です。一応、`grffile` というパッケージを使えば大丈夫みたいですけど。

図のファイルは文書と同じところにおけば大丈夫ですが、図だけ別で管理したいという場合、

```
\graphicspath{{フォルダ1のパス}{フォルダ2のパス}...}
```

とすれば、フォルダ1、フォルダ2... の中も探してくれます。また、個々のコマンドでファイル名を絶対パスや相対パスで指定することもできます。パスの話が分からない人は第11章の第11.1節を読んでください。

`\includegraphics` につけるオプションには以下のようなものがあります。

<code>width=長さ</code>	幅を指定した長さにします。
<code>height=長さ</code>	高さを指定した長さにします。
<code>totalheight=長さ</code>	高さ+深さが指定できます。図を回転したときに便利です。
<code>keepaspectratio</code>	<code>width</code> と <code>height</code> を両方指定したとき、縦横比を変えずに指定した幅、高さに図を入れます。
<code>scale=倍率</code>	画像のサイズが倍率倍されます。
<code>clip</code>	描画領域の外側を描かないという指定です。周囲の文書が侵食されるのを防ぐために指定します。これは指定しておくのが安全です。
<code>trim = x<sub>1</sub> y<sub>1</sub> x<sub>2</sub> y<sub>2</sub></code>	左 $x_1$ 、下 $y_1$ 、右 $x_2$ 、上 $y_2$ だけ図を切り詰めます。単位は 1/72 インチです。
<code>viewpoint= x<sub>1</sub> y<sub>1</sub> x<sub>2</sub> y<sub>2</sub></code>	元の画像の左下の角を原点として、左下の角を $(x_1, y_1)$ 、右上の角を $(x_2, y_2)$ とした長方形の領域を切り出します。こちらも単位は 1/72 インチです。
<code>angle=角度（単位は度）</code>	画像を指定した角度だけ回転します。
<code>origin=位置</code>	画像の回転の中心を指定します。この位置には、 <code>lrcbtB</code> があり、それぞれ左、右、中央、上、下、ベースラインを表しています。この位置の中から 2 つまで選んで指定します。また、 <code>[x=2mm, y=3mm]</code> のように座標で指定することもできます
<code>units=6.2832</code>	回転の角度の単位を、度からラジアンに変更します。

図を入れたいところが何かしらの環境の中だったりするとエラーを吐くことがあります。そのときは、`minipage` 環境を作ってその中で `\includegraphics` をすれば大抵の場合は大丈夫です。この `minipage` 環境は横幅を引数に取り、

```
\begin{minipage}{0.4\hsize}
\centering
\includegraphics{picture.pdf}
\end{minipage}
```

などのようにして使います。この `\hsize` はページの横幅を設定しているコマンドですので、`0.4\hsize` は全ページ幅の 4 割を表しています。

### 5.1.2 図の自動配置

図の自動配置には `figure` 環境を使います。また、図にはキャプションとラベルをつけることができます。例えば以下のように、

```
\begin{figure}
\centering
\includegraphics[width=10cm]{parabola.pdf}
\caption{関数  $y = x^2$  のグラフ}
\label{fig:parabolagraph}
\end{figure}
```

とすれば、下にキャプションのついたグラフが自動的に配置されて出力されます。この図を参照するときは、`\ref{fig:parabolagraph}` において... のようにして 2 回出力すればよいです。この `\label` や `\ref`、2 回実行する意味については相互参照やリンクを扱う第 7 章で詳しく扱います。

この figure 環境には以下のようなオプションがあります。

- t ページ上部に図を出力します。
- b ページ下部に図を出力します。
- p 単独ページに図を出力します。
- h 出来ればその位置に図を出力します。
- H 必ずその位置に図を出力します (float パッケージが必要)。

なお、このオプションはデフォルトでは [tbp] になっています。

図の番号は自動的に「図 1」, 「図 2」, ... のように付きます。これを「Fig.1」のようにしたければ, jsarticle, jsbook なら

```
\documentclass[english]{jsarticle}
```

のように, english オプションを付けます。それ以外のドキュメントクラスでは,

```
\renewcommand{\figurename}{Fig.}
```

のようにプリアンブルに書いておきます。

\caption{図の説明} は図の説明を出力するコマンドですが, \listoffigures とすれば, その説明を使った図目次をその位置に出力することができます。説明が長いときなどは, オプションに短い説明をつけ, それを図目次に出力することもできます。図目次については, 目次を扱う章もご覧ください。

### 5.1.3 図を左右に配置する

2つの図を上下でなく左右に配置したいときもあると思います。ここではそのやり方を扱います。関連のない2つの図を左右に配置するとき, figure 環境の中に minipage 環境を作って,

```
\begin{figure}
\centering
\begin{minipage}{0.4\columnwidth}
\centering
\includegraphics[width=\columnwidth]{figure1.pdf}
\caption{左側}
\label{fig:left}
\end{minipage}
\begin{minipage}{0.4\columnwidth}
\centering
\includegraphics[width=\columnwidth]{figure2.pdf}
\caption{右側(左と関連無し)}
\label{fig:right}
\end{minipage}
\end{figure}
```

のようにすれば, 次のようになります。



図 5.1: 左側

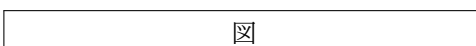


図 5.2: 右側 (左と関連無し)

なお、`\columnwidth` は本文の幅を表します。`minipage` 中では `minipage` の幅が `\columnwidth` になるので上のようにしています。また、`minipage` 自身の幅は好きに指定してください。

関連のある 2 つの図を左右に配置したいときは、`subcaption` パッケージを利用します。まずはプリアンブルに `\usepackage{subcaption}` と書いてください。この使い方は、上とほぼ同じで、`minipage` を `subfigure` に置き換えて、

```
\begin{figure}
\centering
\begin{subfigure}{0.4\columnwidth}
\centering
\includegraphics[width=\columnwidth]{figure1.pdf}
\caption{左側}
\label{fig:left}
\end{subfigure}
\begin{subfigure}{0.4\columnwidth}
\centering
\includegraphics[width=\columnwidth]{figure2.pdf}
\caption{右側(左と連関無し)}
\label{fig:right}
\end{subfigure}
\caption{左右両方}
\label{fig:left-and-right}
\end{figure}
```

とします。この出力は次のようになります。

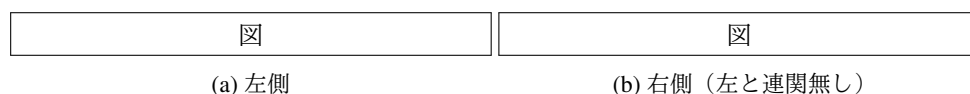


図 5.3: 左右両方

左右の図がくっついて見づらいときは、`\end{subfigure}` と `\begin{subfigure}` の間に適当に `\hspace` でも入れてください。

## 5.2 図の回り込み配置

図を文書の右とか左に配置したいときもあるかと思います。例えばこのように。そんなときは、`wrapfig` パッケージの `wrapfigure` 環境が便利です。

使い方について説明します。まずはプリアンブルに

```
\usepackage{wrapfig}
```

と書いてください。例えば、上のように図を回り込ませたいときは、

```
\begin{wrapfigure}{r}{10zw}
\vspace*{-\intextsep}
\includegraphics[width=10zw]{huukei.jpg}
\end{wrapfigure}
```



のように書きます。`\begin{document}` の 1 つ目の引数に位置 (l か r), 2 つ目の引数にその幅を指定します。オプションで行数も指定できます。ただ、これは自動で計算してくれるので特に指定する必要はありません。まとめると、以下のように使うということです。

```
\begin{wrapfigure}[行数]{位置(l か r)}{幅}
```

上に示した例に`\vspace*{-intextsep}`とありますが、これはデフォルトで入る縦方向のスペース分上に戻す、という操作をしています。これを入れるとちょうどいいって話ですね。また、参考までに、文章との間に入るスペースは`\columnsep`という長さになっています。

なお、`wrapfig`には表の回り込み配置用に `wraptable` 環境も用意されているので、必要ならば使ってみて下さい。また、注意ですが、この `wrapfigure` 環境は箇条書き系の環境（難しいことを言うと、ボックスを生成するもの）と相性が悪いので、併用は避けてください。箇条書きと併用したいときは `mawarikomi` パッケージを使ってみて下さい。また、`float` パッケージを使う場合は、`float` パッケージの後に読み込んで下さい。

## 5.3 色

色の表し方には、RGB と CMYK の 2 通りがあります。RGB (Red, Green, Blue) はディスプレイの色に使われる表し方で、赤・緑・青の 3 色を、0~255 の 256 段階に分け、その値で色を表します。CMYK (Cyan, Magenta, Yellow, black) は印刷物に使われる表し方です。

この RGB と CMYK は互換性がなく、変換がかなり大変らしいのですが、僕はよくわからないですし、変換をしなくてもある程度似た色で印刷されるみたいなので、この話は割愛します。

色を使うときは `xcolor` パッケージを使います。まずはプリアンブル (`\documentclass` と `\begin{document}` の間) に

```
\usepackage{xcolor}
```

と書いてください。

字の色を変えることが多いと思うので、そのやり方を説明します。

グレースケールを指定するときは、

```
{\color[gray]{0.5} 文字}
\textcolor[gray]{0.5}{文字}
```

のようにします。この数値は、黒を 0、白を 1 として、0~1 で指定します。

CMYK を指定するときは、

```
{\color[cmymk]{0.75, 0.64, 0, 1} 文字}
\textcolor[cmymk]{0.75, 0.64, 0, 1}{文字}
```

のようにします。

RGB については、3 通りの表し方があります。各 RGB 値を 255 で割った値を使って、

```
{\color[rgb]{0.75, 1, 0} 文字}
\textcolor[rgb]{0.75, 1, 0}{文字}
```

のようにするのが一般的なようですが、RGB 値をそのまま使って、

```
{\color[RGB]{120, 255, 0} 文字}
\textcolor[RGB]{120, 255, 0}{文字}
```

のようにもできます。また、HTML における色の指定法を用いて、

```
{\color[HTML]{10A3FF} 文字}
\textcolor[HTML]{10A3FF}{文字}
```

のようにすることもできます。

いちいち色を数値で指定するのは面倒ですので、いくつかの色はもともと定義されており、その名前を書くだけで使えます。また、xcolor パッケージに dvipsnames オプションを付けると、もっと多くの色名を使うことができます。

しかし、これだけでは欲しい色が手に入らない、という場合もあると思います。そのときは、次のようにして色を定義します。

```
\definecolor[gray]{x}  
\definecolor[rgb]{r, g, b}  
\definecolor[cmymk]{c, m, y, k}
```

RGB については、文字色の変更のときと同様、オプションを RGB や HTML にして、それに従った記法で定義することもできます。

元々ある色名や定義した色名を使って、次のように色を指定することもできます。

```
{\color{色名} 文字}  
\textcolor{色名}{文字}
```

なお、\color コマンドは、適用範囲を{}で囲まないとそれ以降全ての文字の色が変わってしまうので注意が必要です。

ページの色を変えるには、

```
\pagecolor{色名}
```

とします。それ以降のページは全てその色になるので、白に戻すには\pagecolor{white}とします。文字の背景に色を付けたいときは、

```
\colorbox{色名}{文字}
```

とします。なお、色名はオプションと数値でも指定できます。例えばこんな風にこれに枠を付けたいときは、

```
\fcolorbox{枠色}{背景色}{文字}
```

とします。わくわく



## 第6章 表

この章では表の書き方とその自動配置の仕方を説明します。表を書く場面は少なくないと思うので、是非マスターしましょう。

### 6.1 表の書き方

表を書くには `tabular` 環境を使います。例えば、右下のような表を書くには、左下のように入力します。

```
\begin{center}
\begin{tabular}{rlc}
ID & 名前 & 年齢 \\
1 & 堂成道 椽数 & 30 \\
2 & てふ 太郎 & 36 \\
3 & らてふ 次郎 & 25
\end{tabular}
\end{center}
```

`center` 環境は、表を紙面の中央に配置する命令ですので、表自体には関係がありません。`center` 環境の中で使われている `tabular` 環境が、表を出力するための命令となっています。使い方は、

```
\begin{tabular}{列指定}
内容
\end{tabular}
```

で、この列指定は、

- l 左寄せ (left)
- c 中央寄せ (center)
- r 右寄せ (right)

から、列の数だけ選んで並べます。上の例では、3列を右、左、中央の順で並べたかったので、`\begin{tabular}{rlc}`としたというわけです。

表の中身での行の区切りは`\\`、列の区切りは`&`で行います。第4章4.13節で扱った `align` 環境などと同じですね。

表は1つの大きな文字のように扱われます。つまり、上のように `center` 環境に入れずに文章中に打つ

と、

ID	名前	年齢
1	堂成道 椽数	30
2	てふ 太郎	36
3	らてふ 次郎	25

のようになってしまうということです。基本的には文章中に表を入れた

いわけではないと思うので、`center` 環境や後述の `table` 環境に入れておくのがよいでしょう。

## 6.2 罫線

横線は`\hline`というコマンドで引くことができます。

```
\begin{tabular}{rlc} \hline
ID & 名前 & 年齢 \\ \hline
1 & 堂成道 椽数 & 30 \\ \hline
2 & てふ 太郎 & 36 \\ \hline
3 & らてふ 次郎 & 25 \\ \hline
\end{tabular}
```

ID	名前	年齢
1	堂成道 椽数	30
2	てふ 太郎	36
3	らてふ 次郎	25

`\hline`はその行の上側に線を引く、といったコマンドですので、一番下に罫線を引くには、最終行で改行してから`\hline`を使う必要があります。

縦線は、列指定のときに`|`を線を入れたいところへ書けば、それで出力されます。

```
\begin{tabular}{|rl|lc|} \hline
ID & 名前 & 年齢 \\ \hline
1 & 堂成道 椽数 & 30 \\ \hline
2 & てふ 太郎 & 36 \\ \hline
3 & らてふ 次郎 & 25 \\ \hline
\end{tabular}
```

ID	名前	年齢
1	堂成道 椽数	30
2	てふ 太郎	36
3	らてふ 次郎	25

なお、この罫線で区切られた 1 つの箱を「セル」と言います。下の節でセルという単語を使っていくので覚えておいてください。

また、`\hline`を 2 回使うとその部分の横線が 2 重に、列指定で `r||c` のように `|` を 2 重にするとその部分の縦線が 2 重になります。

`\cline{列番号-列番号}`というコマンドを使うと、部分的に横線が引けます。

```
\begin{tabular}{|ccc|} \hline
虫 & 食 & い \\ \cline{1-2}
穴 & あ & き \\ \cline{2-3}
め & い & ろ \\ \hline
\end{tabular}
```

虫	食	い
穴	あ	き
め	い	ろ

なお、`bookstab` パッケージでも罫線を引くことができます。ただ、横書き文化圏では縦線を引かないため、縦線は引けない仕様となっています。こちらは説明しないので興味があったら調べてみて下さい。

## 6.3 表のレイアウト

表の周りの余白、行や列の間隔、罫線の太さなどを変更することができます。

表を `center` 環境に入れて紙面の中央に置くということが多いと思いますが、`flushleft` 環境や `flushright` 環境を使って左寄せや右寄せをするときもあると思います。ただ、こうしたときに、中央配置のときには見えなかった左右のスペースがあることに気付くと思います。これをなくすには、列指定で

```
{@{}lcr@{}}
```

のように`@{}`をいれてください。この`@{何か}`は、「何か」を表の前後や列と列の間に入れるためのものです。この中身を空にすることによって、デフォルトで入ってしまうスペースを打ち消すことができます。

行送り（行の間隔）は、各行の改行（\\）の後に [長さ] と付ければ、その長さ分だけ増やすことができます。ただ、\hline のある行の行送りを減らすと変な感じになるので注意です。

```
\begin{tabular}{|r|l|c|} \hline
ID & 名前 & 年齢 \\ \hline
1 & 堂成道 椽数 & 30 \\ [10pt]
2 & てふ 太郎 & 36 \\
3 & らてふ 次郎 & 25 \\ \hline
\end{tabular}
```

ID	名前	年齢
1	堂成道 椽数	30
2	てふ 太郎	36
3	らてふ 次郎	25

全体の行送りを一律で変更したいときは、\arraystretch というコマンドを再定義して、

```
\renewcommand{\arrastretch}{倍率}
```

とすれば、行送りがデフォルトの倍率倍されます。

また、array パッケージを読み込んで、\extrarowheight という長さを設定することによって、行送りを一律で増やすことができます。

```
\setlength{\extrarowheight}{長さ}
```

列の間隔は、\tabcolsep をいじって、

```
\setlength{\tabcolsep}{長さ}
```

とすれば、その長さ分&の両脇が空きます。

罫線の太さは、\arrayrulewidth を調整して、

```
\setlength{\arrayrulewidth}{太さ}
```

とします。

また、2 重罫線の間隔は、

```
\setlength{\doublerulesep}{長さ}
```

を使って変えることができます。

## 6.4 左右のセルをまとめる

左右のセルをまとめて 1 つのセルにするには、\multicolumn というコマンドを使います。使い方は、

```
\multicolumn{まとめるセル数}{列指定}{内容}
```

です。配置は、tabular 環境と同様 lcr から選びます。これを使うことで、例えば次のような表を作ることができます。

```

\begin{tabular}{rlc}
\multicolumn{3}{c}{社員表} \\
ID & \multicolumn{1}{c}{名前} & 年齢 \\
1 & 堂成道 椽数 & 30 \\
2 & てふ 太郎 & 36 \\
3 & らてふ 次郎 & 25
\end{tabular}

```

社員表		
ID	名前	年齢
1	堂成道 椽数	30
2	てふ 太郎	36
3	らてふ 次郎	25

一番上の社員表の部分では、3 列をまとめて 1 つのセルにし、それを中央揃えにしています。また、名前のところでは、左寄せの列中で部分的に中央揃えに直す、ということをしています。

罫線もちろん引けます。`\multicolumn` コマンドの列指定で縦線を引くのを忘れないようにしましょう。

```

\begin{tabular}{|r|l|c|} \hline
\multicolumn{3}{|c|}{社員表} \\
ID & \multicolumn{1}{|c|}{名前} & 年齢 \\
1 & 堂成道 椽数 & 30 \\
2 & てふ 太郎 & 36 \\
3 & らてふ 次郎 & 25
\end{tabular}

```

社員表		
ID	名前	年齢
1	堂成道 椽数	30
2	てふ 太郎	36
3	らてふ 次郎	25

## 6.5 上下のセルをまとめる

上下のセルをまとめるには、`multirow` パッケージに入っている `\multirow` コマンドを使います。まずはプリアンブルで

```
\usepackage{multirow}
```

と読み込んでください。

使い方は、

```
\multirow{まとめるセル数}{幅}{内容}
```

です。幅は長さを自分で指定してもよいですが、「\*」とすれば自動で長さを計算してくれるのでこれで大丈夫だと思います。これを使えば、次のような表を作ることができます。

```
\begin{tabular}{|c|c|cc|} \hline
\multirow{3}{*}{パターン} & A & あ & い \\ \cline{2-4}
& B & う & え \\ \cline{2-4}
& C & お & か \\ \hline
\end{tabular}
```

パターン	A	あ	い
	B	う	え
	C	お	か

ただ、この `multirow` には問題があり、上下でまとめている部分の左右のセルが複数行に渡る場合、`multirow` の内容の中央や下揃えがずれてしまいます。また、まとめるセル数は、実際まとめている数と違ってエラーを吐かず動作してしまいます。よって、上で示したくらいの軽い表を作るくらいなら便利なのですが、ちょっと複雑な表を作るときは使いづらいです。

それくらいの複雑な表を作るときは、`tabular` 環境ではなく `tcolorbox` を使いましょう。`tcolorbox` はかなり自由に表を作れる環境です。それゆえにドキュメントが 500 ページほどあるので、ここでは解説しきれません。`tcolorbox` についてはまたどこかで説明しようと思います... 日本語のサイトだと、下に示すものがまとまっていて情報量も多いので、とりあえずリンクだけ貼っておきますね。

「[tcolorbox の基本-物理と TeX に関する話題](#)」

## 6.6 横幅を指定する

普通に表を書くだけでは、横幅は内容によって勝手に決まってしまう。このとき、中身を長くすると紙面からはみ出してしまうこともあって、不便に思うときもあるかもしれません。この横幅をこちらで指定することもできます。列指定を以下のように変更すれば、その列の幅が指定した長さになります。

```
l → p{長さ}
c → >{\centering}p{長さ}
r → >{\raggedleft}p{長さ}
```

一番右の列に `\centering` を入れるとうまくいきませんが、これは右に空の列を追加することで回避できます。

また、表自体の横幅を指定するには、`tabularx` パッケージの `tabularx` 環境を使います。まずはプリアンブルに

```
\usepackage{tabularx}
```

と書きましょう。`tabularx` 環境の使い方は、

```
\begin{tabularx}{幅}{列指定}
...
\end{tabularx}
```

です。列指定に `X` を指定したところは、幅が自由に変わります。長い文を書くと勝手に改行してくれます。

## 6.7 セルの色を変える

セルの色を変えて強調したいときもあるかと思います。行、列でまとめて変えたり、1つのセルだけ指定して変えることもできます。これは `colortbl` パッケージを使えばできますが、第 5 章 5.3 節で説明した `xcolor` パッケージに `table` オプションを付けて

```
\usepackage[table]{xcolor}
```

とすれば `colortbl` パッケージも読み込まれますので、こちらでよいと思います。

色の付け方第 5 章 5.3 節で説明した通りですので、わからなくなったらそちらをご覧ください。

行で一括で色を指定するには、`\rowcolor` コマンドを使って以下のようにします。

```
\begin{tabular}{|c|c|c|} \hline
\rowcolor{red} 1 行目 & 1 行目 & 1 行目 \\ \hline
\rowcolor{orange} 2 行目 & 2 行目 & 2 行目 \\ \hline
3 行目 & 3 行目 & 3 行目 \\ \hline
\end{tabular}
```

1 行目	1 行目	1 行目
2 行目	2 行目	2 行目
3 行目	3 行目	3 行目

また、列でまとめて色を指定するには、以下のようにします。

```
\begin{tabular}{|>{\columncolor{red}}c|>{\columncolor{orange}}c|c|} \hline
1 列目 & 2 列目 & 3 列目 \\ \hline
1 列目 & 2 列目 & 3 列目 \\ \hline
\end{tabular}
```

1 列目	2 列目	3 列目
1 列目	2 列目	3 列目

行と列の色指定が被ってしまったときは、`\rowcolor` が優先されます。

1つのセルだけの色を変えるときは、`\multicolumn` コマンドを使って以下のようにします。

```
\begin{tabular}{|c|c|c|} \hline
白 & 白 & 白 \\ \hline
白 & \multicolumn{1}{|>{\columncolor{orange}}c|}{橙} & 白 \\ \hline
白 & 白 & 白 \\ \hline
\end{tabular}
```

白	白	白
白	橙	白
白	白	白

また、`\usepackage[table]{xcolor}` としていれば、

```
\rowcolors{列数}{色 1}{色 2}
```

というコマンドを使えます。これは、指定した行から 1 行ごとに色 1 と色 2 を交互に繰り返すコマンドです。ここまでくると Excel と同じような表が作れますね。

```

\rowcolors{3}{white}{orange}
\begin{tabular}{|c|c|c|c|} \hline
1 列目 & 2 列目 & 3 列目 & 4 列
目 \\ \hline
1 列目 & 2 列目 & 3 列目 & 4 列
目 \\ \hline
1 列目 & 2 列目 & 3 列目 & 4 列
目 \\ \hline
1 列目 & 2 列目 & 3 列目 & 4 列
目 \\ \hline
1 列目 & 2 列目 & 3 列目 & 4 列
目 \\ \hline
1 列目 & 2 列目 & 3 列目 & 4 列
目 \\ \hline
\end{tabular}

```

1 列目	2 列目	3 列目	4 列目
1 列目	2 列目	3 列目	4 列目
1 列目	2 列目	3 列目	4 列目
1 列目	2 列目	3 列目	4 列目
1 列目	2 列目	3 列目	4 列目
1 列目	2 列目	3 列目	4 列目
1 列目	2 列目	3 列目	4 列目

## 6.8 ページまたぎのできる表

tabular 環境で作る表はページまたぎができません。したがって、長めの表になると、前ページを大きく残したまま次のページに出力されてしまいます。これは紙面の無駄ですし、見づらいので嫌だという人も少なくないと思います。

ページまたぎをするには、tabular 環境の代わりに longtable 環境を使います。ただ、longtable 環境を使うには longtable パッケージを読み込む必要があるの、まずはプリアンブルに `\usepackage{longtable}` と書いてください。

longtable 環境の使い方は tabular 環境と同じです。特に例を示す必要もないと思うので、必要になったら使ってみて下さい。

## 6.9 表に図を挿入する

表中に画像を挿入するには、第 5 章第 5.1 節で触れたように、minipage 環境を使います。

## 6.10 表の配置

表を自動配置するには、table 環境を使います。図のときと同様、表にもキャプションとラベルを付けることができます。表のキャプションは上に付けるのが標準ですので、例えば以下のようにします。

```

\begin{table}
\caption{3 次の魔法陣}
\label{tab:magic-table}
\begin{center}
\begin{tabular}{|c|c|c|} \hline
2 & 9 & 4 \\ \hline
7 & 5 & 3 \\ \hline
6 & 1 & 8 \\ \hline
\end{tabular}
\end{center}
\end{table}

```

以下，説明を加えますが，基本的に figure 環境と同じです。

table 環境のオプションは，figures 環境のオプションと同じなので，表??を参照してください。なお，「H」は float パッケージを読み込んでいないと使えませんので注意してください。

また，回り込み配置をしたければ，wrapfig パッケージの wraptable 環境を使ってください。使い方は第 5 章 5.2 節で説明した wrapfigure 環境と同じですので，そちらを見て下さい。

ま	わ	り
こ	み	。

表の番号は自動的に「表 1」，「表 2」，... のように付きます。これを「Table1」のようにしたければ，jsarticle，jsbook なら

```
\documentclass[english]{jsarticle}
```

のように，english オプションを付けます。それ以外のドキュメントクラスでは，

```
\renewcommand{\tablename}{Table}
```

のようにプリアンブルに書いておきます。

\caption{説明} で表に説明を付けることができますが，\listoftables とすればその説明を使った表目次をその位置に出力することができます。また，オプションを付ければそれが表目次に表示されます。

また，回り込み配置についてもここで触れてしまいます。表の回り込み配置は，wrapfig パッケージの wraptable 環境を使えば簡単にできます。使い方は図の回り込み配置用の wrapfigure 環境と同じなので，詳しくは第 5 章 5.2 節を読んでください。



## 第7章 相互参照・目次・リンク

この章では、相互参照と目次、そしてリンクを扱います。これらは文書内を検索する際に非常に便利です。是非使えるようになっておきましょう。

### 7.1 相互参照

`\label{ラベル}`としてラベルを付けた章や節、図や表、数式などは、`\ref{ラベル}` コマンドを使って参照することができます。また、そのページ数を参照したいときは、`\pageref{ラベル}`というコマンドを使います。ただ、2回実行しないと正しく数字が現れないことに注意してください。このラベル名は、章は `ch:`、節は `sec:`、図は `fig:`、表は `tab:`、数式は `eq:`で始める、というように統一しておくといいです。それぞれへのラベルの付け方は、今まで扱った通りですので、そちらをご覧ください。

2回実行しないとイケないのは、1回目の実行で `label` の情報を別のファイルに書き込み、2回目の実行で `ref` に `label` の情報を与えていくからです。これは次節で扱う目次でも同じです。TeXには2回実行しないと正しく表示されないものがしばしばあるので、2回実行する癖をつけてもいいかもしれません。

ここで注意なのですが、間違っって同じ名前前のラベルを2個設定してしまうと、1回目の実行ではエラーを吐かないのですが、2回目の実行からエラーしてしまいます。なお、僕の環境では「!LaTeX Error: Missing \begin{document}」と出ました。これを解決するには、ラベルの情報を記録している.toc ファイルを削除するしかありません。これに気付くには、エラーの前に出る「LaTeX Warning」で、同じラベルが2個あるといった主旨のものが有りますので、これを見つけたら.toc ファイルを削除してください。

### 7.2 目次

目次を出力するには、

```
\tableofcontents
```

とするだけで OK です。ただ、2回実行しないとちゃんと出力されないのに注意してください。

他にも、`\listoffigures`、`\listoftables` で図目次、表目次が出力されます。第5章第5.1.2節でも触れましたが、`\caption{説明}`の説明がそのまま目次となります。これをもっと短めにしたければ、オプションで指定してください。

目次にどこまで細かく見出しを出力するかは簡単に変更できます。節 (`\section`) までにするには、

```
\setcounter{tocdepth}{1}
```

小節 (`\subsection`) までなら

```
\setcounter{tocdepth}{2}
```

とすればよいです。

また、「\*」つきの、番号が付かない見出しについて (`\section*{はじめに}` など) は、番号だけでなく目次にも出力されません。これを目次に出力したかったら、

```
\section*{はじめに}
\addcontentsline{toc}{section}{はじめに}
```

または

```
\section*{はじめに}|
\addcontentsline{toc}{section}{\numberline{}はじめに}
```

のようにします。後者は、他の節等で番号が入っている分だけ字下げします。

なお、ドキュメントクラスが `book` のときは、序文や後書きは `\frontmatter`、`\backmatter` を使う方がよいです。

## 7.3 リンク

リンクについて説明します。リンクを貼るには、`hyperref` パッケージを使います。オプションにドライバ名を指定できますが、`\documentclass` に `dvipdfmx` オプションを付けておけば十分でしょう。また、日本語を使うには、`hyperref` の後で `pxjahyper` パッケージを読み込む必要があります。まずはプリアンブルに

```
\usepackage{hyperref}
\usepackage{pxjahyper}
```

と書きましょう。

### 7.3.1 PDF 内リンク

`hyperref` (と `pxjahyper`) を読み込めば、目次や相互参照、また次章で扱う参考文献は自動的にリンクが有効になり、クリックすれば参照場所に飛べるようになります。

また、この他にも、好きな文字にリンクを貼ることができます。そのためにまず、リンクで飛ぶ先の文字を以下のようにしてください。

```
\hypertarget{リンク名}{リンク先文字}
```

次に、リンクを有効にしたい文字を次のようにしてください。

```
\hyperlink{リンク名}{リンク元文字}
```

このようにすれば、リンク元文字をクリックするだけでリンク先文字に飛ぶことができます。

### 7.3.2 WEB ページへのリンク

WEB ページへのリンクとはつまり URL のことですが、これにリンクをつけるには

```
\url{URL}
\href{URL}{テキスト}
```

のようにします。URL には  $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  では特殊文字として扱われるものが入っていることが多いですが (ex. `%`, `~`, `_` など), そのまま入れて大丈夫です。

$\mathrm{T}_{\mathrm{E}}\mathrm{X}$  の実行結果として PDF を開いているときはリンクが使えないことがありますが、最初から PDF として開けば基本的にはリンクは有効になっています。

例えば、この章は  
「<https://texwiki.texjp.org/?hyperref>」  
と  
「[ハイパーリンク付き LaTeX 文書](#)」  
というサイトを参考にしました。

### 7.3.3 その他

他にも、メールアドレスや外部ファイルをリンク先にすることができます。メールアドレスを指定するときは、

```
\href{mailto:アドレス}{テキスト}
```

としてください（テキストにはアドレスを使うのがよいと思います）。  
外部ファイルを参照するには、

```
\href{ファイル名}{テキスト}
```

とします。ここで、このファイル名はパスで指定します。パスについては第 11 章まで。  
また、画像には `\hyperimage{画像の URL}`、としてリンクを貼ることができます。

### 7.3.4 hyperref のオプションと PDF の文書情報

`hyperref` を使えば楽にリンクが貼れることがわかりました。ただ、デフォルトだと枠で囲まれる形になるので、これが嫌ならオプションを付けて変更します。また、同時に PDF に文書情報をつけることもできますので、そちらも一緒に扱います。

以下に示したものの以外にも多くのオプションがありますが、使いそうなものだけ抜粋したのでもし不足していれば「[TeX Wiki hyperref](#)」や `hyperref` のドキュメント等を見てください。

colorlinks	リンクを枠囲いでなく文字色の変化で表す。デフォルトは false。 colorlinks=true として変更する。
hidelinks	リンクの色や枠を無効にする。デフォルトは false。
anchorcolor	anchor テキストの色。デフォルトは black。
linkcolor	PDF 内参照用のリンクの色。デフォルトは red。
linkbordercolor	PDF 内参照用のリンクの枠囲いの色。デフォルトは red。
urlcolor	URL 参照用リンクの色。デフォルトは cyan。
urlbordercolor	URL 参照用リンクの枠囲いの色。デフォルトは cyan。
citecolor	参考文献参照用のリンクの色。デフォルトは green。
citebordercolor	参考文献用リンクの枠囲いの色。デフォルトは green。
filecolor	外部ファイル参照用のリンクの色。デフォルトは cyan。
filebordercolor	外部ファイル参照用リンクの枠囲いの色。デフォルトは [rgb]{0, .5, .5}。
bookmarks	PDF にしおりをつける。デフォルトは false。変更は colorlinks のときと同様。
bookmarksnumbered	しおりに章番号等をつける。デフォルトは false。
pdftitle	PDF のタイトルを設定する。デフォルトでは何も設定されていない (空)。
pdfauthor	PDF の作者を設定する。デフォルトでは空。
pdfkeywords	PDF のキーワードを設定する。デフォルトでは空。

以上が使いそうなオプションとなります。このオプションは本来、

```
\usepackage{hyperref}
```

のオプションに設定するものですが、長くなりすぎる場合があるので別にコマンドが用意されています。`\hypersetup{}`の引数にオプションを入れていけば大丈夫です。例えばこのマニュアルの設定は以下のようになっています。

```
%文書情報とリンク
\hypersetup{%
  bookmarks=true,%
  bookmarksnumbered=true,%
  hidelinks,%
  colorlinks=true,%
  linkcolor=black,%
  urlcolor=cyan,%
  citecolor=black,%
  filecolor=magenta,%
  setpagesize=false,%
  pdftitle={LaTeX2e 入門マニュアル},%
  pdfauthor={R. Morita},%
  pdfkeywords={TeX; LaTeX}
}
```

もしリンクを貼りたくないラベルがあれば、`\ref*{label}`のように`\ref`コマンドに「\*」をつけてください。

なお、この `hyperref` は曲者らしく、使うと色々な弊害があるらしいです。今のところ特に実害はないので気にしていませんが、ちょっと凝ったことをしようとしている人はご注意を。ネットでは、`graphicx` の後に読み込むとエラーしたなどという記事もあるので、困ったらその辺を見てみて下さい。

## 第8章 ページレイアウトについて

この章では、ページレイアウトを扱っていきます。L<sup>A</sup>T<sub>E</sub>X の考えでは、本文とページレイアウトは別物として扱うのがよいとされています。つまり、本文中でレイアウトを調整するのは、美しくないということです（勿論多少は本文中でも調整します）。ではどの部分でレイアウトを調整するかというと、それは `\documentclass` とプリアンブルです。ここからは文書の内容以外の部分も扱っていきます。

### 8.1 ドキュメントクラス

T<sub>E</sub>X で文書を作るとき、最初の行におまじないのように `\documentclass{jsarticle}` と書いている人が多いと思います。このドキュメントクラスは、普通に文書を作成するだけなら変える必要がない場面が多いです。ですので、ドキュメントクラス自体については軽く触れる程度にしたいと思います。ドキュメントクラスには、大きく分けて以下の3つの種類があります。

- article** レポートや記事，論文で使われるクラスです。最も一般的に使われるクラスで、ほとんどの人がこのクラスで文書を作成します。節（`\section`）が最上位の構成単位です。
- report** 長めの報告書用のクラスです。**article** より1つ上の、章（`\chapter`）が最上位です。
- book** 本用のクラスです。章（`\chapter`）が最上位です。

基本的には `jsarticle(japan standard article)` を使っていれば大丈夫です。長めの文書を作成したり本を書いたりしたいときは、適当にググってどうにかしてください。

### 8.2 ドキュメントクラスのオプション

この節ではドキュメントクラスにつけるオプションについて扱います。ただ、これも深入りはしません。普通の使い方をする場合で、実用上使いそうなものだけに限って紹介します。

`\documentclass` には様々なオプションをつけることができます。例えば、図を読み込んだりするときに `[dvipdfmx]` というオプションをつけると思います。こんな感じで、以下のようなオプションが用意されています。使いそうなものしか紹介していないので、もっと突っ込んだ設定をしたい人は色々ググってみてください。

10pt	文書全体で、文字サイズを 10pt にします（デフォルト値）。10pt 以外にも次の値に変更できます。→ 9pt, 11pt, 12pt, 14pt, 17pt, 21pt, 25pt, 30pt, 36pt, 43pt
a4paper	文書の用紙サイズを A4 にします。同様に A4, A5, B4, B5 が指定できます。
landscape	用紙の向きを横置きにします。
twocolumn	文書を 2 段組みにします（デフォルトは 1 段組の onecolumn です）。左右の段の間の余白を変更したいときは、 <code>\setlength{\columnsep}{長さ}</code> とすれば OK です。また、中心に太さ $w$ の線を引きたいときは、 <code>\setlength{\columnseprule}{w}</code> とします。3 段組以上を使うには、multicol パッケージを使います。
titlepage	<code>\maketitle</code> で出力する表題と abstract 環境で出力する abstract を、単独ページに出力します。jsarticle では、これらが本文 1 ページ目の上に出力される notitlepage がデフォルトです。
leqno	数式の番号を左側に出力します。デフォルトでは右側に出力されます。
fleqn	数式を本文の左端から一定距離のところから始めます。デフォルトでは数式は中央に配置されます。左端からの距離は、デフォルトでは 3zw くらいですが、これを変えたかったら <code>\setlength{\mathindent}{5zw}</code> によりして設定します。
dvipdfmx	ドライバを指定します。これにしとけばまず大丈夫です。図とかの出力の際に指定が必要な場合が多いので、指定しておいて損はないです。

## 8.3 紙面上下の余白

実用的な話ですが、レポートを書くときなど、A4 用紙 3 枚以内、のような指定がある場合もあると思います。そんなとき、紙面の上下左右のスペースを削って無理矢理押し込める、なんてことができれば嬉しいですね。この節ではそのやり方を説明したいと思います。

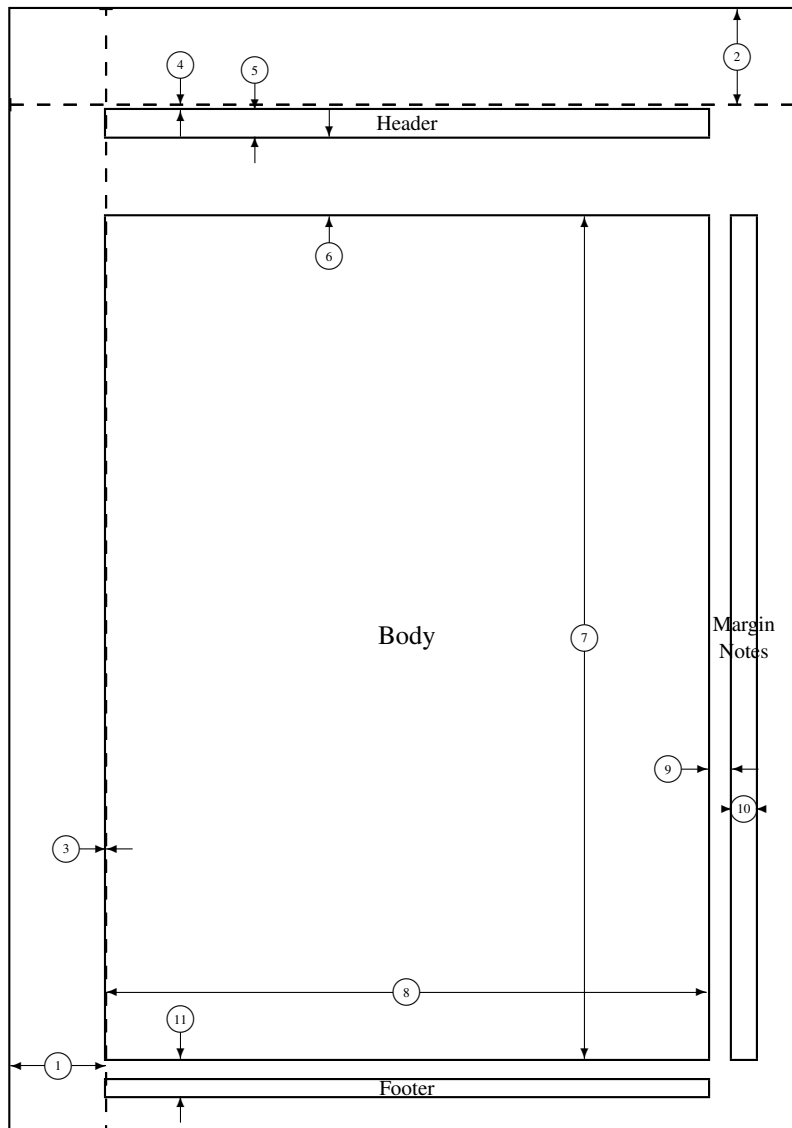
このマニュアルのレイアウトは次のページのようになっています（おそらくデフォルト値です）。それぞれの長さがコマンドとして定義されているので、それらを再定義してあげれば長さを変えることができます。具体的には、

```
\setlength{長さのコマンド}{長さ}
```

とすれば好きな長さにすることができます。もちろん、この長さは、表 ?? に載っている長さで指定してください。また、この命令は、ドキュメント中ではなくプリアンブルに書くとよいでしょう。

なお、縦横で長さの辻褄（合計値）を合わせないと、ヘッダーと本文が重なったり、本文が紙面からはみ出したりしてしまうこともあるので、注意してください。基本的には試行錯誤でいい感じのところを見つける形になると思います。

ちなみに、次ページのレイアウトのレビューは、layout というパッケージを読み込み、`\layout` と打ち込めば出力されます。



1	one inch + \hoffset	2	one inch + \voffset
3	\oddsidemargin = 0pt	4	\topmargin = 4pt
5	\headheight = 20pt	6	\headsep = 60pt
7	\textheight = 634pt	8	\textwidth = 453pt
9	\marginparsep = 18pt	10	\marginparwidth = 18pt
11	\footskip = 28pt		\marginparpush = 5pt (not shown)
	\hoffset = 0pt		\voffset = 0pt
	\paperwidth = 597pt		\paperheight = 845pt

## 8.4 インデント

次はインデントについてです。インデントとは、文字の頭下げのことです。頭下げとは、例えば段落が変わったときに先頭を 1 字空けたり、数式を真ん中らへんにするときの頭の空白のことです。

まずは文章中のインデントについてです。特定の位置で改段落した後、インデントしたくないときは、その手前に `\noindent` と打てばよいです。文章全体でインデントの長さを設定したいときは、プリアンブルで

```
\setlength{\parindent}{長さ}
```

としてください。例えば、

```
\setlength{\parindent}{0pt}
```

とすれば、改段落時の頭下げは一括で 0 になります。

次に、数式のインデントについてです。先ほどドキュメントクラスのオプションを扱った節で説明しましたが、数式はデフォルトでは紙面の中央に来るように設定されており、ここをいじるにはまず `\documentclass` のオプションに `fleqn` を指定します。つまり、

```
\documentclass[fleqn]{jsarticle}
```

というようにしてください。こうすると、デフォルトでは本文左端から 3zw のところから数式が始まるようにになります。これをいじるには、プリアンブルで

```
\setlength{\mathindent}{5zw}
```

のようしてください。

## 8.5 行送り

行送りとは、例えば A という文字が 2 行に並んでるときの、A の下端を通る水平線間の距離のことです。この水平線をベースラインといいます。

`jsarticle` において、行送りは `\baselineskip` というコマンドで定義された長さになっていて、そのデフォルト値は 16pt です。

行送りを変更したい場合、

```
\setlength{\baselineskip}{長さ}
```

のように `\baselineskip` の定義を変更するか、

```
\renewcommand{\baselinestretch}{倍率}
```

とすればいいです。

`\baselineskip` の定義を変えた場合は、文字のサイズを変更したタイミングで元に戻ってしまいます。`\baselinestretch` を変更すれば、文字のサイズを変更しても元には戻りません。

本文中に  $\frac{d}{dx}(\log f(x)) = \frac{f'(x)}{f(x)}$  のような大きい数式があると、行送りはその分だけ大きくなり、文字が重ならないようにします。これは、上下の行が `\lineskiplimit` だけ近づいたら、`\lineskip` より近づかないようにする、というようにして実現しています。この `\lineskip` は、`jsarticle` のデフォルトでは 1pt に設定されていますが、本文中に大きい数式を何度も入れる場合は、

```
\setlength{\lineskiplimit}{3pt}
\setlength{\lineskip}{3pt}
```

のように広めを取っておくとよいです。



## 8.6 字間

字間が狭くて見づらい、なんていう場合は、字間を広げる必要があります。全角文字の間の字間の長さは`\kanjiskip`というコマンドで、全角文字と半角文字の間の字間の長さは`\xkanjiskip`というコマンドで定義されています。これをいじることで字間を調整できます。例えば、全角文字間、全角文字と半角文字の間の字間をデフォルトで 0.5pt にし、改行位置が変になるなどの場合に応じて 0.1pt の伸び縮みを認める、という形にしたかったら、

```
\kanjiskip 0.5pt plus 0.1pt minus 0.1pt
\xkanjiskip \kanjiskip
```

のようにすれば OK です。

## 8.7 ヘッダーとフッター

この節ではヘッダーとフッターの設定の仕方について扱います。名前や所属、日付やページ数を右上に表示したりする場面などで役に立ちます。

### 8.7.1 T<sub>E</sub>X 標準のヘッダーとフッター

ヘッダーやフッターを設定するコマンドは、

```
\pagestyle{ページスタイルの名前}
```

です。このページスタイルには以下のような種類があります。

<code>empty</code>	ヘッダー・フッター（ページ番号含む）を表示しません。
<code>plain</code>	フッターの中央にページ番号が出力されます。デフォルトはこれです。
<code>headings</code>	左ヘッダーに節の名前が、右ヘッダーにページ数が出力されます。ヘッダーの区切りの横線は表示されません。
<code>myheadings</code>	<code>heading</code> と同様ですが、ヘッダーの内容を好きに設定できます。設定方法は以下で示します。

プリアンブルでこれを設定しておけば、ページ全体のヘッダー・フッターが指定したように設定されます。なお、`\pagestyle` は文書中でも使うことができ、それが書かれているページ以降が指定したページスタイルになります。

また、あるページだけ違うページスタイルにしたいときは、

```
\thispagestyle{ページスタイル}
```

を使えば、そのページだけ設定が変わり、次ページから元に戻ります。

`myheadings` でのヘッダーの設定方法について説明します。偶数ページと奇数ページで同じヘッダーを出力するなら、

```
\markright{ヘッダー名}
```

とします。

偶数ページと奇数ページでヘッダーが違うときは、

```
\markboth{左}{右}
```

とすれば、見開きの左ページのヘッダーには「左」、右ページのヘッダーには「右」と出力されます。ただ、もっと自由に設定したいときは、次小節の `fancyhdr` というパッケージを使ってください。

## 8.7.2 fancyhdr を使う

`fancyhdr` というパッケージを使うと、もっと楽に自由にヘッダー、フッターを設定できます。まずはブリアンプル (`\documentclass` と `\begin{document}` の間) に

```
\usepackage{fancyhdr}
```

と書いてください。

全体が同じでいいときは、`fancy` というページスタイルを使って、

```
\pagesyle{fancy}
\lhead{左ヘッダー}
\chead{中央ヘッダー}
\rhead{右ヘッダー}
\lfoot{左フッター}
\cfoot{中央フッター}
\rfoot{右フッター}
```

のようにすれば、それぞれの位置に出力がされます。文書中でこのコマンドを打てば、そのページ以降のヘッダー or フッターが、そこで打ったものになります。また、ヘッダーの線を消したいときは、

```
\renewcommand{\headrule}{}%
```

としてください。

1 ページ目とそれ以降のページ、最後のページでヘッダーやフッター、ページレイアウトが違うとき、逐一上のコマンドを打って変更してもよいですが、

```
\fancypagestyle{ページスタイル名}{内容}
```

というコマンドを使ってページスタイルを定義し、

```
\thispagestyle{ページスタイル}
```

を使って局所的にページスタイルを変更することで、ヘッダーやフッター、ページレイアウトを一括で変更することができます。

これだけではわかりづらいと思うので、具体的に例を見てみます。この文書のヘッダー・フッターの設定を以下に示します。

```
%ヘッダー&フッター
\fancypagestyle{normal}{}%
%ヘッダー
\lhead{\leftmark}
\rhead{\rightmark}
\renewcommand{\headrule}{}%
%フッター
\cfoot{\thepage}
}
\pagestyle{normal}
```

ここで、`\leftmark`、`\rightmark`、`\thepage` はそれぞれ、章、節、ページ数を出力するコマンドです。`\pagestyle` で定義したページスタイルをちゃんと選択することを忘れないでください。

また、ヘッダーやフッターに画像を挿入したいときは、第 5 章第 5.1 節の通り、`minipage` 環境を作れば簡単に入れられます。

## 8.8 横置き

### 8.8.1 PDF 全体を横置きにする

基本的に PDF の紙は縦に置かれていますが、8.2 節に書いた通り、`\documentclass` のオプションに `landscape` を指定すれば紙を横置きにすることができます。

ただこの `landscape`、うまく横置きにならないときがあります。そのときは `geometry` パッケージに `landscape` オプションを付けて読み込めばうまくいくと思います。つまり、次のようにするということです。

```
\documentclass[landscape]{jsarticle}
\usepackage[landscape]{geometry}
\begin{document}
.....
\end{document}
```

これでうまくいかなかったら（多分行くと思いますが）是非自分で調べてみて下さい。

ちなみに、8.7.2 節で紹介した `fancuhdr` パッケージは、`landscape` を使っていないもうまく動くので是非使ってみて下さい。

### 8.8.2 PDF の一部だけ横置きにする

前小節では PDF 全体を横置きにする方法を説明しました。ここでは、PDF の一部を横置きにする方法を説明します。これを実現するために、まずは `lscape` パッケージを読み込みます。

```
\usepackage{lscape}
```

このパッケージに入っている `landscape` 環境の中に入れた部分が、（改ページをして別ページで）横置きになります。

```
\documentclass{jsarticle}
\usepackage{lscape}
\begin{document}
~~~~~
ここまで縦置き

\begin{landscape}
ここは横置き
この前後で改ページされる
\end{landscape}

ここからまた縦置き
~~~~~
\end{document}
```

どこに使うんだよ、と思う人もいるかもしれませんが、例えば横に長い図や表を入れようとして、紙面の右からはみ出てしまうようなときなどに使えます。例えば次のような横にながーい表でも、次ページのように紙面に収めることができます。

この表は	A4 サイズの紙に	普通に	書くと	はみ出て	しまいます。	でも	こんな	横長の	表でも	紙面に
この表は	A4 サイズの紙に	普通に	書くと	はみ出て	しまいます。	でも	こんな	横長の	表でも	紙面に
この表は	A4 サイズの紙に	普通に	書くと	はみ出て	しまいます。	でも	こんな	横長の	表でも	紙面に

表 8.2: 横にながーい表

この表は	A4 サイズの紙に	普通に	書くと	はみ出て	しまいます。	でも	こんな	横長の	表でも	紙面に	収めることが	出来ます。
この表は	A4 サイズの紙に	普通に	書くと	はみ出て	しまいます。	でも	こんな	横長の	表でも	紙面に	収めることが	出来ます。
この表は	A4 サイズの紙に	普通に	書くと	はみ出て	しまいます。	でも	こんな	横長の	表でも	紙面に	収めることが	出来ます。

表 8.3: 横にながーい表

ここでいくつか注意点を挙げておきます。まず、内容は横置きですが、紙自体は縦になります。紙自体が途中で横になると、論文等では困ってしまうので、ある意味当然のデフォルト設定です。また、ヘッダーやフッターも縦置きのとおりと同じようにつきます。これも上と同様の理由です。

横に長いものに出会ったときは是非使ってみて下さい。

### 8.8.3 PDF の一部を紙ごと横置きにする

上で PDF 内の一部だけ横置きにする方法はわかりました。しかし、中には紙ごと横に置きたい人もいます。そんな人のために、`pdfscape` というパッケージがあります。プリアンブルで

```
\usepackage{lscap}
\usepackage{pdfscape}
```

として読み込むだけで、`landscape` 環境を使っているページが横向きになります。

ただ、この `pdfscape` パッケージは、そのページを機械的に横向きにするだけであるということに注意してください。つまり、ヘッダーやフッターを付けていた場合、それらも丸ごと回転してしまい、変なレイアウトになってしまうということです。

例えば、論文やレポートを書いている、横に長い図を `lscap` パッケージの `landscape` 環境で入れた方がいいが、内容を見たいときに非常に見づらい、というときなどは、一時的に横向きにするという目的でこの `pdfscape` を使えばよいので、特にヘッダーとフッターの回転に気を配る必要はありません。

しかし、紙面を横にしたままどこかしらに提出するということは、ヘッダーやフッターが回転してはよくないでしょう。簡単な対策としては、`landscape` 環境を使うページだけヘッダーやフッターを付けないというのが考えられます。表や図のみになる特殊なページですから、これは全然ありだと思います。

でもやっぱりヘッダーとフッターを付けたい、という人もいます。ページ数を付けなくていいのであれば、`pdfpages` というパッケージを用いて上手くやる方法があります。横向きの PDF を別ファイルで用意します。これは `documentclass` のオプションに `landscape` を付けて、`fancyhdr` 等も使って作ればよいです。とりあえずこの PDF の名前を「`yoko.pdf`」としておきます。このような別ファイルで用意した PDF を、そのページを指定して挿入できるのが `pdfpages` パッケージです。プリアンブルに

```
\usepackage{pdfpages}
```

と書き、`yoko.pdf` を読み込みたいところで

```
\includepdf[pages=-]{yoko.pdf}
```

とすれば OK です。もちろん縦置きの PDF にも使えるパッケージですので、必要があれば使ってみて下さい。なお、`pdfpages` については、「[につき：pdfpages](#)」というサイトを参考にしました。様々なオプションについても解説してくださっているので、是非一度見てみて下さい。

しかし、ページ数も欲しいよ、という人もいます。そんな人は、ヘッダーやフッターを手動で配置しましょう。`textpos` パッケージの `textblock` 環境を使えば PDF の紙面上にテキストを配置できます。これに `\rotatebox{角度}{テキスト}` というコマンドを組み合わせれば、横置きに合わせたヘッダー・フッターを付けることができます。`fancyhdr` パッケージと合わせて、いい感じに設定していきましょう。すごく適当な感じになっているので、そのうち書き足します.....

まずは必要なパッケージをプリアンブルで読み込みます。`textpos` には `absolute` オプションを付けます。

```
\usepackage{fancyhdr}
\usepackage[absolute]{textpos}
\usepackage{lscap}
\usepackage{pdfscape}
```

次に `textpos` の初期設定をします。`textpos` の `blocktext` 環境は紙面左上を (0, 0) とした座標でテキストを配置します。その座標の長さの単位を設定します（デフォルトは pt ですが、このままでよいという人はこのままで大丈夫です）。

```
\setlength{TPHrzoModule}{1mm}
\setlength{TPVertModule}{1mm}
```

`blocktext` 環境の使い方は、簡単に説明すると次のような感じです。

```
\begin{blocktext}{幅}{座標}
内容
\end{blocktext}
```

僕は「[天地有情 \[LaTeX\] textpos — テキストブロックを任意の場所に配置](#)」というサイトを参考にさせていただきました。

次に `fancyhdr` でページスタイルを定義します。

```
\fancypagestyle{yoko}{%
%header
\lhead[L]{%
\begin{textblock}{3}(10, 20)
\rotatebox{90}{紙ごと横置き}
\end{textblock}
}
%footer
\cfoot{%
\begin{textblock}{3}(180, 145)
\rotatebox{90}{\thepage}
\end{textblock}
}
\lfoot{%
\begin{textblock}{3}(180, 145)
\rotatebox{90}{フッターも置ける}
\end{textblock}
}
}
```

ここで、`\thepage` はページ番号を出力するコマンドです。

そして、`landscape` 環境内で `\thispagestyle` コマンドを使ってページスタイルを指定すれば OK です。`landscape` 環境は改ページを伴うので、環境の外で使うと違うページのスタイルが変更されてしまうことに注意してください。

```
\begin{landscape}
\thispagestyle{yoko}

ここの紙だけ横置き
\end{landscape}
```

かなり面倒ですが、これならかなり自由に設定できるので、簡単な方法が見つからないときは是非これを試してみてください。

## 第9章 欧文フォントについて

この章では欧文フォントについて説明していこうと思います。欧文文字（アルファベット）などや数式のフォントを自分の好きなフォントにすることができます。キレイな文書を作るのならフォントにも拘らねばなりませんね。

### 9.1 欧文フォントの5要素

$\text{\LaTeX}$ 2 $\epsilon$  において、フォントは以下の5つの要素によって管理されています。それらを軽く説明していきます。

#### 9.1.1 エンコーディング

コンピュータは文字というものを直接扱うことはできません。扱えるのは0と1という数字だけです。そこで、コンピュータで文字を扱おうとしたとき、文字に通し番号を振ってその番号で管理する、という方法が取られました。一般に、この番号の振り方をエンコーディング（encoding）と言います。

しかし、この小節で扱うのは、一般的によく使われる UTF-8 や Shift-JIS などの話ではなく、 $\text{\TeX}$  の中のエンコーディングについての話です。

$\text{\TeX}$  のデフォルトのエンコーディングは OT1 というものですが、今は T1 エンコーディングを使うのがよいです。エンコーディングを T1 に変更するには、プリアンブル（`\documentclass` と `\begin{document}` の間）に

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

と書けば OK です。

#### 9.1.2 ファミリ（書体）

エンコーディングを T1 にしたら、ファミリ（family）もデフォルトの Computer Modern から T1 エンコーディングと相性のいい Latin Modern に変えましょう。とりあえずプリアンブルに

```
\usepackage{lmodern}
```

と書いてください。

これらのファミリ（書体）には、以下の字体があります。基本的な使いどころも以下にまとめておきます。

- セリフ体 : 本文
- サンセリフ体 : 見出し
- タイプライタ体 : コンピューターへの入力を表す部分



「セリフ (serif)」とは、線の端についた飾りのことです。例えば I や W などの上下についている横棒がセリフです。

また、「サンセリフ (sans serif)」についてですが、sans が「ない」という意味を持つので、これはセリフのない、つまり飾りがないということを表します。I や W を見ていただければ違いが判ると思います。サンセリフ体は飾りがないのに加え、線の太さがほぼ一定となっています。

これら3つの字体を切り替えるには、それぞれ以下の表のようにします。なお、本文でのデフォルトはセリフ体です。

- セリフ体 (Serif, Roman) : `\textrm{文字}` or `\rmfamily 文字`
  - サンセリフ体 (Sans Serif) : `\textsf{文字}` or `\sffamily 文字`
  - タイプライタ体 (Typewriter Type) : `\texttt{文字}` or `\ttfamily 文字`
- なお、切り替え方は2通りありますが、今は `\text...\{}` を使うのがよいとされています。

### 9.1.3 シリーズ

次はシリーズ (series) についてです。ウェイト (weight) ともいわれます。これは文字の太さを管理します。一般的には (単ウェイトなら) 次のコマンドで切り替えます。

- Medium : `\textmd{文字}` or `\mdseries 文字`
- Boldface : `\textbf{文字}` or `\bfseries 文字`

本文中のデフォルトは Medium です。単ウェイトなどといった話は、次章の和文フォントについてで扱います。

### 9.1.4 シェープ

シェープ (shape) とは、アップライト (Upright)、イタリック (*Italic*) といった文字の形の事です。以下のコマンドで切り替えます。

- Upright : `\textup{文字}` or `\upshape 文字`
- Italic : `\textit{文字}` or `\itshape 文字`
- SMALL CAPS : `\textsc{文字}` or `\scshape 文字`

本文中のデフォルトは Upright です。これらの他に *Slanted* という、文字を機械的に斜めにした物が用意されているフォントもありますが、これはイタリック体があれば不要です。一応紹介しておく、`\textsl` や `\scshape` で切り替えられます。

### 9.1.5 サイズ

これはそのまま文字サイズの事です。ポイント (pt) という単位で表されることが多いです。文字サイズの変更については、第2章第2.5節で解説した通りです。

ではこれらの用意されたサイズにしかできないのかというと、そんなことはありません。

```
\fontsize{文字サイズ}{行送り}
```

というコマンドを使えば、好きなフォントサイズ行送りに設定できます。例えば、

```
\fontsize{12pt}{20pt}
```

とすれば、フォントは 12pt、行送りは 20pt となります。

## 9.2 フォントの変更について

今度書き足すと思います。気が向いたら。

とりあえずおススメは、

```
\usepackage[T1]{fontenc}
\usepackage{textcomp}
\usepackage[utf8]{inputenc}
\usepackage{mathptmx}
\usepackage[scaled]{helvet}
\renewcommand{\ttdefault}{pcr}
```

とプリアンブルに書くことです。

何をしているかだけ説明しておくと、

- エンコードを T1 にする
- TC(Text Companion) 文字を使えるようにする
- エンコードの違いを吸収 (up $\LaTeX$  と相性悪?)
- 欧文フォント, 数式フォントを Times Roman にする
- 欧文サンセリフ体のフォントを Helvetica にする
- 欧文タイプライタ体のフォントを Coulier にする

です。

この辺の話は美文書作成入門等の方が圧倒的に詳しいので、そちらをご覧ください。

## 第10章 和文フォントについて

この章では和文フォントについて扱っていかうと思います。T<sub>E</sub>X の標準フォントがきもい、フォントを色々変えてみたい、なんて人も少なくないと思います。pxchfon パッケージを使いこなせばフォントは自由自在に変えられるので、楽しんでいきましょう。

### 10.1 和文フォントについて

まずはさらっと和文フォントについての一般的なことを書いておこうと思います。

和文フォントには大きく分けて2種類、明朝体とゴシック体があります。明朝体は本文によく使われるフォントで、例えば線の右端に三角形の飾り（ウロコ）がついているものがそうです。欧文文字のセリフ体に対応します。ゴシック体はタイトルなどによく使われるフォントで、こちらは飾りがなく均一な太さの線で書かれているものです。欧文文字のサンセリフ体に対応します。使い分けは述べた通りですが、読みやすさからゴシック体を本文に使うこともあります。

フォントの中には、サイトでの公開や商用利用の際に年間契約をしなければならないものも多いです。フォントを変える際にはしっかりその辺を調べてからの方が安全ですので覚えておいてください。

### 10.2 書体の切り替え

文字を明朝体またはゴシック体にするには、

明朝体       :   \textmc{文字}   or   {\mcfamily 文字}

ゴシック体   :   \textgt{文字}   or   {\gtfamily 文字}

とします。...\family と書体を変えたい文字を{}でくくらないと、それ以降の文字全てがその書体になってしまうので注意してください。

また、\textbf や\bfseries でも和文文字はゴシック体になります。ただ、これだと**あ A**のように欧文文字がセリフ体のままとなり、デザインとしての一貫性が崩れてしまいます。

これに追加で\textsf や\sffamily を使えば、**あ A**のように欧文文字もちゃんとセリフ体となります。

また、書体の切り替えは上に挙げた\text... と...\family のどちらでもできますが、いまは\text... を使うのがよいとされていますので、こちらを使うようにした方がよいかと思います。

### 10.3 和文フォントを変える

和文フォントを変えるやり方について気が向いたら説明しようと思います。ただ、次節のpxchfonを使うやり方の方が圧倒的に簡単ですので、そちらをおすすめします。

## 10.4 pxchfon パッケージを使う

和文フォントの設定に便利な pxchfon というパッケージがあります。まずはプリアンブルに

```
\usepackage{pxchfon}
```

と書いてください。

### 10.4.1 基本的な使い方（単ウェイトの場合）

まずは基本的な使い方を学んでいきましょう。なお、タイトルにある「単ウェイト」とは、例えば太字（ゴシック体）の中に極太、中太などの太さの違いがない、くらいの意味だと思っておいてください。

パッケージの読み込みの際につけるオプションは色々ありますが、和文フォントのみを変える場合が多いと思うので、noalphabet を指定して

```
\usepackage[noalphabet]{pxchfon}
```

としておけばとりあえず大丈夫です。

使い方ですが、非常に簡単で、プリアンブルに

```
\setminchofont{フォントのファイル名}  
\setgothicfont{フォントのファイル名}
```

と書くだけで、それぞれ元の明朝体、ゴシック体だった部分が指定したフォントになります。

フォントのファイルは、windows であれば windows フォルダ内の fonts というフォルダに入っていると思います。拡張子は「.ttf」,「.ttc」,「.otf」のどれかだと思います。これらの拡張子はそれぞれ、True Type Font, True Type Collection, Open Type Font を約めたものです。これらの意味が気になる方は是非調べてみてください。

例を挙げてみます。例えば次をプリアンブルに書けば、明朝体が「IPA 明朝」、ゴシック体が「IPA ゴシック」になります（デフォルトと変わりません）。

```
\setminchofont{ipam.ttf}  
\setgothicfont{ipag.ttf}
```

ただ、ttc ファイルは少し異なります。ttc は、その名前からわかるように、True Type のフォントの Collection となっています。よって、そのコレクションの中のどのフォントを使うかまで指定する必要があります。これは、\setminchofont や \setgothicfont コマンドのオプションで指定します。例えば、msgothic.ttc には「MS ゴシック」と「MSP ゴシック」が含まれ、それぞれの番号は 0 と 1 です。よって、例えばゴシック体を「MS ゴシック」にしたければ、

```
\setgothicfont[0]{msgothic.ttc}
```

とします。

これで単ウェイトでフォントを変えることはできるようになりました。大体の文書は明朝体とゴシック体が 1 つずつあれば（つまり単ウェイトでも）作れるので、ここまでの説明で十分な人も少なくないと思います。色々フォントを変えてみて下さい。

### 10.4.2 多ウェイトの場合

次は多ウェイトの指定方法について説明しようと思います。単ウェイトの説明からわかると思いますが、多ウェイトとは明朝体、ゴシック体の中に太さの違いがあるフォントが複数含まれるといったような意味で

す。TEX では、明朝体とゴシック体それぞれに 3 ウェイトずつと丸ゴシックの計 7 つまで設定することができます。

多ウェイトにするにはまず、otf パッケージを deluxe オプション付きで、pxchfon の前で読み込みます。つまり、次のようにするということです。

```
\usepackage[deluxe]{otf}
\usepackage[noalphabet]{pxchfon}
```

それぞれの書体、ウェイトの設定は、次のようなコマンドで行います。

```
\setlightminchofont[<番号>]{<フォントファイル名>} : 明朝・細ウェイト (\mcfamily\ltseries)
\setmediumminchofont[<番号>]{<フォントファイル名>} : 明朝・中ウェイト (\mcfamily\mdseries)
\setboldminchofont[<番号>]{<フォントファイル名>} : 明朝・太ウェイト (\mcfamily\bfseries)
\setmediumgothicfont[<番号>]{<フォントファイル名>} : ゴシック・中ウェイト (\gtfamily\mdseries)
\setboldgothicfont[<番号>]{<フォントファイル名>} : ゴシック・太ウェイト (\gtfamily\bfseries)
\setxboldgothicfont[<番号>]{<フォントファイル名>} : ゴシック・極太ウェイト (\gtfamily\ebseries)
\setmarugothicfont[<番号>]{<フォントファイル名>} : 丸ゴシック (\mgfamily)
```

また、このとき、\setminchofont と \setgothicfont はそれぞれ明朝体、ゴシック体の 3 ウェイト全てのフォントを指定したフォントにします。例えば、中ゴシックは使わず、太ゴシックと極太ゴシックの 2 ウェイトでいいというときは、

```
\setgothicfont{GenShinGothic-Bold.ttf}
\setxboldgothicfont{GenShinGothic-Heavy.ttf}
```

のようにして指定することもできます。

### 10.4.3 様々なオプション

パッケージを読み込むときに付けることのできるオプションについて説明していきます。

- **alphabet** : 欧文フォントも指定したフォントで置き換える。rmfamily が明朝体、sffamily がゴシック体に対応する。
- **noalphabet** : alphabet の否定。欧文フォントは置き換えない。
- **oneweight** : 小塚フォントで OTF を単ウェイトで用いるときに指定する。詳しくはドキュメント参照のこと。
- **プリセット指定** : フォントを一括で指定する。具体的なオプション名やフォント名は下で扱う。

プリセット指定について説明しようと思います。これは、よく使う設定をオプションで設定できるようにしてくれている、というものです。

例えば、本文を、「欧文フォントは置き換えず、和文フォントのみ明朝体もゴシック体も単ウェイトの IPA フォントにする」、というのはよくある設定です。これをオプションで一括設定することができます。具体的には、

```
\usepackage[ipa]{pxchfon}
```

とするだけで、

```
\usepackage[noalphabet]{pxchfon}
\setminchofont{ipam.ttf}
\setgothicfont{ipag.ttf}
```

と打つのと同じ効果を得られます。

このような一括設定が用意されているフォントは他にも色々あります。以下にオプションとフォント名だけ全て列挙しておきます。明朝，ゴシックと各ウェイトでそれぞれのフォントが使用されるのかはドキュメントにすべて書いてあるので，これを使うときは是非見ておいてください。

単ウェイト用のオプションは以下の通りです。

- noembed : フォントを埋め込まない
- ms : MS フォント
- ipa : IPA フォント
- ipaex : IPAex フォント

多ウェイト用のオプションは以下の通りです。なお，これらを使用するときは，otf パッケージを deluxe オプションを付けて読み込むのを忘れないでください。

- ms-hg : MS フォント + HG フォント
- ipa-hg : IPA フォント + HG フォント
- ipaex-hg : IPAex フォント + HG フォント
- moga-mobo : Moga フォント + Mobo フォント
- moga-mobo-ex : MogaEx フォント + MoboEx フォント
- moga-maruberi : Moga フォント + モトヤ L マルベリ 3 等幅
- kozuka-pro : 小塚フォント (Pro 版)
- kozuka-pr6 : 小塚フォント (Pr6 版)
- kozuka-pr6n : 小塚フォント (Pr6n 版)
- hiragino-pro : ヒラギノフォント基本 6 書体セット (Pro/Std 版) + 明朝 W2
- hiragino-pron : ヒラギノフォント基本 6 書体セット (Pro/StdN 版) + 明朝 W2
- hiragino-elcapitan-pro : ヒラギノフォント (Mac OS X El Capitan 搭載; Pro/Std 版) + 明朝 W2
- hiragino-elcapitan-pron : ヒラギノフォント (Mac OS X El Capitan 搭載; Pro/StdN 版) + 明朝 W2
- morisawa-pro : モリサワフォント基本 7 書体 (Pro 版)
- morisawa-pr6n : モリサワフォント基本 7 書体 (Pr6N 版)
- yu-win : 游書体 (Windows8.1 搭載版)
- yu-win10 : 游書体 (Windows10 搭載版)
- yu-osx : 游書体 (Mac OS X 搭載版)

以上に挙げた以外にも様々なオプションがありますが，普通に使うならこのくらいで大丈夫です。もっと細かいところまで設定したい人は是非ドキュメントを読んでみて下さい（日本語なので普通に読めます）。

#### 10.4.4 注意点

pxchfon パッケージを使用する際の注意点を列挙しておきます。

- ドライバは dvipdfmx を用いる
- 等幅のフォントのみが使用可能である
- otf パッケージは pxchfon の前で読み込む

また，以上ではフォントを埋め込まない場合については扱っていません。もしフォントを埋め込みたくない場合は，検索したりドキュメントを参照したりしてください。

このパッケージはフォントの埋め込みまでしてくれるみたいなので本当に便利です。map ファイルを用意したりする必要がなくなるので，圧倒的に簡単ですね。

また，この部分を書くのに，「[PXchfon パッケージ～pLaTeX 文書のフォントを簡単に変更～](#)」を参考にしました。

## 10.5 色々なフォント

蛇足ですが色々なフォントを随時紹介していこうと思います。

商用利用できるものは <https://fontbear.net/> というサイトにまとめられているので是非見てみて下さい。

## 第11章 ファイルの分割と統合

文書の量が多くなると、どこに何を書いてあるのかわかりづらくなり、1つの $\text{\TeX}$ ファイルで管理することが難しくなります。そこで、1つの文書を小分けにして別々で管理し、また別のファイルで統合する、という方法を紹介しておこうと思います。

### 11.1 ディレクトリとパス

ファイルを分け、統合するにあたって、ディレクトリとパスというものを知っておく必要があります。以下の小節でそれぞれ説明します。知っている人は読み飛ばしてもらって構いません。

#### 11.1.1 ディレクトリ

ディレクトリとは、色々なファイルを入れてあるフォルダのことです。ファイルを人だと考えれば、ディレクトリは家のようなものです。ディレクトリは入れ子（多層構造）にすることができます。例えば、日本というディレクトリに東京都というディレクトリがあり、その中に23区のディレクトリがあり、その中に本郷三丁目というファイルがある、といった感じです。このように、層の下に入っていく（狭い範囲になっていく）ことを「下に行く」、逆に広い範囲を含む方へ行くことを「上に行く」ということにします。

#### 11.1.2 パス

パスとは、そのファイルがコンピューターのどこにあるか、という住所のようなものです。パス（住所）を指定することで、目的のファイル（人）を探し出して使うことができるようになります。

パスには「絶対パス」と「相対パス」の2種類があります。絶対パスは、一番大元のディレクトリからスタートして指定するものです。住所に例えれば、

宇宙 → 地球 → ユーラシア大陸 → アジア → 日本 → 関東 → 東京 → 文京区  
のような感じです。例えば、僕のPC上でこの $\text{\TeX}$ マニュアルは、

`C:/Users/User/Document/TeX/TeXManual`  
というところにあります。この「Users」が、僕のPCの大元のディレクトリとなっています。

相対パスは、今いるところからの相対位置で指定するものです。住所に例えれば、東京都から東京都に手紙を出すときに、「東京都」の部分省略して区名から書けばOK、のような感じです。相対パスを指定するとき、同じディレクトリにあるファイルを指定するときはただそのファイル名を書けばよく、今いる（使っているファイルのある）ディレクトリより下のディレクトリのファイルを指定するときは、「ディレクトリ名/ファイル名」とし、今いるディレクトリより上のファイルを指定するときは「../ファイル名」とします。上や下に行った数だけ「/」が書かれることになります。



ここで、windows ではパスの区切りが円マーク (¥) やバックスラッシュ (\) となっていますが、 $\text{\TeX}$  でのパスの区切りは全てスラッシュ (/) ですので、注意してください。

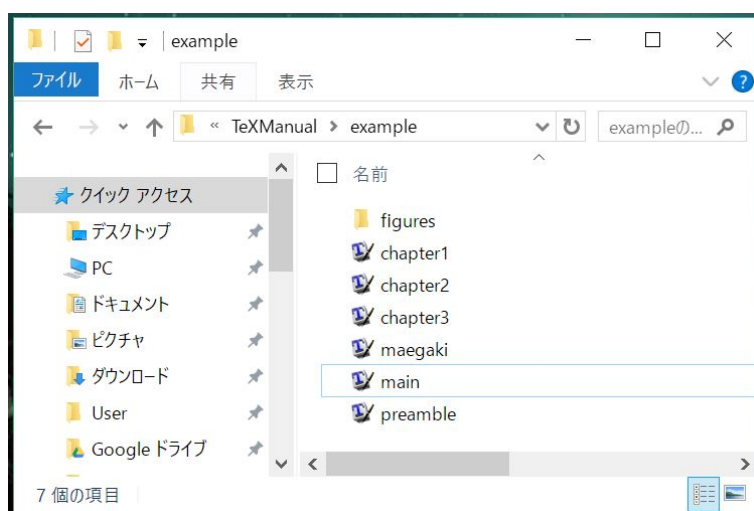
## 11.2 $\text{\TeX}$ 標準のコマンド

$\text{\TeX}$  には、`\input{.tex ファイル}` と `\include{.tex ファイル}` という 2 つのコマンドが用意されています。この 2 つはともに、元の `.tex` ファイルのそのコマンドを、指定された `.tex` ファイルの内容で置き換える、というものとなっています。違いとしては、`\include` は改ページあり、`\input` は改ページ無し、となっています。ちなみに、ファイル名の `.tex` は省略できます。

## 11.3 具体例

色々解説してきましたが、実際どのようにするのか見ないとかなりわかりづらいと思うので具体例を示します。

前書き、目次、第 1~3 章から構成された 1 つの文書を異なるファイルに分割し、統合することを考えます。まずはフォルダを 1 つ作って、前書き (`maegaki.tex`)、第 1~3 章 (`chapter1~3.tex`) と、文書に関連するファイルを全てその中に入れます (ここではフォルダ名を `example` とします)。図があるときはまとめて `figures` フォルダにでも入れておきましょう。また、プリアンブル (`\documentclass` と `\begin{document}` の間) も分けて別の `.tex` ファイルに書きます (`preamble.tex` とします)。最後に統合用の `.tex` ファイル (`main.tex`) も同じ所に置いて準備完了です。いま、`example` フォルダの中身は以下のようになっています。



さて、次は `main.tex` の中身を見てみましょう。`\tableofcontents` は目次を出力するコマンドであることに注意してください。

```

\documentclass[dvipdfmx, a4paper]{jsarticle}
\include{preamble}
\begin{document}
\tableofcontents
\include{maegaki}
\include{chapter1}
\include{chapter2}
\include{chapter3}
\end{document}

```

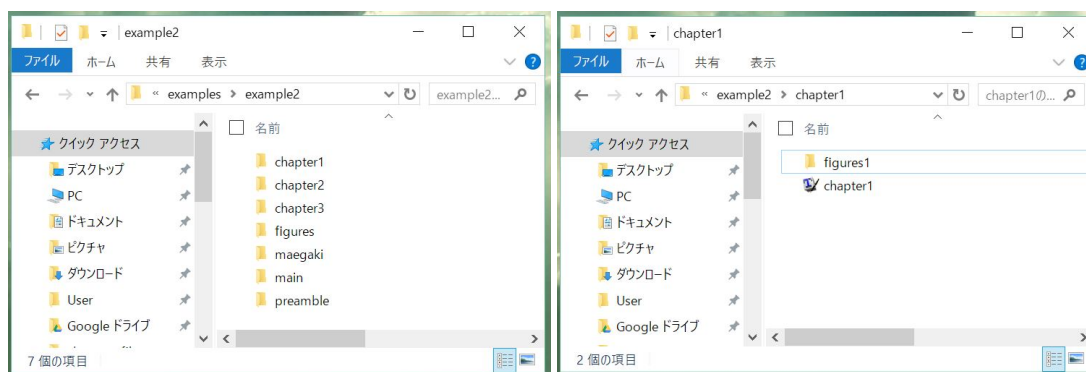
main では分割したファイルを\include を使って統合しています (\include は改ページを伴うので、改ページしたくないときは\input を使ってください)。これで main を実行すれば、これらを全部一纏めの.tex ファイルに書いたときと同じ出力が得られます。

なお、前節で説明した通り、\include や\input は読み込んだ.tex ファイルをそこに書き写すだけなので、読み込むファイル (preamble, maegaki, chapter1~3) には\documentclass~\begin{document}, \end{document} は書かないでください。

この文書を書いている途中は、chapter3 が用意されていないかったりすると思います。また逆に、chapter2 を書いているときは、目次や chapter1 は不要になるかもしれません。そのときは、\include コマンドの前に%をつけてコメントアウトしてください。

## 11.4 import パッケージの利用

上では.tex ファイルを全て同じフォルダに入れています。図が多かったりするときは章ごとにフォルダを作り、その中に図のフォルダを作ることもあるかもしれません。そのとき、フォルダの中身は例えば以下になるでしょう。



このとき、例えば chapter1 を main で\include するには、chapter1.tex のパスを指定しなければなりません。このパスは絶対パスでも相対パスでもいいですが、main からの相対パスで指定する方がよいと思います。これは、絶対パスで指定すると、後々 main のフォルダごと移動するときに全てのパスの指定を書き直さなければならなくなるからです。よって、main.tex では、

```

\include{../chapter1/chapter1.tex}

```

のようにして読み込む形になります (「../」は1つ上のディレクトリに移動することを表します)。

しかし、もし `chapter1.tex` の中で `\includegraphics` を使って図を読み込んでいるなど、他のファイルを読み込んでいる場合、それがそのまま `main.tex` に書き込まれる形になるので、`chapter1.tex` で書いた `\includegraphics` の相対パスがずれてしまい、結果図のファイルが見つからずエラーしてしまいます。

このとき、`\include` で `chapter1` を読み込んでいる場合、よくわからないエラーが出ます（多分解決不可です）。

もし `\input` で読み込んでいる場合、「!LaTeXerror file not found」的なエラーが出ます。こちらは一応解決可能で、`chapter1.tex` 内で他のファイルを読み込んでいるところ（`\include`、`\input`、`\includegraphics` を使っているところ）のパスの指定を `main.tex` からの相対パスに変更すれば解決できます。しかしこの方法ではわかりづらい上、`main` を移動したりファイル名を変更したりしたときに色々なところのパスをいじらなければならず非効率的です。

よって、 $\text{\TeX}$  標準のコマンドだけでは、フォルダ分けをしてファイルを管理する、つまり `\input` 等の入れ子を扱うのが非常に面倒です。

そこで、`import` という便利なパッケージを使います。まずは `\usepackage{import}` とプリアンプルに書きます。この `import` では、

```
\import{絶対パス}{ファイル名.tex}
\subimport{相対パス}{ファイル名.tex}
```

という2つのコマンドが定義されています。この2つのコマンドで `chapter1.tex` を読み込むと、`chapter1.tex` 内での `\input` 等は、`chapter1.tex` から見たパスとして扱われます。したがって、`main` から見た相対パスを考えてパスを書く、といった面倒なことはなくなります。

また、これらのコマンドにはそれぞれ、`\inputfrom`、`\subinputfrom` という別名が用意されています。なお、`\import` と `\subimport` の2つを同じファイル内で使うとエラーするので気を付けてください。

この `import` パッケージを使えば、`main.tex` の中身は以下ようになります。

```
\documentclass[dvipdfmx, a4paper]{jsarticle}
\input{../preamble/preamble}
\begin{document}
\tableofcontents
\subimport{../maegaki/}{maegaki}
\subimport{../chapter1/}{chapter1}
\subimport{../chapter2/}{chapter2}
\subimport{../chapter3/}{chapter3}
\end{document}
```

これで、例えば `chapter1.tex` 内で

```
\includegraphics{figures/fig1.pdf}
```

のように、そこからの相対パスで図を読み込んでもちゃんとファイルを見つけてくれます。

この `import` パッケージはファイルの分割時には非常に便利なので、長い文書を書くときは是非使ってみて下さい。

## 11.5 分割したファイルを実行する

ここまでで、ファイルを分割してそれをまとめることはできるようになりました。ただ、これだけだとちょっと不便です。例えば、ファイルの一部を書き換え、それを出力して修正するとき、その実行

は `main.tex` を用いて行わなければなりません。このとき、他のファイルごと実行すると実行時間がかかりかかりますが、他のファイルを読み込んでいる `\input` などをコメントアウトするのも面倒です。そこで、分割したファイルを実行するやり方を紹介しようと思います。

### 11.5.1 docmute パッケージ

`docmute` パッケージを用いれば、`\input` と `\include` されるファイルの `\begin{document}` と `\end{document}` の外を無視し、その中身だけを読み込むことができます。まずはプリアンブル (`preamble.tex`) に

```
\usepackage{docmute}
```

と書いてください。ただ、この `docmute` パッケージはインストールされていない可能性もあります。その場合は、第 13 章 13.1 節を見て自分でインストールしてもよいですが、後の `standalone` パッケージがほぼ `docmute` の上位互換となっているので、そちらを使った方がよいと思います。とりあえずこれはシンプルなパッケージなので、わかりやすい説明のために取り上げておきます。

使い方を説明します。`main.tex` の中身が以下になっているとします。

```
main.tex
\documentclass[dvipdfmx, a4paper]{jsarticle}
\input{../preamble/preamble}
\begin{document}
\include{../chapter1/chapter1}
\include{../chapter2/chapter2}
\include{../chapter3/chapter3}
\end{document}
```

また、`chapter1.tex` の中身が以下になっているとします。

```
chapter1.tex
\documentclass[dvipdfmx, a4paper]{jsarticle}
\input{../preamble/preamble}
\begin{document}
\chapter{はじめに}
はじめに~~~~~。

\end{document}
```

まずは `chapter1.tex` のみを実行し、内容を確認します。内容が OK になったら、`main.tex` を実行します。そうすると、`\begin{document}` と `\end{document}` の間だけが `main` で読み込まれるので、ファイルの統合もうまくいきます。

ここで注意ですが、プリアンブル部分は無視されるので、`main.tex` と `chapter1.tex` のプリアンブルが異なっているとそれぞれの実行結果が変わる可能性があります。できるだけ同じプリアンブルを使うようにしてください。また、`\include` と `\input` は使えますが、`\import` や `\subimport` は使えないので注意してください。

### 11.5.2 subfiles パッケージ

`subfiles` パッケージを使えば、`chapter1.tex` 等において親の `main.tex` のプリアンブルを引き継いで使うことができます。まずはプリアンブル (`preamble.tex`) に

```
\usepackage{subfiles}
```

と書いてください。

使い方ですが、`chapter1.tex` 等のドキュメントクラスに `subfiles` を指定し、オプションで `main.tex` のパスを指定してください。また、読み込みは `\subfile` コマンドで行ってください。具体的には、以下のようになります。

```
main.tex
\documentclass[dvipdfmx, a4paper]{jsarticle}
\input{../preamble/preamble}
\begin{document}
\include{../chapter1/chapter1}
\include{../chapter2/chapter2}
\include{../chapter3/chapter3}
\end{document}
```

```
chapter1.tex
\documentclass[../main/main.tex]{subfiles}
\begin{document}
\chapter{はじめに}
はじめに~~~~~。

\end{document}
```

こうすることによって、子の `chapter1.tex` で `main.tex` のプリアンブルを読み込んで使うことができます。プリアンブルを共有できるので、いちいち子のファイルでプリアンブルを書いたり読み込んだりする必要がなくて便利です。

ただ、`\import` や `\subimport` に対応するコマンドは用意されていないので注意してください（僕が試したときは、`chapter1.tex` 等で `\input` を使うとテキストのみ読み込まれ、`tcolorbox` や `tikz` が表示されないという形になりました）。

なお、この小節は「[分割した LaTeX ファイルを subfiles を使ってコンパイルする](#)」を参考にしました。

### 11.5.3 standalone パッケージ

`standalone` パッケージを使えば、子のプリアンブルを親に共有することができます。まずは、`subpreambles` オプションを追加して、

```
\usepackage[subpreambles=true]{standalone}
```

としてください。

使い方ですが、`main.tex` での読み込みは `\input` や `\subimport` で行えます。`main.tex` の中身は上と同じとしますので、省略します。子のファイルについてですが、これは以下のようにします。

```
chapter1.tex
\documentclass[class=jsarticle, crop=false, preview=false, dvipdfmx,
a4paper]{standalone}
\input{../preamble/preamble}
\newcommand{\kodomono}{子供}
\begin{document}
\chapter{はじめに}
はじめに~~~~~。
\kodomono

\end{document}
```

ドキュメントクラスには `standalone` を指定し、オプションで `class=jsarticle` 等とします。jsarticle 等につけるオプション (`dvipdfmx` や `a4paper` 等) も普通につけることができます。 `crop=flase` や `preview=false` ですが、僕はよくわからないまま使っているのでは是非自分で調べてみて下さい。

上の例では、`chapter1.tex` のプリアンブルで `\kodomo` というコマンドを定義し、文章中でこれを使っています。 `docmute` や `subfile` ではこれは `main.tex` には読み込まれないので、`main.tex` を実行するとエラーしてしまいます。しかし、`standalone` を使えば、子のプリアンブルが親にも共有されるので、`main.tex` を実行してもエラーしません。他の子のファイルと共有しないようなプリアンブルは、共有のプリアンブルでなく子のそれに書いてもよいでしょう。

`standalone` パッケージでは、`\input` や `\include` に加え `\import` や `\subimport` も使えるので、これらを使いたい人は `standalone` パッケージを使ってみて下さい。

ここで注意ですが、子のプリアンブルが親のプリアンブルに共有されるので、`preamble.tex` の相対パスが `main.tex` からと `chapter1.tex` から異なる場合はエラーします。絶対パスで指定してもいいですが、特に子のプリアンブルを親に共有しなくてもいい場合は、`\usepackage{standalone}` のように、`subpreambles` オプションを外してください。

この `standalone` パッケージは元々 `tikz` という描画パッケージを読み込むときに便利なものです。僕はその用途で使っているわけではないのであまりそちらには詳しくありません。そちらの用途で使うとき用のオプションやコマンドが用意されているので、是非自分で調べてみて下さい。

この小節は、「[Multi-file LaTeX projects](#)」を参考にしました。

#### ※エラーのメモ

`\usepackage[subpreambles=true]{standalone}` としているとき、コンパイル時に `.sta` ファイルが生成される。この `.sta` ファイルには子のプリアンブルが書き込まれ、それが `main` で読み込まれるという仕様になっているようだ。ただ、この `.sta` ファイル、コンパイルエラーが起きたときに `\endstandalonepreambles` というコマンドを書き込まずに閉じてしまい、その次のコンパイルから「1543. }」で始まるエラーが毎回起こるようになる。`.sta` ファイルを削除するか `\endstandalonepreambles` を `.sta` ファイルの最後書き足せばエラーはなくなるが、これではかなり面倒である。当面、`import` が使えるパッケージとしての有用性を発揮してもらうだけになりそう。

上の全てのパッケージを通して、子の `chapter1.tex` 等で使った他ファイルへの参照となる `\ref` は「??」等となりうまく表示されません。`main` においてはちゃんと表示されるので基本的にはそこまで気にしなくて大丈夫だと思いますが、子の PDF をそのまま使う可能性がある場合は注意してください。

## 第12章 正しくきれいな文書の書き方

この章では、数式や日本語、英語の正しい書き方やきれいな書き方を紹介していきます。物理、数学寄りが多いです。

### 12.1 出版物の常識

出版物には色々な常識、テンプレートがあります。主に文字サイズや空白などです。その辺を軽く触れておこうと思います。

まず、文字サイズは「14Q」というのが一般的です。これは documentclass のオプションに「14Q」とつければそうなるので、細かいところまで拘る人は是非。

また、「四分空き」というものがあります。これは、半角文字と全角文字の間に、全角の 1/4 くらいの幅のスペースを入れるというものです。この設定は、`\xkanjiskip` をいじって設定します。

```
\xkanjiskip 0.25zw plus 0zw minus 0zw
```

とすればいいと思います。ただ、これでは文尾がバラバラになってしまう可能性があるので、

```
\xkanjiskip 0.25zw plus 0.125zw minus 0.125zw
```

などとしても良いと思います。

また、細かいのですが、日本語の文章における句読点についても触れておこうと思います。句点は縦書き横書き共通で「。」なのですが、読点は縦書きでは「、」、横書きでは「, (全角コンマ)」とするのが標準となります。細かいところまで気にする人は是非気を付けてみて下さい

### 12.2 数式中の文字の字体

物理の話になってしまいますが、数式の中で、物理量と非物理量の字体について扱います。ここを間違っている人は多いので、是非目を通して見て下さい。

#### 12.2.1 物理量

物理量とは何かというと、単位を持っていて（無次元数もありますが...）、何かしらの式で表される、物理的な量です。具体的に例を挙げれば、長さ  $x$  とか力  $F$ 、圧力  $P$  などのようなものです。

この物理量は、数式中ではイタリック体（斜体）で書くのが標準です。 $x$  は自然ですが、 $x$  は違和感ありますよね。TeX では数式中に入れたアルファベットは自動でイタリック体になるので、これに注意して書く、ということはありません。



### 12.2.2 非物理量

では何に注意するかというと、非物理量です。これは、具体的に言えば物体 A の「A」などです。これは別に物理的な量ではないので、イタリック体にするのは間違いです。

数式中でもこれらの非物理量はローマン体（立体、普通の形）で書きます。例えば、終端速度  $v_f$  の「f」は final の f なのでローマン体ですし、理論熱効率の  $\eta_{th}$  の「th」も theory からきているので、これもローマン体です。添え字の多くはその変数を特徴づけるためのものであり、非物理量であることが多いです。ですの、添え字をつけるときは、`\mathrm{}` というコマンドでローマン体に直すのに気をつけましょう。

なお、中には定圧比熱  $C_p$ 、定積比熱  $C_V$  のように物理量が添え字になっているものもあるので、添え字は全部ローマン体だ、というわけではありません。

また、定数を表す文字（ex. バネ定数  $k$ 、プランク定数  $h$  など）についてですが、これは変化しない物理量を表しているの、イタリック体で大丈夫です。

ここで、勘のいい人は、座標軸を表す  $x, y, z$  や、微分を表す  $d$  は物理量ではないから、ローマン体で書かなければならないのか、と思うかもしれません。これらに関しては、もちろん非物理量ですのでローマン体で書くのが正しいのですが、慣習的にイタリックで書いてしまっているので、それでも大丈夫です。文書内で統一されていれば OK なので、好きな方を使ってください。

## 12.3 単位

長さなどの単位には、国際標準化機構により標準が定められています。ここでは、それに則った単位の書き方について紹介します。

前節の内容から、単位は物理量を表してはいないので、ローマン体で書くのが正しいです。割と多くの人がイタリック体で書いてしまっているので、是非とも注意してください。

文字式と数値式で書き方が異なるので、別々に紹介していきます。

### 12.3.1 文字式の単位

文字式の単位については、

- 式との間にスペースを入れない
- `[]`（亀甲括弧）でくる
- ローマン体で書く

の 3 つの約束を守ればよいです。

### 12.3.2 数値につける単位

数値につける単位は、

- 式との間に半角スペースをいれる
- 括弧はつけない
- ローマン体で書く



の 3 つの約束を守って書いてください。数値に括弧付きで単位を付けている人は少なくないので、注意してください。例えば、わかりづらいかもしれませんが、

1.0 m, 1.0 kg

のようにする、ということです。

ただ、角度を弧度法で表すときの「°」との間にはスペースは不要です（例：60°）。また、%や分など、単位ではなく記号とみなされるものとの間にもスペースは不要です。そのまま 60 分や 25% などと書いてください。

### 12.3.3 siunitx パッケージを使う

上記のように、単位の付け方には様々な制約があります。これを全部守って書こうとすると、 $(2.0 \times 10^{-3} \text{ m}) \times (5.0 \times 10^3 \text{ m/s})$  と打つだけで、

```
(2.0\_\times\_10^{\{-3\}}\_\\\_\\mathrm{m})\_\\\_\\times\_ (5.0\_\times\_10^{\{3\}}\_\\\_\\mathrm{m/s})
```

のようになって、式が長くなり見通しが悪くなります。また、括弧の数も多く、ここに分数を入れようものならまずエラーするでしょう。これを簡単にするために、 $\text{\TeX}$  には siunitx というパッケージが用意されています。使い方は、

```
\SI{数値}{単位}
```

です。これだけで、数値の後に適切なスペースが入り、単位がローマン体で出力されます。これなら式が不用意に長くなりすぎず、エラーも起こしにくいです。

また、数値の部分は、電卓のような書き方をすることもできます。具体的に例を見ると、

```
\SI{2.0e-3}{m}
```

とすれば、 $2.0 \times 10^{-3} \text{ m}$  のように、e の後の数字が 10 の指数となって現れます。なお、この数字に括弧をつけるとエラーするので気を付けてください。

また、単位だけを出力したければ、

```
\si{単位}
```

というコマンドも用意されています。

なお、 $\text{\SI}$  コマンドの 2 つ目の引数（単位の方）に記号系の単位（%や分）を入れてもスペースは入ってしまうので、そこは自分で打ってください。

この siunitx は、他にも様々なことができますが、ここではとりあえず  $\text{\SI}$  と  $\text{\si}$  の紹介にとどめておきます。他の便利なコマンドについては、各種パッケージについての部を見てください。

## 12.4 文章中の数式

文章中（インライン）の数式についてです。といっても、背の低い数式（分数を含まないなど）はそのままきれいに出力されるので、あまり問題はありません。問題となるのは、分数や積分記号を含む、背の高い数式です。これらを文章中に入れると、通常は小さくなって見づらくなります。そこで、普通の大きさを表示するために  $\text{\displaystyle}$  等を用いると、今度はその上下の行に干渉して全体のバランスが悪くなります。

インラインに分数を入れる場合は、「/」で分数を表して、  
「コインを投げて裏が出る確率は  $1/2$  であるが...」  
などとするのが一般的です。

ただ、これでは複雑な数式は表現しづらいし読みづらくもあると思う人もいるかもしれません。その場合は `\dfrac` や `\cfrac` などを使っていいただければ、大きく分数が表示されて読みやすくなると思います。ただ、そのままでは上下の行と近づきすぎて読みづらくなるので、`\lineskiplimit` と `\lineskip` を設定するのがよいと思います。そちらについては第 8 章 8.5 節の行送りについての部分を読んでください。

## 第13章 各種パッケージの使い方について

ここまで、`fancthdr` や `hyperref`, `import` など便利なパッケージを紹介してきましたここでは、まだ紹介していない便利なパッケージについて説明したいと思います。

### 13.1 パッケージのインストールとファイルの置き場所

$\mathrm{T}_{\mathrm{E}}\mathrm{X}$  には色々なパッケージがありますが、中には最初にダウンロードした中に入っていないものもあります。こういったものは、どこからダウンロードしてくるしかありません。まずは「パッケージ名 ダウンロード」とでも検索して、ダウンロードしてください。

ダウンロードした `zip` ファイルの展開をすると、欲しいパッケージのファイル、`.sty` ファイル（スタイルファイル）が手に入ります。これが、パッケージの本体となります。

しかし、ただダウンロードと展開をただけでは、 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  がこの `.sty` ファイルを見つけることができず、パッケージが見つからないといったエラーが出ます。よって、この `.sty` ファイルを、 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  がつけられるところに置く必要があるのです。

この、「 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  がつけられるところ」には色々ありますが、一番簡単なのは、そのパッケージを使う `.tex` ファイルと同じフォルダに入れることです。

ただこれでは何度も使うときに面倒だし美しくないです。 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  が探してくれるディレクトリに `.sty` ファイルを置けば万事解決です。そのディレクトリは、`windows` では、`w32tex` ディレクトリ内の、「`C:/w32tex/share/texmf-dist/tex/latex`」です。なお、 $\mathrm{p}_{\mathrm{L}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$  専用のもは、最後のディレクトリを「`platex`」にしてください。ここに適当なディレクトリを作って、その中に `.sty` ファイルを置いていけばよいでしょう。

また、`texmf-local` 直下に「`ls-R`」ファイルがあるときは、コマンドプロンプトで `mktexlsr` を実行してください。

この、パッケージが置いてあるディレクトリは、例えば `amsmath` なら、コマンドプロンプトで

```
kpsewhich amsmath.sty
```

を実行すればわかります。頑張って探してみてください。

### 13.2 siunitx

単位をつけるのに便利な `\SI` コマンドが入っている `siunitx` パッケージですが、他にもいろいろ便利なコマンドがあるので紹介しておきます。

#### 13.2.1 単位自体について

まずは単位についてのお話です。単位は、`\metre` や `\gram` といったコマンドでも出力できます。また、「 $^{-1}$ 」を表す「`/`」は `\per` で出すことができます。また、`m2` は、`m^2` でも出せますし、

`\square\metre` や `\metre\squared` ともできます (3 乗は `\cubic` でできます)。単位のコマンドについてはドキュメントにすべて載っているのは是非見てみて下さい。なお、僕には見た目では `\metre` と `m` 等の違いは判りませんでしたので、好きな方を使えばよいと思います。「<sup>-1</sup>」を表す「/」も「/」をそのまま打って大丈夫です。

また、角度についてですが、弧度法で表すとき、通常では

```
60^{\circ}
```

のように打たなければなりません。これは面倒ですが、`siunitx` には便利なコマンドが用意されています。

```
\ang{60}
```

と打てば  $60^\circ$  と出力されます。もし分や秒までいれたいなら、`;` で区切って

```
\ang{60;2;5}
```

とすれば  $60^\circ 2' 5''$  と出力されます。

### 13.2.2 単位の区切り

単位の区切りは、乗算なら「`.`」、除算は「`\per (/)`」で行います。例えば、次のような感じです。

```
\SI{1.0}{kg.m/s^2}
```

(または `\SI{1.0}{\kilo\gram.\metre\per\square\second}`) これは  $1.0\text{kg}\cdot\text{m}/\text{s}^2$  のように出力されます。

この乗算の区切りですが、デフォルトでは半角スペースよりちょっと狭いくらいのスペースです。これを「`\cdot`」にしたい人もいます。そのとき、`\SI{1.0}{kg{\cdot}m/s^2}` などとしてもよいですが、これではスペースがかなり空いてしまう上面倒です。

それを解決するオプションが、「`inter-unit-product`」です。

```
\si[inter-unit-product = \ensuremath{\cdot}]{kg.m}
```

とすれば、 $\text{kg}\cdot\text{m}$  と出力されます。それでもまだスペースが広いような気がするので、僕は `\cdot` の後に `\hspace{-1.5pt}` を入れて使っています。

ただ、これだといちいちそれぞれのコマンドにオプションを付けねばならず、面倒極まりないのではと思う人もいるかも知れません。`siunitx` のコマンドには様々なオプションを付けることができますが、それを一括設定するためのコマンドが用意されています。それが、`\sisetup` です。プリアンブルに、引数にオプションを入れた `\siunitx` コマンドを打つことで、文書全体での `siunitx` のコマンドにそのオプションが適用されます。`hyperref` パッケージなどにも同様のコマンドがありますね。プリアンブルに、

```
\sisetup{inter-unit-product = \ensuremath{\cdot}}
```

と書けば、文書中全ての `siunitx` で付けた単位の区切りが「`.`」になります。

また、除算の区切りについても触れておきます。除算は `\per` で区切りますが、デフォルトでは `\per` を付けた後の 1 つだけに「<sup>-1</sup>」が付きます。例えば次のようになります。

```
\si{J \per mol \per K} \rightarrow J/mol/K
```

これを「/」に変更したいときは、オプションに

```
per-mode = symbol
```

とします。

```
\si[per-mode = symbol]{J \per mol \per K} \rightarrow J/mol/K
```

なお、デフォルトの `per-mode` は「reciprocal」というものなので、一部だけデフォルトのものに戻したいときはそのコマンドのオプションに「`per-mode = reciprocal`」と書いてください。

### 13.2.3 数値の演算

掛け算を出力したいとき、通常ではいちいち `\times` を打たなければなりませんが、`siunitx` 中のコマンド `\num` を使えば、

```
\num{1.2 x 3.2 x 6.4}
```

と打つだけで  $1.2 \times 3.2 \times 6.4$  と出力されます。

この「x」を「\*」に変えたいときは、

```
input-product = *
```

というオプションを付けます。こちらの方が好きな人は是非設定してみてください。

また、分数についてですが、

```
quotient-mode = fraction
```

というオプションを追加することで、 $\num{1/3} \rightarrow \frac{1}{3}$

というように打つことができるようになります。`\num{1.62/2e-4}` のような打ち方も可能です。

これは `\frac` の形式で分数を出力しますが、これを例えば `\dfrac` に変えたければ、

```
fraction-function = \dfrac
```

というオプションを追加すればよいです。

なお、`\num` コマンドの引数の中に「()」は入れられないので（誤差を表す特殊な役割があるため）、() を入れたいときはその中で `\num` コマンドを使う形になると思います。

### 13.2.4 その他

他にも、自動で四捨五入したり、数字の配列を作ったり、単位の色を変えたり、新しい単位を定義をしたり、単位付きの数値の表をきれいにしたり、太字にしたりフォントを変えたりできます。また、細かい設定は `\sisetup` でオプションを指定すれば変えることができます。その辺は是非ドキュメントを読んでやってみてください。

## 13.3 文字を回す

文字通り文字を回します。Qiita を読んでいたらこのパッケージを作った方の記事を見つけて、面白かったのでまとめちゃいました。まずは `tcfaspin` パッケージをダウンロードし、プリアンブルで読み込んでください。

この使い方は、

```
\faSpin{文字}
```

のようにするだけです。これを Adobe Reader などの Adobe 製品で開けば OK です。こうするだけで、

のように文字が回ります。

また、これは入れ子にすることもでき、

のようにもできます。

インライン形式（\$ に挟まれた形）であれば数式も回せますよ

これを有効にするには、2 回実行しなければならないので注意してください。

なお、この部分は、「[これからの時代は LaTeX を回転させよう！](#)」から引用しました。

## 第14章 その他の話題

まだ触れられていない話題の中で僕が知っているものについて、とりあえず列挙と簡単な紹介だけしておこうと思います。今後書き足すかもしれません。

- 索引 : 文書の終わりに索引を付けることができます。
- 引用・参考文献 : 引用マークや参考文献を簡単に付けることができます。論文やレポートを書くときには必須ですね。
- 欧文フォント : 欧文フォント（アルファベット等）についての話です。
- 縦書き :  $\text{T}_\text{E}\text{X}$  で縦書きをすることもできます。小説等を書きたい人などには必須ですね。tarticle や tbook といったドキュメントクラスを使えばすぐにできます。
- マクロの作り方 : コマンドや環境を自分で作ることができます。xparse パッケージを使うとより高度なものが作れると思います。
- listings : listings パッケージについて紹介します。様々なプログラミング言語（含  $\text{T}_\text{E}\text{X}$ ）のソースコードを貼るときに非常に便利です。情報系のレポートや論文を出すときには必須ですね。
- TikZ :  $\text{T}_\text{E}\text{X}$  で図を描きます。簡単な線画ならすぐ描けるようになります。回路図なんかも描けるので非常に便利です。公式ドキュメントがかなり充実しているので、描きたいものをドキュメントから探し、そのあたりの説明を読むことで大体のものは描けます。このマニュアルのタイトルのページもこの TikZ で作りました。図が必要になったときは是非試してみてください。
- tcolorbox : 表において色付きやセルの結合をもっと自由に行いたい場合、tcolorbox パッケージが便利です。日本語のサイトもちろほらありますが、これも公式ドキュメントが充実しているので、そちらを使ってもいいかもしれません。
- スタイルファイル (.sty) の作り方 : スタイルファイル、つまりパッケージを自分で作ることができます。意外と簡単にできるので、よく使うマクロが増えてきたら作ってみるのもいいと思います。

- クラスファイル (.cls) の作り方 : クラスファイル (jsarticle など) を自分で作ることができます。ページレイアウトなどが決まったときはクラスファイルにしましてもよいかもしれません。
- 化学式 : 化学式, 化学反応式や複雑な構造式といったものも書くことができます。これには mhchem パッケージと chemfig パッケージを使います。「[TeX による化学組版](#)」と「[chemfig パッケージによる構造式描画](#)」の 2 つのサイトを読めばかなり多くの化学式を打てるようになります。また, 公式ドキュメントも充実しているので, 是非見てみて下さい。
- オンラインで  $\text{\TeX}$  をする : Cloud $\text{\TeX}$  などを使えば, オンラインでも  $\text{\TeX}$  ができます。
- Python $\text{\TeX}$  : Ruby, Python などのプログラミング言語と  $\text{\TeX}$  を組み合わせることができます。具体的には,  $\text{\TeX}$  中にコードを埋め込んで, それを実行することができます。 $\text{\TeX}$  に用意されているマクロだけでは作れないものも作ることができる可能性を秘めた面白いパッケージです。



## 関連図書

- [1] 奥村晴彦/黒木裕介. 改訂第7版  $\text{\LaTeX}$  2 $\epsilon$  美文書作成入門. 技術評論社, 2017
- [2]  $\text{\TeX}$  Wiki. <https://texwiki.texjp.org/>.
- [3] Kumazawa Yoshiki. <http://www.biwako.shiga-u.ac.jp/sensei/kumazawa/tex/>.
- [4] zr\_tex8r.  $\text{\LaTeX}$  文書で“美しい日本の”ルビを使う～pxrubricaパッケージ～. [http://qiita.com/zr\\_tex8r/items/42466cbcb670a3a2dc](http://qiita.com/zr_tex8r/items/42466cbcb670a3a2dc), 2017.
- [5] あざらし. tcolorboxの基本-物理と $\text{\TeX}$ に関する話題. <http://texmedicine.hatenadiary.jp/entry/2015/12/17/000339>, 2015.
- [6] 水谷正大. ハイパーリンク付き $\text{\LaTeX}$ 文書. [http://www.isc.meiji.ac.jp/~mizutani/tex/link\\_slide/hyperlink.html](http://www.isc.meiji.ac.jp/~mizutani/tex/link_slide/hyperlink.html), 2014
- [7] PXchfon パッケージ～ $\text{\LaTeX}$ 文書のフォントを簡単に変更～. <http://zrbabbler.sp.land.to/pxchfon.html>, 2016.
- [8] 天地有情.  $\text{\LaTeX}$  textpos — テキストブロックを任意の場所に配置. <http://konoyonohana.blog.fc2.com/blog-entry-218.html>, 2016.
- [9] につき : pdfpages. <http://abenori.blogspot.jp/2015/07/pdfpages.html>, 2015.
- [10] sankichi92. 分割した $\text{\LaTeX}$ ファイルをsubfilesを使ってコンパイルする. <http://qiita.com/sankichi92/items/1e113fcf6cc045eb64f7>, 2017.
- [11] Multi-file  $\text{\LaTeX}$  projects. [https://ja.sharelatex.com/learn/Multi-file\\_LaTeX\\_projects](https://ja.sharelatex.com/learn/Multi-file_LaTeX_projects), 2017.
- [12] doraTeX.  $\text{\TeX}$ による化学組版. <http://doratex.hatenablog.jp/entry/20131203/1386068127>, 2013.
- [13] doraTeX. chemfig パッケージによる構造式描画. <http://doratex.hatenablog.jp/entry/20141212/1418393703>, 2014.
- [14] zr\_tex8r. これからの時代は $\text{\LaTeX}$ を回転させよう！. [http://qiita.com/zr\\_tex8r/items/2dbaabff6a795661d413](http://qiita.com/zr_tex8r/items/2dbaabff6a795661d413), 2017
- [15] FONT BEAR. <https://fontbear.net/>.