

好き好き L^AT_EX 2_ε 中級編

渡辺徹

第 0.0.2 版

2007 年 6 月 30 日

Thor Watanabe

Dept. of System Information Science

Future University-Hakodate

thormac(at)gmail(dot)com

<http://mytexpert.sourceforge.jp/>

Copyright © 2005, 2007 渡辺徹

この文書をフリーソフトウェア財団発行の GNU フリー文書利用許諾契約書 (バージョン 1.1 かそれ以降から一つを選択) が定める条件の下で複製, 頒布, あるいは改変することを許可する. 変更不可部分, 表カバーテキスト, 裏カバーテキストは指定しない. この利用許諾契約書の複製物は *GNU Free Documentation License* (GNU フリー文書利用許諾契約書) という章 (付録 A) に含まれている.

本冊子に記載されている企業, 団体の名前や製品名等はそれぞれの権利帰属者の商標または商標登録であり所有物です. 本冊子では ™ 及び ® は明記していません.

駆け出し **L^AT_EX** 使いへ捧ぐ



謝辞

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

履歴

2005年3月23日に正式に『好き好き L^AT_EX 2_ε 中級編』を公開しました。この冊子は『好き好き L^AT_EX 2_ε 中級編』であり、初級編の続編です。初級編を読んでおられないのであれば、まずはそちらをご覧ください。併せて『マクロ活用編』と『周辺ツール編』も先に読んでみてください。

あれっおかしいな?と思ったら

この文書は私1人で執筆しておりますから、どこかに間違いや誤植がある確率が高くなっています。「あれっおかしいな?」と思う箇所がありましたらホームページ、

<http://mytexpert.sourceforge.jp>

のフォーラムかメールアドレス

[thormac\(at\)gmail\(dot\)com](mailto:thormac(at)gmail(dot)com)

にご連絡ください。

目次

謝辞	iii
----	-----

第 1 章 マクロ作成の基礎知識	1
------------------	---

1.1	ゲームのルール	1
1.1.1	L ^A T _E X の基盤	1
1.1.2	T _E X の文法	1
1.1.3	L ^A T _E X の文法	2
1.2	定義と代入	3
1.2.1	定義	3
1.2.2	書式付き引数	4
1.2.3	定義の中の定義	5
1.2.4	グルーピング	6
1.2.5	代入	8
1.3	エンジン始動!	9
1.3.1	文字コード	9
1.3.2	カテゴリーコード	13
1.3.3	文中の特殊記号の扱い	14
1.4	データ型	15
1.4.1	カウンタ	15
1.4.2	長さ	17
1.4.3	スキップ	19
1.4.4	ボックス	20
1.4.5	トークンリスト	21
1.4.6	ブール値	22
1.4.7	アロケーション——変数領域確保	23
1.4.8	\setbox0 とは	23
1.4.9	数値表現	24
1.5	演算とか	25
1.5.1	四則演算	25
1.5.2	柔軟な演算——calc	27
1.6	条件判断	27

1.6.1	トークンの判断	27
1.6.2	整数の判断	29
1.6.3	長さの判断	30
1.6.4	常に真、常に偽	30
1.6.5	柔軟な条件判断——ifthen	31
1.7	マクロの調べ方	32
1.7.1	定義を調べる	33
1.7.2	中身を調べる	33
1.7.3	頑丈なマクロを調べる	34
1.8	そのほか気を付けること	35
1.8.1	何もしないけど役に立つ命令	35
1.8.2	ホワイトスペースの扱い	35
1.9	なんたら	36
第 2 章 L^AT_EX 解体		37
2.0.1	L ^A T _E X 流の定義	37
2.0.2	頑丈なコマンドの定義	39
2.1	array/tabular 環境実装への道 其の一	41
2.2	環境型コマンドの提供	42
2.3	外部ファイルの書き出し/読み出しをする	43
2.4	長さから ‘pt’ を取り除く	43
第 3 章 クラス解体		45
3.1	クラスファイルの設計	45
3.1.1	最小構成	45
3.1.2	版面を構成する最低限のパラメータ	46
3.1.3	普通の文字の大きさの定義	46
3.2	jsarticle	46
第 4 章 マクロ解体		49
4.1	L ^A T _E X 標準パッケージ概説	49
4.1.1	indentfirst	49
4.1.2	alltt	49
4.1.3	afterpage	50
4.1.4	xspace	50
4.1.5	calc	51

第 5 章 応用と実例 53

5.0.6	表紙の作成	53
5.1	問題集の作成	53
5.1.1	画像ファイルの読み込みの工夫	55
5.2	BibTeX で出力した文献一覧 .bbl の著者名の重複処理	55
5.3	著者名の終わりをピリオドでなくコンマにしたい	56
5.4	参考文献一覧における著者名の重複	56
5.5	電話番号などのハイフンの位置	57
5.6	pTeX と jTeX の判別	58
5.7	文字列の積み重ね	58
5.8	MoeTeX logo	60
5.9	ThorTeXTypo Wiki のロゴ	60
5.10	数値の基数変換	60
5.10.1	10 進数から 2 進数へ	60
5.10.2	2 進数から 10 進数へ	60
5.10.3	8/16 進数から 10 進数	61
5.11	“a. a and b. a, b and c.” の自動出力	61
5.12	PDF アノテーション機能のメモアイコン位置	62
5.13	hyperref において hyperlink が nest になる例	64
5.14	marginpar に table を	64

第 6 章 マクロ作成の基礎知識 67

6.1	リスト処理	67
6.2	リスト処理 (簡略化バージョン)	67
6.2.1	文字処理	70
6.2.2	文字列処理	70
6.3	展開と置換	70
6.3.1	展開順序	70
6.3.2	展開の抑制	70
6.3.3	閑話休題	70
6.4	グルーピング	70
6.4.1	局所	70
6.4.2	大域	70
6.4.3	グループの区切り	70
6.4.4	入れ子	70
6.4.5	波括弧の重み	70
6.5	モード	70

6.5.1	水平	71
6.5.2	垂直	71
6.5.3	内部/限定	71
6.5.4	ボックスの内容	71
6.5.5	モードのあれこれ	71
6.5.6	\framebox 実装への道 その 1	71
6.6	カウンタ	74
6.6.1	T _E X のカウンタ	74
6.6.2	L ^A T _E X のカウンタ	74
6.6.3	親子カウンタ	74
6.6.4	相互参照	74
6.7	ボックス・グルー	75
6.7.1	グルー	75
6.7.2	罫線	75
6.7.3	リーダー	75
6.7.4	ボックスの仕組み	75
6.8	ファイル	76
6.8.1	補助ファイルの役割	76
6.8.2	ファイル入力	76
6.8.3	ファイル出力	76
6.8.4	ファイルに関わるその他のコマンド	76
6.9	エラー	77
6.9.1	T _E X のエラー	77
6.9.2	L ^A T _E X のエラー	77
6.9.3	T _E X の警告	77
6.9.4	L ^A T _E X の警告	77
6.9.5	エラーの出し方	77
6.9.6	脆弱・頑丈	78
6.10	フォント	79
6.10.1	NFSS	79
6.10.2	基本属性	79
6.10.3	数式モードでのフォント	79
6.10.4	フォントの定義	79
6.11	クラスファイルの構築	79
6.11.1	もっとも短いクラス	79
6.11.2	ちょっと長いクラス	79
6.11.3	さらにちょっと長いクラス	79
6.11.4	版面	79
6.11.5	柱・ノンブル	79
6.11.6	表題	79

6.11.7	見出し	80
6.11.8	図表	81
6.11.9	目次	81
6.11.10	文献一覧・索引・用語集	83
6.12	空白	83
6.12.1	自動で入る空白	83
6.12.2	手動で入れる空白	83
6.12.3	スペースファクタ	83
6.13	出力ルーチン・ページの構成法	83
6.13.1	出力	83
6.13.2	マークの基本	83
6.13.3	output	83
6.14	ページ構成	83
6.14.1	パラメータ	83
6.14.2	段落処理	85
6.14.3	スペースファクタ	86
6.14.4	ハイフネーション	86
6.14.5	ハイフネーション	86
6.14.6	行分割	87
6.14.7	ページ分割	87
6.14.8	ブレイクポイント	87
第 7 章 L^AT_εX 2_ε 解体		89
7.1	予備知識	89
7.2	リスト系環境	92
7.2.1	リスト型環境	92
7.2.2	自作の箇条書き型の環境	92
7.2.3	list 環境	92
7.3	箇条書き環境の自作	92
7.3.1	trivlist 環境	95
7.3.2	enumerate など	95
7.4	数式	95
7.4.1	揃える	95
7.4.2	揃えない	95
7.4.3	複数行	95
7.4.4	数式番号	95
7.4.5	各種環境	95
7.5	表組み	96
7.5.1	tabbing 環境	96

7.5.2	array 環境	96
7.5.3	tabular 環境	96
7.5.4	既存環境の拡張	96
7.6	浮動体	96
7.6.1	制御用パラメータ	96
7.6.2	図表見出し	96
7.6.3	図表目次	96
7.6.4	浮動体制御	96
7.7	参考文献	96
7.7.1	引用形式	96
7.7.2	一覧形式	96
7.8	ページスタイル	96
7.8.1	既存のスタイル	96
7.8.2	ページスタイルのしくみ	97
7.8.3	自作のスタイル	101

第 8 章 L^AT_EX メモ帳 105

8.1	汎用的なくり返し式	105
8.2	jsclasses で図表番号の後にコロンを表示したいとか	106
8.3	T _E X 解剖	108

付録 A GNU Free Documentation License 111

1.	APPLICABILITY AND DEFINITIONS	111
2.	VERBATIM COPYING	113
3.	COPYING IN QUANTITY	113
4.	MODIFICATIONS	113
5.	COMBINING DOCUMENTS	115
6.	COLLECTIONS OF DOCUMENTS	115
7.	AGGREGATION WITH INDEPENDENT WORKS	115
8.	TRANSLATION	116
9.	TERMINATION	116
10.	FUTURE REVISIONS OF THIS LICENSE	116
	ADDENDUM: How to use this License for your documents	116

参考文献 119

命令索引	121
------	-----

索引	125
----	-----



第 1 章

マクロ作成の基礎知識

ここでは L^AT_EX でのマクロ・クラスの解説の前に理解すべきことを紹介します。L^AT_EX は T_EX を基盤としたマクロの集まりですから、まずは T_EX の基礎を習得すべきでしょう。この章では T_EX の概念的な知識を学んでみようとします。ランニングと筋力トレーニングのような基礎訓練になりますが、最後まで読んでみてください。

1.1 ゲームのルール

▼ 1.1.1 L^AT_EX の基盤

L^AT_EX はどのように構築されているのでしょうか。それらは全て `source2e.tex` に書かれています。これを理解するには何かが足りません。その足りない知識とは T_EX についてです。L^AT_EX は T_EX を基盤としていますから T_EX に関する知識、いわゆるプリミティブと呼ばれる命令について理解する必要があります。プリミティブとは T_EX のプログラム自体に直接組み込まれているコマンドのことで、数百個あります。これら全てを理解する必要はありませんが、L^AT_EX のマクロを理解する上で必要なものが沢山あります。L^AT_EX は T_EX を基盤としてそのシステムが構築されています (図 1.1)。

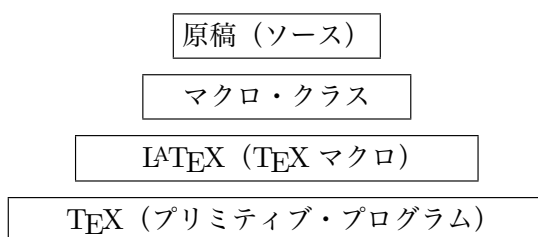


図 1.1 T_EX と L^AT_EX の関係

▼ 1.1.2 T_EX の文法

T_EX は組版を行なうプログラムです。それに関わる基本的な機能をボックス・グループモデルと呼ばれるような形態で提供しています。文章構成要素・数式要素・図表など、ありとあらゆるものを「高さ (height)」「幅 (width)」「深さ (depth)」「基準点 (reference point)」の四つを持ったボックスとして扱い、そのボックスを特定のページの適切な場所

に配置するためにグルーと呼ばれるバネのようなもので張り付けて行きます*1。

T_EX はボックス・グルーモデルに則ってページを組版しますが、T_EX にどのようにページを組めば良いかを指定するためにコマンドと呼ばれる文字の綴を文章と併せて記述することになります。

T_EX が提供するコマンドは分かる人には分かるのですが、まるでアセンブリ言語でも扱っているかのような使いにくさがあります。複合条件判断のやりにくさ、演算子の記述制限、アロケーションに関する制約、カテゴリーコード等に見られるような内部コードの導入など、プログラミングをかじったことのある人でもちょっと戸惑うような概念が沢山あります。

このような使いにくさ*2にも理由があります。言語仕様をアセンブリに近づける事により、T_EX は並列性を高めることができ、他のプログラムに比べて非常に高速に実行できるようになるからです。Donald E. Knuth 氏は T_EX の高速性を優先したようです*3。

そのため、プログラミングもやったことがなく、数学もちょっと苦手な方には T_EX は扱い辛い言語になっているかもしれません。

▼ 1.1.3 L^AT_εX の文法

T_EX は組版をするための最高の機能を提供してくれますが、インタフェースは最高ではないようです。文書を執筆する上で人間が理解しやすい方が良いのは当然のことなので、Leslie Lamport 氏が T_EX に SGML の概念を導入し L^AT_εX を開発しました。この L^AT_εX の特徴的な事としては宣言型のコマンドで文章の構成要素をマークアップ（意味付け）していくという方法になっていることでしょう。例えばある画像 `<filename>.eps` を中央揃えで文書に配置したければ次のように `document` 環境の中に `center` を記述し、その `center` 環境の中に `\includegraphics` 命令を入れ子にします。

```
1 \begin{document}
  \begin{center}
    \includegraphics[width=10cm,height=5cm]{filename.eps}
  \end{center}
\end{document}
```

現在は Leslie Lamport 氏がほぼ一人で開発した L^AT_εX（この当時の L^AT_εX を L^AT_εX 2.09 と呼びます）から、現在は L^AT_εX 3 プロジェクトチームが開発をしている L^AT_εX 2_ε が広く使われています。将来的には L^AT_εX 3 というバージョンが世に登場することになり、この冊子のタイトル『好き好き L^AT_εX 2_ε』というのも時代遅れになるかも知れませんが、基礎的な事は変わらないと思いますので、この冊子では L^AT_εX 2_ε を前提として解説をしていく事にします。

*1 もちろん、T_EX が最終的にある文字 X をページに配置するにはその文字 X が持っているフォントメトリクスなど、必要となる各種情報が参照されます。

*2 使いにくいかどうかを判断するのは個々のユーザなので、ここでは一般的な高級言語に見られるような機能が実装されているかどうかという事で判定してもらいたい。それでも愛味だと思えますけど。

*3 T_EX のソースを見れば分かりますが、並列性に関する種々の工夫が随所に見受けられます。さすが Knuth です。

1.2 定義と代入

L^AT_EX 的な定義には `\newcommand` や `\newenvironment` などがありました。しかし、これらも実はマクロとして定義されたものに過ぎません。本来マクロの定義には T_EX のプリミティブ `\def` (他にも `\edef`, `\gdef`, `\xdef` も含みます) を使うことになります。

▼ 1.2.1 定義

あるマクロを定義するには `\def` を使います。これはすでに定義済みのマクロでも定義済みかどうかの判定をせずに、再定義してしまうものです。状況に応じて `\def` ではなく `\edef` を使います。

```
\global<\long|\outer><\def|\edef><制御綴><引数と書式><定義>
```

とりあえず、`\global` は大域的にマクロを定義したいときに付けるもので、なくても構いません。`\long` はそのマクロが改段落 (`\par`) を含むような長い場合に使います。`\outer` はそのマクロが他のマクロの引数の中にあってはならないことを指定します。`\long` と `\outer` は共になくても構いません。`<定義>` には引数や他のマクロを含むテキストを記述することが出来ます。

▷ 例題 1.1 引数を表すには井桁と一桁の数字 ‘`#<n>`’ を使い、1-9 が使用できます。次のファイルをタイプセットしてそれを確認してください。

```
\def\hoge{hoge} \hoge\par
\def\hoge#1{[#1]} \hoge{びっくり}\par
\def\hoge#1#2{(#1,#2)} \hoge{x}{y}\par
\def\hoge#1#2#3#4#5#6#7#8#9{#1; #2; #3; #4; #5; #6; #7; #8; #9; \gege}
5 \def\gege#1#2{#1, #2.} \hoge{A}{B}{C}{D}{E}{F}{G}{H}{I}{J}{K}\par
\def\hoge#1{‘#1’} \hoge abcde\par
```

結果は “hoge, [びっくり], (x,y), A; B; C; D; E; F; G; H; I; J, K. ‘a’bcde” などとなります。四つ目の `\hoge` では引数を 11 個受け取るために別の命令 `\gege` を呼び出すようにしています。

▷ 例題 1.2 `\def` に慣れるために次のような記述をタイプセットしてください。また実行結果を吟味してください。

```
\def\nyo#1{\item[で○こ]「#1によ」}
\def\nyu#1{\item[ぶ○こ]「#1にゆ」}
\def\piyo#1{\item[び○こ]「#1びよ」}
4 \def\gema#1{\item[げ ○]「#1げま」}
\begin{description}
\nyo{ああ、ひまだ}
\nyu{そうか}
\nyo{なんか、面白いことはおこらないか}
9 \nyu{昨日は暴れん坊が来た}
\nyo{そうだった}
\gema{もう、こりごり}
\nyu{だれかきた}
```

```

\gema{だれ}
14 \piyo{びよー、びよー}
\nyo{いきなり、だれだ}
\nyu{だれ}
\gema{だれ}
\piyo{びよびよびよ。悪、参上です}
19 \end{description}

```

▷ 例題 1.3 `\edef` は〈定義における引数とマクロを展開してから〉〈制御綴〉を定義します。次の記述をタイプセットして、出力結果を吟味してください。

```

1 \def\A{AAA} \def\B{BBB}
\edef\hoge{\A~and \B}
\def\geho{\A~and \B}
\hoge, \geho\par
\def\A{CCC} \def\B{DDD}
6 \hoge, \geho

```

`\A` と `\B` が再定義された後の `\hoge` と `\geho` では `\hoge` は前の内容が置換されています。

▷ 例題 1.4 `\def` では自分自身を参照するようなマクロは定義することが出来ません。

```

\def\A{ほげ}
\def\A{ほげ\A}
\A\par

```

上記ソースを実行すると

```
! TeX capacity exceeded, sorry [main memory size=1000001].
```

なるエラーとなります。こうならないためには `\edef` を使います。

```

\def\A{ほげ}
2 \edef\A{ほげ\A}
\A\par

```

▷ 例題 1.5 改段落 (`\par` などにより複数の段落) を含むようなマクロを定義するには `\def/\edef` に `\long` を補います。

```

\long\def\A{ほげ\par ほげ}
\long\def\B{\par\vskip 1em\hrule\vskip 1ex}
\B \A \B \A

```

```
ほげ
ほげ
```

```
ほげ
ほげ
```

▼ 1.2.2 書式付き引数

▷ 例題 1.6 引数を受け取るときに「書式」を指定して受け取る事が出来ます。これはパターンマッチなどと呼ばれる事もあります。次の記述をタイプセットして実行結果を吟味してください。

```

\def\hoge#1/#2/#3#4{#1年#2月#3#4日}
2 \hoge 2005/03/09 \par
\def\hoge#1{\gege#1owari}
\def\gege#1/#2/#3owari{#1年#2月#3日}
\hoge {2005/3/9}

```

日付を表すのに月日の桁が一桁だった場合は最後の引数がありませんので、別の方法を用いています。

馴染みやすい例として日付 (date) があります。Vine Linux 3.x の日本語化された date コマンドでは `date` とすれば次のように表示されます。

```
2005年 2月 15日 火曜日 16:52:03 JST
```

この出力を逆に英語のフォーマットに戻すマクロを考えます。

```
<yyyy> 年 <mm> 月 <dd> 日 <D> 曜日 <hh>:<mm>:<ss> JST
```

これは常に決まった書式になっていますので `hoge.tex` に日付を書き出します。

```
$ echo "\hiduke 'date'" >hoge.tex
```

このようにして `file.tex` において次のようにすれば、実行結果は `'16:52:03 2005/ 2/ 15'` となります。

```

\documentclass{jarticle}
\begin{document}
\def\hiduke#1年#2月#3日#4曜日#5:#6:#7JST{#5:#6:#7\space#1/#2/#3}
4 \input{hoge}
\end{document}

```

▼ 1.2.3 定義の中の定義

例えば `\hoge` を定義するような記述があるとします。

```
\def\aaa#1#2{#1 は #2 です}
```

このマクロの中で新たに `\bbb` という引数を取る命令を作成したいとします。

```
\def\bbb#1{#1 とか}
```

この場合、

```
\def\aaa#1#2{#1 は #2 です、\def\bbb#1{#1 とか}}
```

としたのでは `\bbb` の引数が `#1` となり、定義中の `#1` は `\aaa` の一つ目の引数として解釈されます。これを防止するためには 定義の深さの数だけ `#` を増やします。

```
\def\aaa#1#2{#1 は #2 です、\def\bbb##1{##1 とか}}
```

例題として日付を定義するようなものを示します。

```

\def\Year#1{%
  \def\Month##1{%
    \def\Date####1{#1/##1/####1}%
4   }%
}
\Year{2004}
\Month{11}
\Date{25}

```

井桁 # をエスケープさせるためには ## とするということになります (Make の \$ みたいなものつす)。

▼ 1.2.4 グルーピング

▷ 例題 1.7 あるグループの中で何らかのマクロを `\def`/`\edef` で定義すると、そのグループの中だけでマクロが使えます。ですから、次の三つの `\hoge` はすべて

! Undefined control sequence. ということで、未定義になります。

```

{\def\hoge{ほ} \edef\hoge{\hoge げ}}
2 \hoge, {\hoge}, {\hoge}}\par

```

これをグループの外でも有効にするには `\global` を補います。

```

{\global\def\hoge{ほ} \global\edef\hoge{\hoge げ}}
\hoge, {\hoge}, {\hoge}}\par

```

しかし、わざわざ `\global` を毎回補うのは面倒なので `\def` においては `\gdef`、`\edef` においては `\xdef` という短縮形が用意されています。ですから、上記のソースは次のように書いても同じ事になります。

```

{\gdef\hoge{ほ} \xdef\hoge{\hoge げ}}
\hoge, {\hoge}, {\hoge}}\par

```

自分のグループの上層に影響力を与えるには `\global` などのような大域化が必要になりますが、自分よりも下層の場合は必要ありませんので、次のような記述でなんら問題ありません。

```

\gdef\hoge{ほ} \xdef\hoge{\hoge げ}
\hoge, {\hoge}, {\hoge}}\par

```

複数の引数を受け取るマクロを定義するにもいくつか方法があります。L^AT_EX 的には `\newcommand` で次のように定義できます。

```

\usepackage{color}
\newcommand*\hoge[2]{%
  \color{red}#1 is red.
  \color{blue}#2 is blue.}
\hoge{赤}{青}

```

赤 is red. 青 is blue.

T_EX 的には `\def` で次のようにやるのが分かりやすいでしょうか。

```
\def\hoge#1#2{%
  {\color{red}#1 is red.}
  {\color{blue}#2 is blue.}}
\hoge{赤}{青}
```

赤 is red. 青 is blue.

次の例では `\hoge` に直接引数を与えるのではなく `\@hoge` と `\@@hoge` の二つにそれぞれ引数の一つずつ渡すようにしています。

```
\makeatletter
\def\hoge{\bgroup\color{red} \@hoge}
\def\@hoge#1{#1 is red. \egroup
  \bgroup\color{blue}\@@hoge}
\def\@@hoge#1{#1 is blue.\egroup}
\hoge{赤}{青}
\makeatother
```

赤 is red. 青 is blue.

`\bgroup/\egroup` はマクロで、とりあえずはグループの開始と終わりを意味するものとして認識してください。

マクロ `\hoge` を次のように定義したとします。

```
\def\hoge{{\color{red} \@hoge}
\def\@hoge#1{#1 is red. }
3 {\color{blue}\@@hoge}
\def\@@hoge#1{#1 is blue.}}
```

このようにすると `\hoge`, `\@hoge`, `\@@hoge` の定義のテキストの中でグループの対応が取れずにエラーになります。要するに書き換えるとうなるでしょう。

```
1 \def\hoge{%
  {\color{red} \@hoge}%
  \def\@hoge#1{#1 is red. }%
  {\color{blue}\@@hoge}%
  \def\@@hoge#1{#1 is blue.}%
6 }
```

なんと恐ろしいことでしょうか。このようにならないためにはさき程の `\bgroup`, `\egroup` を使うようにします。これら以外にも `\begingroup`, `\endgroup` を用いることができますが、とりあえず前者の方だけ覚えておいてください。

```
\def\hoge{\begingroup\color{red} \@hoge}
\def\@hoge#1{#1 is red. \endgroup
  \begingroup\color{blue}\@@hoge}
4 \def\@@hoge#1{#1 is blue.\endgroup}
\hoge{赤}{青}
```

`\group/\egroup`, `\begingroup/\endgroup`, `{/}` によるグループの中ではスコープが効いている時、グループの外側にも定義を有効にさせるには `\gdef/\xdef` を使うこととなります。

```
\bgroup \def\hoge{hoge1} \egroup \hoge
\begingroup \def\hoge{hoge2} \endgroup \hoge
{\def\hoge{hoge3} \endgroup \hoge}
% 上記三つはすべて未定義としてエラーになる。下記三つは有効。
```

```
5 \bgroup \gdef\hoge{hoge1} \egroup \hoge
   \begingroup \gdef\hoge{hoge2} \endgroup \hoge
   {g\def\hoge{hoge3} \endgroup \hoge
```

▼ 1.2.5 代入

例えば次のように `\A` が定義されていたとします。

```
\def\A{あがが}
```

このマクロが将来的に変更されることが分かっており、定義の内容を保存しておきたいときは `\def` ではなく `\let` というプリミティブを使います。

```
\global\let<制御綴> = <トークン>
```

大域化を行なう `\global` とイコール `'=` は省略可能です。

▷ 例題 1.8 なんだか、代入と言われても良く分からないので、次の記述をタイプセットして、その実行結果を吟味してください。

```
\def\A{あがが}
\let\origA=\A
\A, \origA\par
4 \def\A{げげ}
\A, \origA\par
```

結果は「あがが, あがが」「げげ, あがが」となることから、`\A` を定義し直しても `\origA` には「あがが」が保存されていることとなります。

TEX では定義 `\def` と代入 `\let` は別々の働きをします。代入はその場で中身が決まり、あとから中身が変わることはありませんが、定義の場合はそれを再定義することにより中身が変更されます。

```
\def\hoge{aaa}
\let\foo=\hoge
\def\bar{bbb}
\foo, \hoge
```

という場合の出力は `aaa, bbb` となります。

```
1 \def\bar{ccc}
   \def\hoge{aaa\bar}
   \let\foo=\hoge
   \def\hoge{bbb}
   \foo, \hoge% \foo には aaa\bar という定義が代入されている。
```

ならば `aaacc, bbb` になり古い `\hoge` の中身が保存されていることとなります。

例えば `figure` 環境の前後に必ず罫線を引くことにする場合は `\figure` と `\endfigure` の定義をいったん保存しておき、次のように書き換えてしまうことも出来ます。

```
\let\origFigure=\figure
\let\origEndFigure=\endfigure
\def\figure{\par\vskip1ex\hrule\origFigure}
```

```

\def\endfigure{\origEndFigure\hrule\par\vskip1ex}
5 ほげほげ。
\begin{figure}[htbp]
\centering\fbbox{ここに画像が入るんだらう}
\caption{サンプル}
\end{figure}
10 げほげほ。

```

1.3 エンジン始動!

『好き好き L^AT_EX 2_ε 初級編』の第 6 章にて、記号、コマンド、カテゴリーコード、グループリング、入れ子、スコープ、宣言と命令などの基本は解説してあります。ここではそれらがどのように実装されているのかなどを考えます。

▼ 1.3.1 文字コード

まずはコンピュータの中で文字や記号がどのように表現されているのかを知らなくてはなりません。コンピュータの中で文字や記号は単なる数値として表現される事になります。「ほげ」という文字列があれば 16 進数で ‘0xA4 0xDB 0xA4 0xB2’ と表記することも出来ます。このように、コンピュータで文字を表現するために使われる数値を文字コードと呼び、もっとも標準的なコード ASCII などがあります。ASCII の文字コード一覧を表 1.1 に示します。

表 1.1 ASCII の文字コード一覧

<i>x</i>	'0	'1	'2	'3	'4	'5	'6	'7	
'00x	⟨NUL⟩	⟨SOH⟩	⟨STX⟩	⟨ETX⟩	⟨EOT⟩	⟨ENQ⟩	⟨ACK⟩	⟨BEL⟩	"0x
'01x	⟨BS⟩	⟨HT⟩	⟨LF⟩	⟨VT⟩	⟨FF⟩	⟨CR⟩	⟨SO⟩	⟨SI⟩	"0y
'02x	⟨DLE⟩	⟨DC1⟩	⟨DC2⟩	⟨DC3⟩	⟨DC4⟩	⟨NAK⟩	⟨SYN⟩	⟨ETB⟩	"1x
'03x	⟨CAN⟩	⟨EM⟩	⟨SUB⟩	⟨ESC⟩	⟨FS⟩	⟨GS⟩	⟨RS⟩	⟨US⟩	"1y
'04x	□	!	"	#	\$	%	&	'	"2x
'05x	()	*	+	,	-	.	/	"2y
'06x	0	1	2	3	4	5	6	7	"3x
'07x	8	9	:	;	<	=	>	?	"3y
'10x	@	A	B	C	D	E	F	G	"4x
'11x	H	I	J	K	L	M	N	O	"4y
'12x	P	Q	R	S	T	U	V	W	"5x
'13x	X	Y	Z	[\]	^	_	"5y
'14x	'	a	b	c	d	e	f	g	"6x
'15x	h	i	j	k	l	m	n	o	"6y
'16x	p	q	r	s	t	u	v	w	"7x
'17x	x	y	z	{		}	~	⟨DEL⟩	"7y
	"8	"9	"A	"B	"C	"D	"E	"F	y

表 1.1 では一文字 7 ビットまでで、8 進数で言えば '000 から '177 までの記号が定義されています。T_EX では '000 から '377 までの記号を割り当てられる 8 ビットのエンコー

ディングに対応した文字コードを許容することが出来ます。L^AT_EX2_ε の標準では OT1 と呼ばれる 7 ビットエンコーディングが選択されることになります。

表 1.1 における文字 ‘A’ は 8 進数で ‘101’、16 進数では “41” と表すことが出来ます。

T_EX の原稿で表 1.1 のそれぞれの文字を指定するには表 1.2 の通りに記述すれば良いことになります。

表 1.2 文字コードの指定方法

<i>x</i>	'0	'1	'2	'3	'4	'5	'6	'7	
'00 <i>x</i>	^^@	^^A	^^B	^^C	^^D	^^E	^^F	^^G	"0x
'01 <i>x</i>	^^H	^^I	^^J	^^K	^^L	^^M	^^N	^^O	"0y
'02 <i>x</i>	^^P	^^Q	^^R	^^S	^^T	^^U	^^V	^^W	"1x
'03 <i>x</i>	^^X	^^Y	^^Z	^^[^^\	^^]	^^^	^^_	"1y
'04 <i>x</i>	□	!	"	#	\$	%	&	'	"2x
'05 <i>x</i>	()	*	+	,	-	.	/	"2y
'06 <i>x</i>	0	1	2	3	4	5	6	7	"3x
'07 <i>x</i>	8	9	:	;	<	=	>	?	"3y
'10 <i>x</i>	@	A	B	C	D	E	F	G	"4x
'11 <i>x</i>	H	I	J	K	L	M	N	O	"4y
'12 <i>x</i>	P	Q	R	S	T	U	V	W	"5x
'13 <i>x</i>	X	Y	Z	[\]	^	_	"5y
'14 <i>x</i>	'	a	b	c	d	e	f	g	"6x
'15 <i>x</i>	h	i	j	k	l	m	n	o	"6y
'16 <i>x</i>	p	q	r	s	t	u	v	w	"7x
'17 <i>x</i>	x	y	z	{		}	~	^^?	"7y
	"8	"9	"A	"B	"C	"D	"E	"F	y

実は文字を出力するには `\char` という命令が使えて、文字コードを指定すると該当する記号をそのエンコーディングに応じて出力してくれます。

```
\char<数値>
\symbol{<数値>}
```

〈数値〉は 8 進数表記でも 16 進数表記でも 10 進数でも構いません。

▷ 例題 1.9 次の記述をタイプセットすると結果はどうなるでしょうか。

```
A, \char65, \char"41, \char'101, \symbol{65}, \symbol{"41}, \symbol{'101}.
```

結果はやはり ‘A, A, A, A, A, A, A.’ となるでしょう。

▷ 例題 1.10 次のファイルをタイプセットすると結果はどうなるでしょうか。

```
\documentclass{jarticle}
\begin{document}
\edef\temp{o}
4 \let\end\enddocument
\catcode"6F=\active \edef^^/{d\temp uy\temp}%"
hoge
hoge
```



```
hoge
9 \end
```

文字 ‘o’ のカテゴリコードを `\active` にして、‘douyo’ として定義しています。しかし `\edef` の段階で `o` が自分自身の定義を参照するので無限ループに陥りますので、予め `\temp` を定義しておきます。

実際は `` とか `<CR>` などの「記号」は存在しないので `'000-'037`, `'177` (8ビットエンコーディングの場合は `'178-'377` も含みます) には別の記号が割り当てられています。この割り当ての規則はそれぞれのエンコーディングで異なっていますが、T_EX ではこの割り当ての標準的なものが定められており、もっとも基本的なものとして *OT1* というエンコーディングがあります。他にも 8ビットエンコーディングの場合は *T1*、数式の場合はまた別のエンコーディングとなります。例えば Donald E. Knuth 氏がデザインした *OT1* エンコーディングの Computer Modern フォントのローマン体 `cmr10` は表 1.3 に示す記号が割り当てられている事になります。

表 1.3 `cmr10` 中の記号一覧

x	'0	'1	'2	'3	'4	'5	'6	'7	
'00x	Γ	Δ	Θ	Λ	Ξ	Π	Σ	Υ	"0x
'01x	Φ	Ψ	Ω	ff	fi	fl	ffi	ffl	"0y
'02x	ı	ı	˘	˙	˚	˛	˜	˝	"1x
'03x	ı	ß	æ	œ	ø	Æ	Œ	Ø	"1y
'04x	-	!	"	#	\$	%	&	'	"2x
'05x	()	*	+	,	-	.	/	"2y
'06x	0	1	2	3	4	5	6	7	"3x
'07x	8	9	:	;	i	=	ı	?	"3y
'10x	@	A	B	C	D	E	F	G	"4x
'11x	H	I	J	K	L	M	N	O	"4y
'12x	P	Q	R	S	T	U	V	W	"5x
'13x	X	Y	Z	["]	^	˘	"5y
'14x	‘	a	b	c	d	e	f	g	"6x
'15x	h	i	j	k	l	m	n	o	"6y
'16x	p	q	r	s	t	u	v	w	"7x
'17x	x	y	z	-	—	"	~	¨	"7y
	"8	"9	"A	"B	"C	"D	"E	"F	y

▷ 例題 1.11 次の記述をタイプセットするとどうなるでしょうか。

```
1 Office~or O\char"0E ce, \AE~or \char'035.\par
  $\Gamma$\~or \char'000, ---~or \symbol{"7C}
```

予想通り ‘Office or Office, Æ or Æ. Γ or Γ, — or —’ となります。

直接文字コードを指定するとキーボードからアクセスできないようなコマンドにアクセスできるようになります。しかし、いつも直接コードを指定していると痛い目に会います。次のファイルをタイプセットすると結果はどうなるでしょう。

```

\documentclass{article}
\usepackage[T1]{fontenc}
3 \usepackage{txfonts}
\begin{document}
Office~or O\char"0E ce, \AE~or \char'035.\par
$\Gamma$\~or \char'000, ---~or \symbol{"7C}
\end{document}
    
```

どうやら T1 というエンコーディングと OT1 では記号の配置が異なるので結果は ‘Office or O ce, Æ or fl. Γ or ` , — or |’ となってしまいます。この理由を知るには T1 エンコーディングである txr のコード一覧を見てみる必要があります。

表 1.4 txr 中の記号一覧

<i>x</i>	'0	'1	'2	'3	'4	'5	'6	'7	
'00x	`	´	^	~	¨	ˆ	˚	ˇ	"0x
'01x	˘	-	·	‚	˙	˚	<	>	"0y
'02x	“	”	„	«	»	-	—		"1x
'03x	o	ı	j	ff	fi	fl	ffi	ffl	"1y
'04x	˘	!	"	#	\$	%	&	'	"2x
'05x	()	*	+	,	-	.	/	"2y
'06x	0	1	2	3	4	5	6	7	"3x
'07x	8	9	:	;	<	=	>	?	"3y
'10x	@	A	B	C	D	E	F	G	"4x
'11x	H	I	J	K	L	M	N	O	"4y
'12x	P	Q	R	S	T	U	V	W	"5x
'13x	X	Y	Z	[\]	^	_	"5y
'14x	‘	a	b	c	d	e	f	g	"6x
'15x	h	i	j	k	l	m	n	o	"6y
'16x	p	q	r	s	t	u	v	w	"7x
'17x	x	y	z	{		}	~	-	"7y
'20x	Ǻ	Ą	Ć	Č	Ď	Ě	Ę	Ǧ	"8x
'21x	Ł	Ł	Ł	Ń	Ń	Ń	Ń	Ń	"8y
'22x	Ř	Ś	Š	Ş	Ť	Ť	Ů	Ů	"9x
'23x	Ÿ	Ž	Ž	Ž	ı	ı	đ	§	"9y
'24x	ă	ą	ć	č	ď	ě	ę	ǧ	"Ax
'25x	ł	ł	ł	ń	ń	ń	ń	ń	"Ay
'26x	ř	ś	š	ş	ť	ť	ů	ů	"Bx
'27x	ÿ	ž	ž	ž	ı	ı	ı	ı	"By
'30x	À	Á	Â	Ã	Ä	Å	Æ	Ç	"Cx
'31x	È	É	Ê	Ë	Ì	Í	Î	Ï	"Cy
'32x	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	Œ	"Dx
'33x	Ø	Ù	Ú	Û	Ü	Ý	Þ	Ï	"Dy
'34x	à	á	â	ã	ä	å	æ	ç	"Ex
'35x	è	é	ê	ë	ì	í	î	ï	"Ey
'36x	ð	ñ	ò	ó	ô	õ	ö	œ	"Fx
'37x	ø	ù	ú	û	ü	ý	þ	ÿ	"Fy
	"8	"9	"A	"B	"C	"D	"E	"F	y

▼ 1.3.2 カテゴリーコード

初級編でカテゴリーコードに関する基本的な事は紹介済みです。l^AT_EXplain.dtx において実際に次のように定義されています。\\catcode コマンドと特定の〈文字〉を意味する表現 (例えば ‘\\{’) とイコール ‘=’ の後に設定したいカテゴリーコードを指定するだけです。

```
\\catcode <文字> = <カテゴリーコード>
```

〈文字〉には左シングルクォートを伴う文字表現と、数値表現の両方ができます。イコール ‘=’ は省略可能です。

```
\\catcode'\\=0 % バックスラッシュ -> エスケープ文字
\\catcode'\\{=1 % 左波括弧 -> グループの開始
3 \\catcode'\\}=2 % 右波括弧 -> グループの終わり
\\catcode'\\$=3 % ドル -> 数式モードへの移行
\\catcode'\\&=4 % アンパサンド -> 整列区切り
\\catcode'\\^M=5 % 改行 (CR) -> 行の終わり
\\catcode'\\#=6 % 井桁 -> マクロの変数
8 \\catcode'\\^=7 % ハット -> 上付き添字
\\catcode'\\_ =8 % アンダーバー 下付き添字
\\catcode'\\^^@=9 % NULL 文字 -> 無効
\\catcode'\\^^I=10 % タブ文字 -> スペース
\\catcode'\\ =10 % スペース -> ホワイトスペース
13 \\catcode'\\A=11 ... \\catcode'\\Z=11 % 大文字 -> 普通の文字
\\catcode'\\a=11 ... \\catcode'\\z=11 % 小文字 -> 普通の文字
\\catcode'\\%=14 % パーセント -> コメント
\\catcode'\\^^?=15 % デリート文字 -> 無効
\\catcode'\\~ =\\active % チルダ (\\chardef\\active=13)
18 \\catcode'\\^L=\\active \\outer\\def^^L{\\par}% form-feed -> \\outer\\par
```

\\catcode において ‘^^M’ のような表現が使われています。これは表 1.2 に示す通り、キーボードから直接アクセスできない文字を指定するための表現方法です。

これに加えてアットマーク ‘@’ は裏側で普通の文字として扱いたいので、マクロ・クラス中では以下の様に変更されています。

```
\\catcode'@=11 % アットマーク -> 普通の文字
```

アットマークをアルファベットとして扱うか、そのたの記号として扱うかは結構問題となるので、切替えを簡単にするため \\makeatletter と \\makeatother が用意されています。

```
\\def\\makeatletter{\\catcode'@=11\\relax}
\\def\\makeatother{\\catcode'@=12\\relax}
```

索引を生成するために \\index 命令などを使いますが、\\index の引数中では T_EX の特殊文字が邪魔をしてしまうので、次のように主要な記号をそのたの記号にカテゴリーコードを変換する \\@sanitize 命令もあります。

```
\\def\\@makeoother{\\catcode'#1=12\\relax}
\\@sanitize{\\@makeoother\\ \\@makeoother\\ \\@makeoother\\$\\@makeoother\\&%
3 \\@makeoother\\#\\@makeoother\\^\\@makeoother\\_\\@makeoother\\%\\@makeoother\\~}
```

カテゴリコードが 11 の波括弧（グループ区切り）を使うことがあるので、記憶の片隅にでも留めておいてください。

```
{\catcode'[=1 \catcode']=2
2 \catcode'={11 \catcode'}=11
\gdef\@charlb[{}
\gdef\@charrb[{}
]% } brace matching
```

`\gdef` などによって命令を定義するにはグループの開始を示すカテゴリコード '1' の文字と終わりを示す '2' の文字がなければなりません。そのため、局所的にそれらを角括弧で代用することで波括弧をグループの開始や終了を示す文字として使わないようにしています。`\catcode` で波括弧のカテゴリコードを変更しているので、最後のグループの終わりは角括弧でなければなりません。

▷ 例題 1.12 タイプライタ体の並括弧にアクセスしたいとき、次のどの方法を取るのが無難か考えてください。

```
\makeatletter
% {, } が文字として出てくる
\string{ \string}\par % NG
% 括弧の対応が取れることが前提となる
5 \bgroup\ttfamily \string{ \string} \egroup \par
% cmtt10 のように OT1 エンコーディングの場合は OK
\texttt{\char'173} \texttt{\char'175}\par
% これは LaTeX に予めあるコマンド
\texttt{\@charlb} \texttt{\@charrb}\par
10 % \{, \} は結局 数式フォントの並括弧を使っている。
% その証拠に cmr10 に並括弧は存在しない。
% LaTeX Font Warning: Font shape 'OMS/cmtt/m/n' undefined
% (Font) using 'OMS/cmsy/m/n' instead
% (Font) for symbol 'textbraceleft' on input line 9.
15 \texttt{\{} \texttt{\}}\par % NG
{\ttfamily \textbraceleft\space \textbraceright}\par % NG
{\usefont{OT1}{cmr}{m}{n} \texttt{\char'173} \texttt{\char'175}}\par
{\usefont{T1}{txr}{m}{n} \texttt{\char'173} \texttt{\char'175}}\par
% 未知のエンコーディングに遭遇したらどうなる?
20 \makeatother
```

OT1 や T1 エンコーディングのように既知のものしか用いられない場合は直接文字コードを指定しても良いのですが、まったく知らないものでも対応できるように、「文字としての並括弧」を使うことが出来る `\@charlb`, `\@charrb` がもっとも汎用性は高そうですね。

▼ 1.3.3 文中の特殊記号の扱い

文中で `<` `>` `^` `_` `'` `,` `"` などを記述すると、当然ながらエラーになります。しかし、以下の例は恐ろしいことをやっているため、一応出力できるようになっています。知らないコマンドが沢山ありますので、まだ読まなくても良いです。

```
\documentclass{jarticle}
\makeatletter
```

```

% "
\def\dq{"}%
5 \catcode 34=\active
\newif\ifDQ
\def"{\ifmode \rq\rq \else \ifDQ \rq\rq \DQfalse \else
  \DQtrue \lq\lq\fi\fi}% " % ‘^’ and ‘_’
\catcode‘^’=\active
10 \catcode‘\_’=\active
\def~{\ifmode \sp \else \~\relax \fi}
\def_{\ifmode \sb \else \_ \relax \fi}
% ‘<’ and ‘>’
\def\leftangle{<}
15 \def\rightangle{>}
\catcode‘<’=\active
\catcode‘>’=\active
\def<{\ifmode \leftangle \else $\langle$\fi}
\def>{\ifmode \rightangle \else $\rangle$\fi}
20 % ‘(\lq) and ’(\rq)
\newif\ifSQ
\catcode‘\’=\active
\catcode‘\’=\active
\def’{\ifmode \rq \else \ifSQ \rq \SQfalse \else \SQtrue \lq\fi\fi}
25 \def’{\ifmode \lq \else \SQtrue \lq \fi}
\makeatother
\begin{document}
\texttt{http://www.any.dom.jp/hoge_hoge/}\par
$hoge_hoge$\par
30 <hoge>\par
Hello, <hoge> is $x<3$.\par
’Hoge’ is my ‘hoge.’ ‘hoge’ is ‘hoge’.\par
d\‘am\’e, d\‘am\’e\’e\’e\’e\’e. H"o"o"o.\par
"Hoge" is my ‘‘hoge.’’ "hoge" is ‘‘hoge.”\par
35 \end{document}

```

1.4 データ型

TEX には「カウンタ (`\count`)」、「長さ (`\dimen`)」、「スキップ (`\skip`)」、「数式スキップ (`\muskip`)」、「ボックス (`\box`)」、「トークンリスト (`\toks`)」、「ファイル」などのデータ型変数が存在します。それぞれの変数は各々のコマンドでアロケーションします。「ファイル」に関してはデータ型として扱って良いかどうかは疑問が残りますが、簡単のためデータ型に分類しておきます。このような型を持つ変数がメモリのある領域に確保されたものを TEX ではレジスタと呼ぶことにします。

▼ 1.4.1 カウンタ

32 ビット長の整数を保存できる「カウンタ」なるデータ型があります。TEX 的には `\newcount` で確保します。符号付なので 2,147,483,647 から -2,147,483,647 ("7FFFFFFF から -"7FFFFFFF) の数値を 10 進表記、8 進表記、16 進表記、文字表記などで指定できます。

```
\newcount<レジスタ名>
<レジスタ名> = <数値>
```

<レジスタ名> には制御綴で適当な名前を付けます。確保済かどうかの確認はされません。値を代入するときのイコールは省略可能です。カウンタレジスタの中身を表示するには `\the` を使うものと `\romannumeral` を使う方法とがあります。

```
\the <レジスタ名> (<レジスタ> の中身を表示)
\romannumeral <レジスタ名> (<レジスタ> をローマ数字で表示)
\number <レジスタ名> (<レジスタ> の数値表現を表示)
```

あるレジスタの数値だけを持ってくるには `\number` というプリミティブも使用できます。`\the` や `\number` によって中身を取り出すと、その中身は文字列として出力されます。

▷ 例題 1.13 次の記述をタイプセットして、その出力結果を吟味してください。

```
\newcount\hoge
\hoge="FF
\the\hoge\par \romannumeral\hoge\par \number\hoge\par %"
```

▷ 例題 1.14 次の記述をタイプセットして、その出力結果を吟味してください。

```
\newcount\cnta
2 \def\hoge#1{\cnta=#1 \the\cnta\par}
\hoge{"00000000} \hoge{"00000001}
\hoge{"7FFFFFFE} \hoge{"7FFFFFFF}
\hoge{"-7FFFFFFF} \hoge{"-7FFFFFFE}
\hoge{2147483647} \hoge{-2147483647}
7 \hoge{'\a} \hoge{'177} \hoge{'.'}
```

上記の結果は ‘0 1 2147483646 2147483647 -2147483647 -2147483646 2147483647 - 2147483647 97 127 46’ となります。

L^AT_EX では既に次の三つのカウンタが確保されています。

```
\newcount\@tempcnta
\newcount\@tempcntb
3 \newcount\count@ % 255 番
```

この三つのカウンタはマクロパッケージやクラスファイルの中で一時的に使うレジスタとして自由に使用できます。ただし、複数のマクロパッケージでこの使い回しのレジスタを使用すると値が勝手に書き換えられてしまうので不都合が発生する場合があります。

先程の `\def\hoge#1{\cnta=#1 \the\cnta\par}` をもう少し工夫して次のように書いたとしましょう。

```
\long\def\set#1#2{#1=#2\the\@tempcnta\par}
2 [\@tempcnta=0 \the\@tempcnta]\par
\set{\@tempcnta}{5}
\set{\@tempcntb}{\the\@tempcnta}\par
[\the\@tempcnta], [\the\@tempcntb]\par
```

結果は ‘[0] [50], [5050]’ となります。おや、これはなんということでしょう。`\set` に本来やってほしいのは一つ目の引数に二つ目の数値を設定し、その中身を表示することにあるはずですが、この場合は表示したい数値 `\the\@tempcnta` が二つ目の引数に続いて代入されてしまいます。レジスタに値を代入するとき、T_EX は欲張りなので、数値と思しき記述をすべて飲み込んでしまうようです。ですから、この場合は「もう少し肩の力を抜いてよ」という意味で `\relax` というプリミティブを使います。`\relax` を次のように `\set` の中に入れるとどうなるでしょう。

```
\long\def\set#1#2{#1=#2\relax \the\@tempcnta\par}
```

結果は ‘[0] 5 5 [5], [5]’ となることから、ようやくうまく行ったようです。

このように数値（長さを含む）の代入においては、その数値の終わりに `\relax` などを補わなければ、値が意図した結果にならないことがあります。

▼ 1.4.2 長さ

カウンタと同じように「長さ」を表すレジスタがあり、これを新規に確保するには `\newdimen` 命令を使います。

```
\newdimen<レジスタ名>
<レジスタ名> = <数値|長さ>
```

数値の代入方法はカウンタレジスタの場合と同様にできますが、加えて長さを代入することが出来ます。さらに `\the`, `\number`, `\ronannumeral` も使用できますが大変なことになります。

L^AT_EX では次の六つの長さレジスタが既に確保されています。

```
\newdimen\@tempdima
\newdimen\@tempdimb
\newdimen\@tempdimc
4 \newdimen\dimen@ % 0 番
\newdimen\dimen@i % 1 番
\newdimen\dimen@ii % 2 番
```

一時的にどのマクロからでも使うことが出来ます。

何かの長さを示すときには、ある基準となる単位が必要になります。単位には絶対的な単位と相対的な単位の 2 種類があります。絶対的な単位にはメートル ‘m’ のように速度の変化しない（と言われている）光速が進むことのできる距離を基準にしているものもあります。

T_EX (pT_EX) には表 1.5 に示すような寸法が用意されています。

例えば `\parindent` の値を指定するには次のようにします。

```
\parindent=1zw
\parindent-10pt
\parindent 3 em
4 \parindent 1.5 zw
```

イコール ‘=’ はあってもなくても構いません。数値と単位のあいだの空白は適当に付けたら付けなかったりして構いません。また、単位の後のホワイトスペースは吸収されます。

表 1.5 寸法単位

単位	説明		実際の長さ
bp	ビッグポイント	72 bp = 1 in	
cc	シゼロ	1 cc = 12 dd	┌┐
cm	センチメートル	2.54 cm = 1 in	┌┌┌┐
dd	ディドーポイント	1157 dd = 1238 pt	
in	インチ	1 in = 72.27 pt	┌┌┌┌┌┐
mm	ミリメートル	10 mm = 1 cm	
pc	パイカ	1 pc = 12 pt	┌┐
pt	ポイント		
sp	スケールポイント	65536 sp = 1 pt	
em	現在の 'M' の幅		┌┐
ex	現在の 'x' の高さ		┐
zh	現在の和文の高さ		┌┐
zw	現在の和文の幅		┌┐

▷ 例題 1.15 次の記述を云々。

```
1 \@tempdima=65536sp
  \the\@tempdima, \number\@tempdima
```

結果は '1.0pt, 65536' となるでしょう。

▷ 例題 1.16 次の記述を云々。

```
\newdimen\dimena
\newcount\counta
3 \dimena=10.55pt \relax
  \the\dimena, \the\counta\par
  \counta=\dimena \relax
  \the\dimena, \the\counta\par
  \dimena=100sp \relax
8 \counta=\dimena \relax
  \the\dimena, \the\counta\par
```

結果は '10.55pt, 0' とか '10.55pt, 691405' とか '0.00153pt, 100' となるでしょう。このことからカウンタに長さを代入すると単位を sp としてその数値を代入するという事が分かります。sp は T_EX が扱うことが出来る最小の寸法単位です。

マクロの中で都合良く '1pt' や '0pt' を表記するために \p@ と \z@ の二つを次のように確保します。

```
1 \newdimen\p@ \p@=1pt
  \newdimen\z@ \z@=0pt
```

▷ 例題 1.17 次の記述を云々。


```

\the\p@, \number\p@\par
\@tempdima=\p@ \relax
3 \the\@tempdima, \number\@tempdima\par
\@tempdima=50\p@ \relax
\the\@tempdima, \number\@tempdima\par
\newdimen\dimena \dimena=50pt
\@tempdima=30\dimena \relax
8 \the\@tempdima, \number\@tempdima\par

```

結果は ‘1.0pt, 65536 1.0pt, 65536 50.0pt, 3276800 1500.0pt, 98304000’ となることから、50\p@ や 30\dimena と記述すると予めその長さ同士を掛けたものが代入されるようになります。しかし、カウンタの場合は同でしょうか。

```

\@tempcnta=30 \relax
2 \@tempcntb=5\@tempcnta\relax
\the\@tempcntb\par
\@tempcntb=50 \relax
\@tempcnta=\@tempcnta\@tempcntb \relax
\the\@tempcnta\par

```

カウンタの場合は **! Missing number, treated as zero.** なるエラーが発生し代入できません。どうやら長さレジスタだけに許される構文のようです。

▼ 1.4.3 スキップ

スキップは「長さ」「縮み率」「伸び率」の三つの数値を持つレジスタです。スキップは可変長の長さで、バネのような役割りをします。

```

\newskip<レジスタ名>
\newmuskip<レジスタ名>
<レジスタ名> = <長さ> plus <伸び率> minus <縮み率>

```

数式様には \newmuskip を使います。〈伸び率〉と〈縮み率〉は省略可能です。

「長さ」の時と同じ様に「スキップ」にも次のような空のスキップ \z@skip が確保されています。

```

\newskip\z@skip \z@skip=0pt plus 0pt minus 0pt

```

L^AT_EX では予め次の三つのスキップレジスタが確保されています。

```

\newskip\@tempskipa
\newskip\@tempskipb
\newskip\@skip@ % 0 番
4 %\newskip\@flushglue \@flushglue = 0pt 1fil

```

▷ 例題 1.18 次の記述を云々。

```

1 \@tempskipa=10pt plus 5pt \relax
\@tempskipb=5\@tempskipa \relax
\the\@tempskipa\par
\the\@tempskipb\par

```

結果は ‘10.0pt plus 5.0pt 50.0pt’ となります。長さの場合の代入と同じ様にできますが、〈縮み率〉と〈伸び率〉は消され、〈長さ〉だけが代入されることとなります。

▼ 1.4.4 ボックス

ボックスレジスタは何らかのボックスを保存することが出来ます。

```
\newbox<レジスタ名>
\setbox<レジスタ名> = <ボックス>
\box<レジスタ名>
```

ボックスレジスタに内容を保存するには `\setbox`、中身を使うには `\box` を用います。

「ボックス」も「長さ」の場合と同じ様に `\voidb@x` が確保されます。

```
1 \newbox\voidb@x % 中身は空になっている
```

L^AT_EX では次の使い回しの `\tempboxa` が用意されています。

```
\newbox\@tempboxa
```

▷ 例題 1.19 次の記述を云々。あるボックスの高さ、深さ、幅を求めるために `\ht`、`\dp`、`\wd` が使えます。中身を見たいときには `\the` を補います。

```
\setbox\@tempboxa=\hbox{いいい}
\the\ht\@tempboxa, \the\dp\@tempboxa, \the\wd\@tempboxa\par
\box\@tempboxa
```

結果は ‘7.77588pt, 1.38855pt, 28.86649pt いいい’ などとなります。

▷ 例題 1.20 とりあえず、以下の記述を云々。使用されているマクロまでは理解しなくても良いので、出力結果だけを吟味してください。

```
\makeatletter
2 \fboxrule=.3pt \fboxsep=0pt
\def\my@word@fbox#1{\@for\@tempa:=#1\do{\@tempa\space}}
\def\my@char@fbox#1{%
  \@tfor\@member:=#1\do{%
    \if\@member,\space \else
7 \framebox{\phantom{\@member}}\kern-\fboxrule
    \fi}}
\def\my@word@fbox#1{\@for\@member:=#1\do{\framebox{\@member}\space}}
\my@word@fbox{!,\TeX,capacity,exceeded.,sorry,[main,memory,size=1000001]}\par
\my@word@fbox{!,\TeX,capacity,exceeded.,sorry,[main,memory,size=1000001]}\par
12 \my@char@fbox{!,\TeX,capacity,exceeded.,sorry,[main,memory,size=1000001]}\par
\makeatother
```

概ね、次のようになります。

結果は、「♡ ほげほげ ♡ ほげほげ」となります。例えば `\everypar` は段落がまさに組まれようとしている時に実行されるトークンリストパラメータです。これを使って行数を表示することが出来るわけです。もう少し工夫すると次のようになります。

```

それぞれ。 \par
2 うんうん。 \par
% \begin{document} のあとでもここはまだ文章を組み上げる前の
% 垂直モードにある。
\makeatletter
%\parindent=\z@
7 \let\orig@par=\par
\newcount\cnt@lines
\cnt@lines=\@ne
\def\par{\advance\cnt@lines\@ne \orig@par}
\everypar={\llap{\scriptsize\@arabic\cnt@lines:\space}}
12 \makeatother
% ここで垂直モードの終了となり、文章の構成要素が表れるため、
% \par が自動的に呼び出された事と同義になるため、トークンリスト
% パラメータ \everypar が呼び出される。
あれあれ \par
17 これこれ \par
それぞれ \par
だれだれ \par
どれどれ \par
ほうほう \hfill そうそう \ \ こうこう。 \par

```

L^AT_EX での `\par` のオリジナルを `\orig@par` に保存しておきます。次に `\par` の定義を書き換えます。これは `\cnt@lines` という行数をカウントするカウンタ型変数となり、初期値を 1 にしておきます (行は 1 行めから始まるため)。次に `\everypar` なるパラメータに行数を表示するためのトークンリスト

```
\llap{\scriptsize\@arabic\cnt@lines:\space}
```

を代入します。なんだか、`verbatim` 環境に応用できそうな予感がしますね。

▼ 1.4.6 ブール値

「フラグを立てる」とか「スイッチを入れる/切る」とか色々と呼び方はありますが、L^AT_EX でもブール型の変数を使うことができます。

```

\newif\if<レジスタ名>
\<<レジスタ名>true (真にする)
\<<レジスタ名>false (偽にする)
\if<レジスタ名> <真の場合> \else <偽の場合> \fi

```

判断で使用する時、`\else` による <偽の場合> は省略可能です。

L^AT_EX では既に以下の `\if@tempswa` が確保されています。

```
\newif\if@tempswa
```

▷ 例題 1.22 以下の記述を云々。

```

\newif\ifcomment
\commenttrue
\ifcomment \else
4 コメントだよ
\fi
\commentfalse
\ifcomment \else
コメントかな
9 \fi

```

結果は「コメントかな」となります。

▼ 1.4.7 アロケーション——変数領域確保

L^AT_EX 2_ε ではあるデータ型の変数を確保する手段としてカウンタを使っています。例えば T_EX には 0–255 番までの ボックスレジスタ `\box` がありますが、0–9 番のレジスタは T_EX が使うものとして、他のマクロなどが使ってははいけません、これは L^AT_EX も例外ではありません。そこで L^AT_EX ではアロケーション（変数の領域を確保する）段階で 10 番からボックスレジスタを確保するように設定されています。

```

1 \count14=9 % \box レジスタは 10, 11, ..., 255 を使う

```

▼ 1.4.8 `\setbox0` とは

よく他人のマクロを覗くと

```

\setbox0=\hbox{#1}
\@tempdima=\wd0

```

などのような記述を見掛けることになります (L^AT_EX 2_ε ではこれは obsolete な方法として推奨されていません)。これは T_EX のボックスレジスタ 0 番に `\hbox` で #1 を組んだときの幅 `\wd` を `\@tempdima` という長さレジスタに代入するという文です。これと等価なことを L^AT_EX では

```

\newlength{\hoge}
\settowidth{\hoge}{#1}

```

等とすれば `\hoge` に #1 の幅が設定されることになります。

実は T_EX のアロケーション処理の実装を覗いてみるとわかるのですが、とりあえず、

```

\newbox\hakoA
\newcount\countA
3 \newdimen\dimenA
\newskip\skipA
\newtoks\toksA
\newif\ifA
hoge\bye

```

と書いた `\hoge.tex` を

```
tex hoge
```

として実行し、`<hoge>.log` を覗いて下さい。そうすると

```
This is TeX, Version 3.14159 (Web2C 7.4.5) (format=tex 2005.1.31)
 22 FEB 2005 17:37
**hoge
4 (./hoge.tex
  \hakoA=\box16
  \countA=\count26
  \dimenA=\dimen16
  \skipA=\skip18
9 \toksA=\toks12
  [1] )
Output written on hoge.dvi (1 page, 212 bytes).
```

という事になります。これは新規に `\hakoA` というのを用意するときに、T_EX は裏側で数字に置き換えている (`\box` レジスタの 16 番としてアロケーションしている) ということなのです。この事実から

```
\newbox\hakoA
\setbox16=\hbox{hoge}% 要するに \hakoA のこと
\box\the\hakoA % \box 16 に展開される
4 \bye
```

とすることにより `\hakoA (16)` に `\hbox{hoge}` を代入し、`'\the\hakoA'` により 16 番を取得して `\box` 命令でその内容を出力しています。別にこのような回りくどい方法を取らなくとも

```
1 \newbox\hakoA
  \setbox\hakoA=\hbox{hoge is hoge.}% これは \setbox \box 16 に展開されそうだが...
  \box\hakoA % これは \box \box 16 にされるのでは心配になるが適切に展開される
  \bye
```

としても同じ事です。

このような事実から

```
1 \setbox0=\hbox{hoge}
  \box0
```

とすると

```
hoge
```

が出力されるというのはお分かりでしょう。そして 0 番めのボックスレジスタはユーザーのアロケーションの対象外で、誰でも使っても良い一時的 (temp) なものということが伺えます。

▼ 1.4.9 数値表現

数値表現の別の方法として L^AT_EX では以下のマクロが頻繁に使われています。

L^AT_EX では予め、`lplain.dtx` にて文字列として次の数値が定義されています (`\@ne`, `\tw@`, `\thr@@`, `\sist@n`, `\@xxxii`, `\@cclv`)。

```
\chardef\@ne=1
\chardef\tw@=2
\chardef\thr@@=3
4 \chardef\sist@n=16
\chardef\@xxxii=32
\chardef\@cclv=255
```

255 を超える数値に関しては `\chardef` ではなく `\mathchardef` を使います (`\@cclvi`, `\@m`, `\@M`, `\@MM`)。

```
\mathchardef\@cclvi=256
\mathchardef\@m=1000
\mathchardef\@M=10000
4 \mathchardef\@MM=20000
```

10001–10004 (`\@Mi`, `\@Mii`, `\@Miii`, `\@Miv`) も同様に定義されています。

```
1 \mathchardef\@Mi=10001
\mathchardef\@Mii=10002
\mathchardef\@Miii=10003
\mathchardef\@Miv=10004
```

‘-1’ だけは - と 1 なので、これは `\countdef` を使い、カウンタとして定義しなければなりません。

```
1 \countdef\m@ne=22 \m@ne=-1
```

▷ 例題 1.23 以下の記述を云々。

```
\@tempdima=\z@ \relax \the\@tempdima\par% 0.0pt
\@tempdima=\tw@\p@ \relax \the\@tempdima\par% 2.0pt
\@tempdima=\@cclv\p@ \relax \the\@tempdima\par% 255.0pt
```

結果は「0.0pt 2.0pt 255.0pt」となるでしょう。

1.5 演算とか

さて、一通りデータ型をみてきたので、ここでどのような演算ができるのかちょっと考えてみましょう。それぞれのデータ型に応じた各種操作のコマンドが色々ありますが、ここではもっとも基本的なものを紹介します。

▼ 1.5.1 四則演算

T_EX での四則演算には「たしざん」「かけざん」「わりざん」が用意されています。「ひきざん」はマイナスを付けて「たしざん」を代用してください。

```
\advance<数値> by <値> (たしざん)
\multiply<数値> by <値> (かけざん)
\divide<数値> by <値> (わりざん)
```

by は省略可能です。ていうか、普通は付けません。

▷ 例題 1.24 次の記述を云々。

```
\@tempcnta=\tw@ % a = 2
2 \advance \@tempcnta by \thr@@ % a + 3 = 5
  \the\@tempcnta\par
  \multiply \@tempcnta by \@ne % a x 1 = 5
  \the\@tempcnta\par
  \divide \@tempcnta \tw@ % a / 2 = 2
7 \the\@tempcnta\par
```

結果は '5 5 2' って感じでしょう。

▷ 例題 1.25 次の記述を云々。

```
\@tempdima=\tw@ \p@ \relax % a = 2
{} \the\@tempdima\par
3 \advance \@tempdima \p@ \relax % a + 1 = 3
  {} \the\@tempdima\par
  \multiply \@tempdima \thr@@ % a x 3 = 9
  {} \the\@tempdima\par
  \divide \@tempdima \tw@ % a / 2 = 4.5
8 {} \the\@tempdima\par
```

結果は '2.0pt 3.0pt 9.0pt 4.5pt' となることから、なんだか長さの場合は `\multiply` と `\divide` で寸法なしで演算しなければならないようです。

▷ 例題 1.26 次の記述を云々。

```
\@tempskipa = \sixin@n\p@ \@plus \tw@\p@ \@minus \tw@\p@\relax
2 {} \the\@tempskipa\par
  \advance \@tempskipa \p@ \@plus \thr@@\p@ \relax
  {} \the\@tempskipa\par
  \multiply \@tempskipa \thr@@
  {} \the\@tempskipa\par
7 \divide \@tempskipa \tw@
  {} \the\@tempskipa\par
  \@tempskipa = -\@tempskipa \relax
  {} \the\@tempskipa\par
```

結果は次のようになります。

```
16.0pt plus 2.0pt minus 2.0pt
17.0pt plus 5.0pt minus 2.0pt
51.0pt plus 15.0pt minus 6.0pt
255.5pt plus 7.5pt minus 3.0pt
-25.5pt plus -7.5pt minus -3.0pt
```

`plus` とか `minus` は演算しとして使われているわけではなく、伸び率と縮み率を指定するためにあります。かけざん/わりざんをすると長さ、伸び率、縮み率が個別に演算されます。

試しに ‘YYYY/MM/DD/ hh:mm’ の形式でその原稿をタイプセットした日付を求めるマクロを作成します。あらかじめ `\time` というコマンドにその日の 00:00 から経過した「分」が保存されているものとします。一時用のカウンタ `\count0`（「時」用）と `\count2`（「分」用）が用意されているとします。もちろん年月日はそれぞれ `\year`, `\month`, `\day` にあります。

```
\begingroup
% \time には 00:00 から経過した分数が保存されている
\count0=\time % \count0 = \time
\divide \count0 by 60 % (\time / 60) = 「時間」
5 % ここから「分」を求める
\count2=-\count0 % -「時間」
\multiply \count2 by 60 % (-「時間」 x 60)
\advance\count2 by \time % \time - (「時間」 x 60) = 「分」
% ‘YYYY/MM/DD/ hh:mm’ を出力する \now 命令
10 \edef\now{\the\year/\two@digits{\the\month}/\two@digits{\the\day}~%
    \two@digits{\the\count0}:\two@digits{\the\count2}}
今日は \now です。
\endgroup
```

気を付けることとしては「分」を求めるときに行なう演算の順序です。括弧の中を先に計算することで必要とする変数を少なくしています。また、‘2005/1/3 1:23’ の場合には ‘2005/01/03 01:23’ となるように `\two@digits` を使っています。レジスタの内容を表示するのはもちろん `\the` でしたね。

▼ 1.5.2 柔軟な演算——calc

T_EX/L^AT_EX の四則演算ではどうも頼りないので、calc パッケージの力を借りるのも一つの方法です。calc パッケージに関しては『好き好き L^AT_EX 2_ε マクロ活用編』を参照してください。

1.6 条件判断

「こっちのりんごのほうが大きいかな」「いやこっちのほうが甘いはず」とか、人間は様々な状況で判断をすることがあります。T_EX では「文字」「文字列」「カテゴリーコード」「文字コード」「長さ」「奇数」「トークン」などのデータ型に応じた条件判断用のコマンドがあります。判断の形式は基本的に次のように `\if`, `(\else,)` `\fi` を含んでいます。

```
\if<条件> <真の場合> \else <偽の場合> \fi
```

`\else` とか `<真の場合>`、`<偽の場合>` なんかは省略可能ですが、`\fi` だけは省略出来ません。これは分岐の終わりを示すために必ず必要です。

▼ 1.6.1 トークンの判断

一文字同士の文字コードを比較するには `\if` コマンドを使います。

```
\if<トークン1><トークン2>
```

いずれかのトークンが制御綴の場合、マクロなどは可能なかぎり展開されます。

次の場合は全て真になります。

```
\long\def\hoge#1#2{\if#1#2 true\else false \fi\par}
2 \hoge a a% true
\def\A{A} \def\B{A}
\hoge A \A% true
\hoge \A \B% true
```

\A とか \B とかは展開すると 'A' になるので、すべて 'true' が出力されます。しかし、次のようなプリミティブを比較すると結果は 'true false' となります。

```
\long\def\hoge#1#2{\if#1#2 true\else false \fi\par}
\hoge \let \par % true
\def\A{A}
\hoge \A \par % false
```

コマンド \let などではそれ以上展開することが出来ないプリミティブであり展開されつくしたトークン `\let` になった時に、実はカテゴリコードは 16 として認識されていることとなります。このとき文字コードは 256 として特別扱いされます。

文字コードではなくカテゴリコードを比較するには \ifcat なるプリミティブを使います。

```
\ifcat<トークン1><トークン2>
```

次の例では結果は 'true false true' となります。

```
1 \long\def\hoge#1#2{\ifcat#1#2 true\else false\fi\par}
\hoge * . % true
\hoge A 1 % false
\hoge \par \let % true
```

'A₁₁' と '1₁₂' ではもちろん 'false' になります。 \par と \let は共にプリミティブ `\par` と `\let` でカテゴリコードは 16 となるので 'true' になります。

トークン同士の性質が同様かどうかを比較するには \ifx を使います。

```
\ifx<トークン1><トークン2>
```

トークン同士の性質と言ってもピンと来ないので、まずは以下の例を実行してみてください。

```
1 \long\def\hoge#1#2{\ifx#1#2 true\else false\fi\par}
\hoge * * % true (カテゴリコードも文字コードも同一)
\hoge \relax \relax % true (まったく同じプリミティブ)
\hoge \par \let % false (同じカテゴリコード、文字コードでも、中身が違う)
\def\A{A} \def\B{A} \def\C{A} \def\D{B}
6 \hoge \A A % false (制御綴と単なる文字では偽になる)
\hoge \A \C % false (展開すると同じだが、\C は 2 回の展開が必要なのでだめ)
\hoge \C \D % false (1 回目の展開で \A, \B となるので、これも偽)
\hoge \A \B % true (1 回目の展開で A, A となるので真)
\hoge \hoge \dame % false (\hoge は定義済み、\dame は未定義)
11 \hoge \geho \dame % true (\geho, \dame 共に未定義)
```

星同士 '*' はもちろん真になります。 \if, \ifcat の場合とは別に \par と \let 同士では偽になります。 \A と \B は一回目の展開が行なわれたときに両者が 'A' になるので真

になります。 `\ifx` では両者のトークンが制御綴であった場合、一回目の展開で中身が等しければこれも真になります。加えて未定義の制御綴同士も真になります。

例えば p_TE_X と j_TE_X のいずれかを使用しているかを判別するために `\fmtname`, `\jfmtname`, `\pfmtname` を調べることがあります。

▼ 1.6.2 整数の判断

整数やカウンタレジスタを比較するには `\ifnum` プリミティブを、奇数かどうかを判断するには `\ifodd` を使います。

```
\ifnum<数値1><関係演算子><数値2>
\ifodd<数値>
```

関係演算子は `<`, `>`, `=` のいずれかしか使用できません。

例えば日付の '2005/1/3' を '2005/01/03' のように一桁だけの数値に自動的に 0 を付加するには `\two@digits` なる命令を使います。

```
\def\two@digits#1{\ifnum#1<10 0\fi\number#1}
```

さらに三桁以上の場合でも同様に定義できます。

```
% 100 以下ならば 10 以下であることは明白なので \ifnum の中に
% それ以降の \ifnum をいれてしまう。このようにすることで判断
% の回数を減らすことができる。
4 \def\three@digits#1{\ifnum#1<100 0\fi \ifnum#1<10 0\fi\number#1}
```

▷ 例題 1.27 以下の記述を云々。

```
1 \@tempcnta=\z@
\ifnum \@tempcnta >0 true\fi
\def\hoge{\ifnum \@tempcnta<11 \the\@tempcnta,\space
\advance \@tempcnta \@ne \hoge\fi}
\hoge
```

結果は '0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,' となります。これは『もしも `\@tempcnta` が 11 未満ならば、その内容を出し、`\@tempcnta` をインクリメントしてから再帰的に `\hoge` を呼び出す』ということを行なっていますので、擬似的な for 文を実現しています。しかし、実は `\ifnum` による擬似的な再帰処理を行なうときに `\fi` が `\hoge` の後方にあるため、`\hoge` が `\ifnum` の処理を終わる前よりも先に実行されることとなります。そのため、先に `\ifnum` を終わってもらうようにするためには `\expandafter` なるプリミティブを使うこととなります。とりあえず次のようにするのが T_EX に親切な方法です。

```
\def\hoge{\ifnum \@tempcnta<11 \the\@tempcnta,\space
\advance \@tempcnta \@ne \expandafter\hoge\fi}
```

場合分けによって数値を判断するには `\ifcase` 文を用います。

```
\ifcase<数値><0 の場合>\or <1 の場合>\or … \or <n の場合>\else <それ以外>\fi
```

例えば 太陽暦の 5 月を 太陰暦の 皀月に自動的に変換したいときに (陽暦の 5 月が陰暦の 皀月ではないというツッコミはおいといて) `\ifcase` なる分岐命令が使えます。

さて、次のように定義すると「5月 は陰暦で皐月でしょう。」という出力になります。

```
\newcommand*\oldreki[1]{%
  \ifcase #1 \or 睦月\or 如月\or 弥生\or 卯月\or 皐月\or
3 水無月\or 文月\or 葉月\or 長月\or 神無月\or 霜月\or 師走\fi}
5 月は陰暦で\oldreki{5}でしょう。
```

▼ 1.6.3 長さの判断

単位付きの長さを比較するには `\ifdim` を使います。

```
\ifdim<寸法1><関係演算子><寸法2>
```

▷ 例題 1.28 以下の記述を云々。

```
1 \@tempdima=\thr@@\p@ % a = 3pt
   \ifdim \@tempdima > \p@ true\else false\fi\par
   \ifdim 10pt < 11pt true\else false\fi\par
   \ifdim \@tempdima = \thr@@\p@ true\else \false\fi\par
```

結果は ‘true true true’ となります。

▼ 1.6.4 常に真、常に偽

なんに使えるのかちょっと悩んでしまいますが、「常に真になる」とか「常に偽になる」というプリミティブが用意されています。

```
\iftrue (常に評価が真になる)
\iffalse (常に評価が偽になる)
```

とりあえず、以下の具体例を実行してみてください。うまく行けば「うぐう」と表示されるはずです。

```
\iftrue % -> true になる
う% true
\else
ここは実行されない。% false
\fi
                                     うぐう
\iffalse % -> false になる
ここも実行されない% true
\else
ぐう% false
\fi
```

▷ 例題 1.29 これは `\let` を使うと面白いことが出来ます。

```
1 \iftrue
   ここは表示される
   \fi
   \iffalse
```

```

    ここは常に表示されない
6  \fi
    % \comment \endcomment を作る
    \let\comment\iffalse
    \let\endcomment\fi
    \comment
11 ここもコメントアウトされる。
    \endcomment
    \renewenvironment{comment}{\iffalse}{\fi}
    \begin{comment}
    ここはどうなるだろうか
16 \end{comment}

```

どうやら `\renewenvironment` で定義した `comment` 環境ではエラーになるようです。これはまた別の話ですが...

▼ 1.6.5 柔軟な条件判断——`ifthen`

Leslie Lamport 氏と David Carlisle 氏による `ifthen` パッケージを使うと、柔軟な条件判断のコマンドが使えるようになります。

```

\ifthenelse{<条件文>}{<真の場合>}{<偽の場合>}
\whiledo{<条件文>}{<処理>}

```

`\ifthenelse` `\whiledo` における `<条件文>` としては長さの比較 (`\lengthtest`)、奇数かどうかの判定 (`\isodd`)、文字列の比較 (`\equal`) などがあります。

```

\boolean{<名前>} (ブール値を評価)
\isodd{<数値>} (奇数かどうか)
\equal{<文字列1>}{<文字列2>}
\lengthtest<<長さ1> <関係演算子> <長さ2>>

```

とりあえず、ブール型変数に関するコマンドも追加されています。

```

\newboolean{<名前>}
\setboolean{<名前>}{<true|false>}

```

▷ 例題 1.30 以下の記述を云々。

```

あれは\ifthenelse{\isodd{\pageref{are}}}{奇数}{偶数}ページにあります。
この文書は\ifthenelse{\boolean{draft}}{完成版}{未完成版}です。

```

結果は「あれは奇数ページにあります。この文書は完成版です。」などとなるでしょう。

▷ 例題 1.31 以下の記述を云々。

```

\def\A{ほげ} \def\B{ほげ}
う、\ifthenelse{\equal{\A}{\B}}{うみいー}{うぐう}\par
3 多分\ifthenelse{\lengthtest{\textwidth < \textheight}}{縦置き}{横置き}だろう

```

結果は「う、うみいー 多分縦置きだろう」となるでしょう。

▷ 例題 1.32 以下の記述を云々。

```

\newboolean{hoge}
2 \ifthenelse{\boolean{hoge}}{true}{false}
\setboolean{hoge}{true}
\ifthenelse{\boolean{hoge}}{true}{false}

```

結果は ‘false true’ となります。

▷ 例題 1.33 以下の記述を云々。

```

1 \newcounter{hoge}
\setcounter{hoge}{100}
\whiledo{\value{hoge}<107}{\roman{hoge}\space\stepcounter{hoge}}

```

結果は ‘c ci cii ciii civ cv cvi cvii cviii’ となります。

`\and`, `\or`, `\not`, `\(`, `\)` を使うと複合条件判断もできます。

▷ 例題 1.34 以下の記述を云々。

```

\newboolean{A} \newboolean{B} % 入力 A/B
2 \newcommand*\TF[1]{\ifthenelse{\boolean{#1}}{1}{0}} % 論理値を出す
\newcommand*\SET[2]{\setboolean{A}{#1}\setboolean{B}{#2}} % 入力 A/B の値の設定
\newcommand*\AND[2]{% 論理積
AND(\TF{A}, \TF{B}) $=$ \ifthenelse{\boolean{A}\and\boolean{B}}{1}{0}}
\newcommand*\XOR[2]{% 排他的論理和
7 XOR(\TF{A}, \TF{B}) $=$ \ifthenelse{%
\(\(\not \boolean{#1}\) \and \boolean{#2}\)\or
\(\ \boolean{#1} \and \(\not \boolean{#2}\)\)}{1}{0}}
\SET{false}{false} \AND{A}{B}, \XOR{A}{B}\par
\SET{false}{true} \AND{A}{B}, \XOR{A}{B}\par
12 \SET{true}{false} \AND{A}{B}, \XOR{A}{B}\par
\SET{true}{true} \AND{A}{B}, \XOR{A}{B}\par

```

結果は次のようになります。

```

AND(0, 0) = 0, XOR(0,0) = 0
AND(0, 1) = 0, XOR(0,1) = 1
AND(1, 0) = 0, XOR(1,0) = 1
AND(1, 1) = 1, XOR(1,1) = 0

```

1.7 マクロの調べ方

他人が作成したマクロを使っていて、そのマクロの定義等を調べたいときがあります。または単にレジスタの内容が知りたいときがあります。これらを調べるには `\meaning` とか `\show` などのプリミティブが使えます。

▼ 1.7.1 定義を調べる

例えば、長さレジスタ `\@tempdima` が何番にアロケーションされているかを調べるには `\show` というプリミティブを使って調べます。

```
\show<トークン>
```

`\show` に与える引数の `<トークン>` は制御綴でもレジスタでも文字でも、なんでも構いません。T_EX が一つのトークンとして読めるものであれば何を書いても答えてくれます。

長さレジスタ `\@tempdima` のアロケーションを調べるには次のようにします。

```
\makeatletter
2 \show\@tempdima
\makeatother
```

`\@tempdima` の結果が得られます。この場合は結局のところ `\@tempdima` というトークンを展開すると `\dimen14` になるという事が分かります。

```
> \@tempdima=\dimen14.
1.2 \show\@tempdima
```

コンソールに '?' と表示されますが、構わず `Enter` を押してしまつて構いません。

▷ 例題 1.35 次の記述を云々。

```
\show\relax % プリミティブ
\show\everypar % プリミティブ
3 \show\hoge % 未定義
\show\show % プリミティブ
\show\section % 節見出し
```

結果は次の通りです。

```
> \relax=\relax.
> \everypar=\everypar.
> \hoge=undefined.
> \show=\show.
> \section=\long macro:
->\@startsection {section}{1}{\z@ }{1.5\Cvs \@plus .5\Cvs \@minus .2\Cvs }{.5\Cvs \@plus .3\Cvs }{\reset@font \Large \bfseries }.
```

`\relax`, `\everypar`, `\show` はプリミティブなので、そのまま同じ表示になります。`\hoge` というコマンドが未定義の場合は 'undefined' と言われてしまいます。`\section` は改段落を含んでも良い、(`\long`) を伴うマクロだと分かりますし、`\@startsection` という別のマクロを呼び出していることになっています。

▼ 1.7.2 中身を調べる

長さレジスタ `\@tempdima` の場合はその中身の数値が知りたいときがあるのでレジスタの内容を表示するプリミティブ `\showthe` があります。

```
\showthe<レジスタ名>
```

<レジスタ名>には `\toks`, `\dimen`, `\skip` などが使えることになります。

▷ 例題 1.36 次の記述を云々。

```
\newcommand\myshowthe[2]{#1=#2\relax \noindent\show#1, \showthe#1\par}
\myshowthe{\@tempdima}{1cm}
3 \myshowthe{\@tempskipa}{1cm plus 1mm minus 1mm}
\@temptokena={\section}
\noindent \show\@temptokena, \showthe\@temptokena\par
\setbox\@tempboxa=\hbox{show}
\noindent \show\@tempboxa, \showthe\@tempboxa\par
```

ただし `\box` の場合はうまくいっているのかわかりませんね。これはまた別のコマンドがありそうです。

しかし、毎回タイプセットの中断をして `\show` コマンドで定義を調べるのも面倒なので、そのまま文書にタイプセットしてくれる `\meaning` なるプリミティブがあります。

```
\meaning<トークン>
```

▷ 例題 1.37 次の記述を云々。

```
\ttfamily
\meaning\relax \par % プリミティブ
3 \meaning\everypar \par % プリミティブ
\meaning\hoge \par % 未定義
\meaning\show \par % プリミティブ
\meaning\section % 節見出し
```

当たり前ですが `\show` の場合と同じ内容になっています。レジスタ等の中身を配置したいときは `\the` プリミティブを使えば良いだけです。

```
\ttfamily
\newcommand\myshowthe[2]{#1=#2\relax \the#1\par}
\myshowthe{\@tempdima}{1cm}
4 \myshowthe{\@tempskipa}{1cm plus 1mm minus 1mm}
```

▼ 1.7.3 頑丈なマクロを調べる

例えば `\LaTeX` のような頑丈 (robust) なコマンドの定義内容を調べようと `\meaning\LaTeX` とすると `macro:->\protect \LaTeX` となるため、本来の定義内容を知ることが出来ません。L^AT_EX 標準のコマンドであれば大抵は `source2e.tex` をタイプセットしたものから検索することができます。どこからやってきたのかも分からない場合も含めて `\protect` されているコマンドは次のようにすると簡単に定義内容を調べることが出来ます。

```
\let\protect<\show|\meaning><制御綴>
```


▷ 例題 1.38 次の記述を云々。

```

1 \long\def\ccs#1{\noindent \bgroup \ttfamily
   \string#1\space$=$\space\meaning#1\egroup\par}
   % \TeX は大抵 robust ではない、\LaTeX は robust の場合が多い
   \ccs \TeX \ccs \LaTeX
\long\def\robustccs#1{\noindent \bgroup \ttfamily
6  \let\protect\meaning \string#1\space$=$\space#1\egroup\par}
   % \protect されていないもの使うと、#1 がそのまま出力される
   \robustccs \TeX \robustccs \LaTeX

```

`\protect` を付加されていないコマンドはそのままコマンドが文中で使われたことと同じになります。`\string` はとりあえず、この場合はバックスラッシュを普通の文字に変換できる特殊なコマンドとして解釈してください。

1.8 そのほか気を付けること

▼ 1.8.1 何もしないけど役に立つ命令

ああ、たりい、めんどくせえ。

```

ここで文章が終わる\\
2 [{\Large みだし}]\
ここから文章が始まる。

```

はエラーになるぞお。

```
ここで文章が終わる\\\relax
```

とか

```
ここで文章が終わる\{\}
```

とか

```
ここで文章が終わる\\\empty
```

などとしないとだめぽ。

要するに T_EX は改行も一つのスペースにしてしまうので、次の行の先頭にある要素も前の行の引数と誤認する可能性があります。また数値表現等が次の行に表れるとこれも引数として取られることがあります。ですから、数値表現のあとや任意引数を取るようなマクロのあとには `\relax` を補うようにします。`\empty` の場合はトークンとして T_EX が処理するときに吸収されてなくなります。

▼ 1.8.2 ホワイトスペースの扱い

T_EX では次のような規則に基づいてタブ、空白、改行を処理しています。

- 二つ以上の連続するスペースは一つの空白とする。
- 一つの改行は一つのスペースになる。
- 連続した改行は改段落とする。

第 2 章

L^AT_EX 解体

▼ 2.0.1 L^AT_EX 流の定義

マクロの〈制御綴〉にアルファベット（のカテゴリコードを与えられた）以外の文字、例えばアットマーク '@' やコロン ':' などが含まれていると、カテゴリコードが異なるので、正しく定義できません。

```
\def\hoge!hoge{ほげほげ}
```

上記の `\hoge!hoge` は アットマークのカテゴリコードが変更されていたとしても感嘆符 '!' にて文字の区切りができるので、`\hoge` とその後には `!hoge` が必ず伴って用いられるという意味になってしまいます。

どのような記号が含まれていても、その〈制御綴〉でマクロを定義したいときは次のようにします。

```
\def\mydef#1#2{\expandafter\def\csname #1\endcsname{#2}}
\def\myuse#1{\csname #1\endcsname}
\mydef{hoge!hoge}{ほげほげ}
4 \myuse{hoge!hoge}
```

```
\@namedef{<コマンド名>}
```

```
\@nameuse{<コマンド名>}
```

```
1 \def\@namedef#1{\expandafter\def\csname #1\endcsname}
\def\@nameuse#1{\csname #1\endcsname}
```

```
\@ifnextchar <文字>{<真の場合>}{<偽の場合>}
```

```
\@ifstar {<真の場合>}{<偽の場合>}
```

```
\@dblarg{<コマンド名>}{<引数>}
```

`\@ifstar` と `\@ifnextchar` の使い方

```
\documentclass{article}
\makeatletter
3 \def\hoge{\@ifstar{star \@hoge}{\@hoge}}
\def\@hoge{\@ifnextchar [{\@hoge}{\@hoge[\@empty]}}
\def\@hoge[#1]#2{[#1](#2)}
\makeatother
\begin{document}
8 \hoge[do]{re}\par
\hoge*[a]{jo}\par
```

```
\hoge{sfoa}\par
\hoge*{sfal}\par
\end{document}
```

次のような使い方もあります。

```
\def\@sore#1{[#1],\space}
\def\sore{\@ifnextchar \bgroup {\@sore}{\@sore{default}}}
3 \makeatother
\sore{hoge}
\sore
```

```
\documentclass{article}
\begin{document}
\makeatletter
\newcount\dot@point
5 \dot@point=\@ne
\def\@dot@points[#1]{% 描画する点の大きさ
  \dot@point=#1\relax
  \dot@points
}
10 \def\dot@points(#1,#2){% 再帰的に点を描画するための命令
  \put(#1,#2){\circle*{\dot@point}}%
  \@ifnextchar({\dot@points}{})%
}
\def\dottedpoints{% 点の大きさが指定されているかどうかの判定
15  \ifnextchar[{\dot@points}{\dot@points}]%
}
\makeatother
\begin{picture}(100,100)(0,0)
\setlength\unitlength{1pt}
20 \dottedpoints(0,0)(10,10)(20,50)(20,100)
\dottedpoints[10](100,100)(50,0)(50,50)(30,30)(20,20)
\end{picture}
\end{document}
```

問題、星を付けたときだけに中黒にするにはどうすれば良いでしょうか？ ヒントは `\ifstar` とか

```
\@ifundefined{<コマンド名>}{<真の場合>}{<偽の場合>}
\@ifdefinable{<制御綴>}{<真の場合>}
```

▷ 例題 2.1 次の実行結果から `\undefined` や `\empty`, `\@empty` などの意味をもう少し理解してください。 未定義何か苦 `undefined`

定義済み何か苦 `macro:->hoge`

定義済み空苦 `macro:->`

このことから L^AT_EX の `\@ifundefined` は何をしていると考えられるでしょうか。

```
\expandafter\ifx\csname#1\endcsname\relax
2 \expandafter\@firstoftwo
\else
\expandafter \@secondoftwo\fi
```

色々方法があるのですが、まずはすでに L^AT_EX で定義されている命令を見てみましょう。

```
1 \def\@ifundefined#1{\expandafter\ifx\csname #1\endcsname\relax
  \expandafter\@firstoftwo\else\expandafter\@secondoftwo\fi}
\long\def\@firstoftwo#1#2{#1}
\long\def\@secondoftwo#1#2{#2}
```

まあ、ちょっとした命令を作ってみますか。結果はいかほど。

```
\long\def\CheckDefine#1{%
  \expandafter\ifx\csname #1\endcsname
  未定義 \else 定義済み\fi}
\CheckDefine{hoge} \CheckDefine{document}
```

ここで `\ifx` の一つ目の引数は、`\CheckDefine` で与えられた一つ目の引数となります。しかし、二つ目の引数はなぜか `\relax` になっています。`\relax` は一体どのような定義になっているのでしょうか。これは調べる必要があります。

冗長性

```
1 \def\@cons#1#2{\begingroup\let\@elt\relax\xdef#1{#1\@elt #2}\endgroup}
\def\@car#1#2\@nil{#1}% 先頭だけ取り出す
\def\@cdr#1#2\@nil{#2}% 先頭だけ取り除く
\def\@carcube#1#2#3#4\@nil{#1#2#3}%
```

▼ 2.0.2 頑丈なコマンドの定義

ある時点で展開すると不都合が起きる（外部ファイルに書き出す事が想定される）場合には `\protect` などによりコマンドを保護します。あらかじめ `\protect` を付加して定義するには `\DeclareRobustCommand` を使います。

```
\DeclareRobustCommand{<制御綴>}[<数値>][<標準値>]{<定義内容>}
```

▷ 例題 2.2 次のようにして `\LaTeX` がどのように定義されているのかを調べてください。

```
1 \def\check@meaning#1{\bgroup\ttfamily
  \expandafter\meaning\csname#1\endcsname\relax\egroup\par}
\check@meaning{LaTeX}
```

結果は次のようになります。

```
\protect \LaTeX
```

続いて、`\protect` を無理矢理書き換えることで `\LaTeX` の定義を調べてみてください。

```
{\let\protect \show \LaTeX}
```

定義時には `\DeclareRobustCommand` が使われていることを理解してください。結果から `\DeclareRobustCommand` は何をしていると考えられますか。次のように入力してその答えを確認してください。

```
\check@meaning{DeclareRobustCommand}
\check@meaning{declare@robustcommand}
```

さらに `\@star@or@long` は何をするマクロかを考えてください。

`\@star@or@long` は次のように定義されています。

```
\def \@star@or@long#1{%
  \ifstar{\let\l@ngrel@x\relax#1}{\let\l@ngrel@x\long#1}%
3 }
\let\l@ngrel@x\relax
```

`\newcommand` は次のように定義されています。

```
1 \def\newcommand{\@star@or@long\new@command}
```

このことから星付きの場合は `\long` を付けずに定義するということになります。

▷ 例題 2.3 L^AT_EX ではカウンタを確保するのに `\newcounter` を使うこともできます。
`\newcounter` によって確保したカウンタ `hoge \arabic` と `\@arabic` の違いを理解してください。

```
\def\check@meaning#1{\bgroup\ttfamily
  \expandafter\meaning\csname#1\endcsname\relax\egroup\par}
\check@meaning{arabic}
4 \check@meaning{@arabic}
```

上記の結果から `\arabic` と `\@arabic` の定義が

```
1 \def\arabic#1{\expandafter\@arabic\csname c@#1\endcsname}
\def\@arabic#1{\number#1}
```

であると分かります。そうすると `\arabic{page}` はできても `{\arabic{c@page}}` はできなし、`\@arabic{c@page}` はできても `\@arabic{page}` はできないことが分かります。

▷ 例題 2.4 次の例から `\@fnsymbol` では最大いくつまで表示できますか。

```
\check@meaning{fnsymbol}
\check@meaning{@fnsymbol}
```

▷ 例題 2.5 次の結果から `\@alph` はどのような限界があると考えられますか。

```
\@tempcnta= 3 \@alph\@tempcnta\space
\@tempcnta=13 \@alph\@tempcnta\space
3 \@tempcnta=25 \@alph\@tempcnta\space
\@tempcnta=26 \@alph\@tempcnta\space
\@tempcnta=27 \@alph\@tempcnta\space
```

▷ 例題 2.6 次の出力結果から `\@roman` はどのようなキャパシティを持っていると考えられますか。

```

\@tempcnta= 3 \@roman\@tempcnta\space
\@tempcnta=13 \@roman\@tempcnta\space
\@tempcnta=23 \@roman\@tempcnta\space
\@tempcnta=50 \@roman\@tempcnta\space
5 \@tempcnta=103 \@roman\@tempcnta\space
\@tempcnta=255 \@roman\@tempcnta\space

```

▷ 例題 2.7 \@roman の例を参考に \@alph を次のような動作がするように拡張してください。

a, b, c, ..., z, aa, ab, ac, ..., az, ba, bb, bc, ...

▷ 例題 2.8 \pagenumbering と等価な \pagenokazu を作成してみてください。
 \@roman\c@page とすれば良いのでおおむね \c@page を 1 に初期化し、恐らく
 \thepage を \csname @#1\endcsname\c@page と定義すれば良いことになります。

```

\def\pagenokazu#1{%
  \c@page=\@ne
  \def\thepage{\csname @#1\endcsname\c@page}}

```

しかし ltpageno.dtx では \global が使われています。その理由を考えてください。

```

\def\pagenonumbering#1{%
2  \global\c@page\@ne
  \gdef\thepage{\csname @#1\endcsname\c@page}}

```

2.1 array/tabular 環境実装への道 其の一

まずは \hbox と \vbox に関する説明から、\hbox は中身を水平に組む、途中改行はなし。
 \vbox は中身を垂直に組む。まずは以下の出力例を吟味した方が良いでしょう。

```

% これは割とうまく組れる
2 \fbox{\vbox{\hbox{a}\hbox{b}\hbox{c}}}\par
% \vbox は組むべき幅が \linewidth だと誤解する
\fbox{\hbox{\vbox{a}\vbox{b}\vbox{c}}}\par
% これも割とうまく組まれる
\fbox{\hbox{\hbox{a}\hbox{b}\hbox{c}}}\par
7 % \vbox は \vbox のなかでは 幅が分からず
\fbox{\vbox{\vbox{a}\vbox{b}\vbox{c}}}\par

```

実質的に 長さを特別に指定しない場合はこのような恐いことになるので \vbox のなか
 では \hbox, \hbox のなかでは \vbox ということになります。T_EX 上では \fbox とか
 \framebox などは定義されていないので L^AT_EX 側で \hrule/Cvrule を使って

```

\def\waku#1{\ifvmode\leavevmode\fi
2  \hbox{\vrule\vbox{\hrule\hbox{hoge}\hrule}\vrule}}

```

などとしています (実際はもうちょっと手を加える)。さきに \vrule で囲む方法もちろ
 んあります。

```

\def\vwaku#1{\ifvmode\leavevmode\fi
  \vbox{\hrule\hbox{\vrule\hbox{#1}\vrule}\hrule}%
3 }

```

次に 3 行 3 列の単純な表を作ることを考えます。先ほどの `\hbox` と `\vbox` を使って簡単な例題を見たので、ある程度は分かるでしょう。

2.2 環境型コマンドの提供

次のファイル `<hoge>.tex` を `ptex` (`tex`, `pdftex` でも) でタイプセットしてみてください。

```

\let\TeXorigEnd\end
2 \def\begin#1{\csname #1\endcsname}
\def\end#1{\csname end#1\endcsname}
%
\long\def\newenvironment#1#2#3{%
  \expandafter\def\csname #1\endcsname{#2}%
7 \expandafter\def\csname end#1\endcsname{#3}%
}
%
\newenvironment{document}{\relax}{\TeXorigEnd}
\newenvironment{math}{${}$}
12 \newenvironment{displaymath}{${}$}
\def\({$}%$
\def\){$}%$
\def\[{$$}%$$
\def\[{$$}%$$
17 %
\begin{document}
Hello, World!!
\begin{math}
f(x) = ax + b
22 \end{math}
Hello, World!!
\begin{displaymath}
E = mc^2.
\end{displaymath}
27 hoge $a^2$, hoge.
\[
f'(u) = f(u)(1-f(u))
\]
\end{document}

```

何をやっているのか、吟味して下さい。

▷ 例題 2.9 `\begin` と `\end` は何をする命令でしょうか。

```

\check@meaning{begin}
\check@meaning{end}

```

それ以降、何らかの属性を変更する宣言型命令は環境にすることができます。

```
\begin{ttfamily}
```

ここは Typewriter 体になるんでしょうか?‘

ここは Typewriter 体になるんでしょうか;`

```
\end{ttfamily}
```


2.3 外部ファイルの書き出し/読み出しをする

`\section` や `\caption` の引数は必ず展開されるようになっています。これは次の場合を想定しています。

```
\documentclass{jarticle}
\begin{document}
3 \tableofcontents% この時点では \hoge の定義内容を知らない
\def\hoge{foo}% ここで \hoge が定義されている
\section{\hoge}% 見出しを出力すると同時に \jobname.aux に
% \@writefile{toc}{\contentsline {section}{\numberline {1}foo}{1}}
% を書き出す。 \jobname.toc には
8 % \contentsline {section}{\numberline {1}foo}{1}
% が文書の組版が終わった段階 (2 度目の \jobname.aux の読み込み時点)
% で書き出される。
\end{document}
```

この場合、`\tableofcontents` は `\jobname.toc` の中身を参照します。これは `\section` 命令によって書き出されたものです。もしも、`\hoge` を展開しなかったらあいは `\hoge` の定義の前に 目次を出力する羽目になります。`\section/\caption` などの引数の中身を展開することにより、後方参照マクロの未定義エラーを回避しているとも考えられます。

しかし、しばしばこの展開が悪さをすることがあります。

```
\section{見出し  $f(x)=ax+b$ }
\section{見出し  $\(f(x)=ax+b\)$ }
```

とした場合には $\$$ は T_EX のプリミティブなので頑丈 (robust) です。しかし、`\(, \)` は L^AT_EX のマクロなので `\section` などの動く (moved) 引数の中では脆弱 (fragile) です。

2.4 長さから ‘pt’ を取り除く

```
\@tempdima = 100.5pt
```

として `\@tempdima` を設定したとしたならば

```
\the\@tempdima
```

で

```
100.5pt
```

というトークンが得られるが、この 100.5pt から単位の ‘pt’ を取り除きたいときがしばしばある。単純に `\@tfor` で

```
\makeatletter
\def\hoge{\the\linewidth}
\@tfor\temp@str:=\hoge\do{%
4 \if\temp@str p\relax
\else
```

```
    \if\temp@str t\relax
    \else
      \temp@str
9    \fi
    \fi
  }
\makeatother
```

とできるというわけではない。ここで一つ厄介なのは ‘pt’ のカテゴリーコードである。と、安易にここでカテゴリーコードを変更してもいけない。

(執筆中)

第 3 章

クラス解体

3.1 クラスファイルの設計

▼ 3.1.1 最小構成

いきなり大規模なクラスファイルを設計仕様と思っても無理なので、まずは最小構成のクラスファイルを作ります。そこで次のクラスファイル `hoge.cls` を作成します。

```
\ProvidesClass{hoge}[2004/08/09 LaTeX2e class file]
\renewcommand\normalsize{\fontsize{10pt}{12pt}\selectfont}
\endinput
```

実は `\ProvidesClass` と `\renewcommand` の 2 行があればうまく動きます。多分そのうち日本語 `TeX` にも依存したクラスになるだろうと思いますから `\NeedsTeXFormat` を宣言しておきます。

```
\NeedsTeXFormat{pLaTeX2e}
```

これであなたもクラス設計者になれました。

しかし、これだけではあまりにもさびしいので、今まで体裁を調整する命令を活用します。何か目標があったほうが良いので、今回は用紙が A4 で縦位置、フォントサイズが 10 ポイント、1 行 44 文字、1 ページ 36 行のレポート用のクラスファイルにします。ページ番号はページ下部中央にアラビア数字で出力します。すると先ほどのクラスファイルは次のようになります。

```
\NeedsTeXFormat{pLaTeX2e}
\ProvidesClass{hoge}[2004/08/09 LaTeX2e class file]
\renewcommand\normalsize{\fontsize{10pt}{18pt}\selectfont}
4 \normalsize
\setlength\textwidth{44zw}
\setlength\textheight{36\baselineskip}
\parindent=1zw
\pagenumbering{arabic}
9 \pagestyle{plain}
```

まあ、何ができるか試してくださいな。フォントサイズを変更するコマンドや見出しの命令なども使えないことでしょう。ただ、通常のクラスファイルとちがって、このままでは

```
Underfull \vbox (badness 10000) has occurred while \output is active [1]
```

みたいなエラーが出ることになります。

▼ 3.1.2 版面を構成する最低限のパラメータ

さて、今回は次のようにしてクラスファイルを作成しました。

```
1 \NeedsTeXFormat{pLaTeX2e}
  \ProvidesClass{hoge}[2004/08/08 pLaTeX2e class file]
  \renewcommand\normalsize{\fontsize{10pt}{18pt}\selectfont}
  \normalsize
  \setlength{\textwidth}{40zw}
6 \setlength{\textheight}{36\baselineskip}
```

この段階で

```
Underfull \vbox (badness 10000) has occurred ....
```

なるエラーになります。まあ、原因は版面を構成するパラメータを認識できていないことになります。とりあえず、エラーを回避するだけならば `\textheight` を次のように

$$\text{\textheight} = n\text{\baselineskip} + \text{\topskip}$$

とするとうまく行きます。 n は適当に 1 ページ何行にするかを決めます。`\baselineskip` は行送りのスキップです。`\topskip` はページ上部に挿入するスキップです。とりあえず、この二つを `\textheight` に代入するとなんとかなるでしょう。

よってエラーを回避するためのクラスファイルは次のように記述します。

```
\NeedsTeXFormat{pLaTeX2e}
\ProvidesClass{hoge}[2004/08/11 pLaTeX2e class file]
\renewcommand\normalsize{\fontsize{10pt}{18pt}\selectfont}
4 \normalsize
  \setlength{\textwidth}{40zw}
  \setlength{\textheight}{35\baselineskip}
  \setlength{\topskip}{1\baselineskip}
  \addtolength{\textheight}{\topskip}
```

このようにすると 1 ページ辺り 36 行文の行送りが取られることになります。

版面を構成するパラメータは次のようになります...

▼ 3.1.3 普通の文字の大きさの定義

L^AT_EX のクラスファイルのなかで、文字の大きさを変更するための命令を作ります。基本として用意すべきなのは `\magstep` の整数倍の大きさに変更するような命令です*¹。

3.2 jsarticle

マクロ・クラス作成に関わる命令の数は膨大で、それらを一つずつ解説しては読者の皆さんも疲れるだろうから、ここは実際に奥村晴彦氏の `jsarticle` を解剖してガリッと理解しましょう。

順に見ていきましょう。最初は `\NeedsTeXFormat` と `\ProvidesClass` 命令です。

*¹ ただし、当該フォントのファミリーがどのように宣言されているかで対応も変わります。

```

\NeedsTeXFormat{pLaTeX2e}
2 \ProvidesClass{jsarticle}[2004/02/25 okumura]

```

ここでは必要とされる T_EX の種類 (pL^AT_EX 2_ε) と提供するクラスファイルの名前 (jsarticle) を指定します。

```

\newif\if@restonecol
\newif\if@titlepage
3 \newif\if@enablejfam \@enablejfamtrue

```

`\newif` 命令は新規にブール型の変数を定義します。クラスファイル (`\langle file \rangle.cls`) やマクロ (`\langle file \rangle.sty`) の中ではアット '@' は通常の英字と同じように扱われますから、`\makeatletter` や `\makeatother` といった命令は使われていません。

```
\newif\if<ブール変数>
```

次はクラスオプションの宣言をします。

```

\DeclareOption{a4paper}{%
2 \setlength\paperheight {297mm}%
  \setlength\paperwidth {210mm}}
\DeclareOption{a5paper}{%
  \setlength\paperheight {210mm}%
  \setlength\paperwidth {148mm}}

```

`\DeclareOption` 命令でそのクラスファイルが提供するオプションを宣言できます。

```
\DeclareOption{<オプション>}{<内容>}
```

{<オプション>}を宣言するときの{<内容>}は何かの定義でも構いませんし、パラメータの設定でも構いません。

フォントサイズの設定をします。

```

\DeclareOption{9pt}{\renewcommand{\@ptsize}{-1}}
\DeclareOption{10pt}{\renewcommand{\@ptsize}{0}}
\DeclareOption{11pt}{\renewcommand{\@ptsize}{1}}
4 \DeclareOption{12pt}{\renewcommand{\@ptsize}{2}}

```

`\@ptsize` というフォントサイズの基準を決める命令を再定義しています。

用紙の縦と横の長さを入れ替えるための処理をしています。

```

1 \DeclareOption{landscape}{%
  \setlength\@tempdima {\paperheight}%
  \setlength\paperheight {\paperwidth}%
  \setlength\paperwidth {\@tempdima}}

```



第 4 章

マクロ解体

4.1 L^AT_EX 標準パッケージ概説

▼ 4.1.1 indentfirst

L^AT_EX は見出しの後の字下げを `\if@afterindent` なるブール変数で制御しています。欧文のクラスファイルでは `\chapter` の後や `\section` 後は時下げしないスタイルもあります。L^AT_EX の `article`, `report`, `book` などのクラスファイルは `\chapter` や `\section` の見出し直後に字下げしないようになっています。これを簡単に字下げするようにするには「字下げしない」という命令に対して「字下げする」という命令を代入すれば良いこととなりますので、次の 2 行を書くだけになります。

```
1 \let\@afterindentfalse\@afterindenttrue
   \@afterindenttrue
```

どうです？ 簡単ですよ？ David Carlisle 氏による `indentfirst` も全く同じ記述になっています。

▼ 4.1.2 alltt

べた書き環境の `inputex` 環境がありますが、タブ記号や改行記号などを出力したいときにはちょっと不便です。この場合は Johannes Braams 氏の `alltt` パッケージが使えます。これは、`\`、`{`、`}` の三つの命令だけがそのまま使えるという `alltt` 環境を提供します。これと同じような環境を自分でも定義してみましょう（ただし、様々な状況を全く無視した簡易版です）。まず `\`、`{`、`}` 以外の文字はカテゴリーコードを変更するために `\mysanitize` 命令を定義します（`\@sanitize` 命令はバックスラッシュのカテゴリーコードも変更するので、ここでは使いません）。

```
\makeatletter
\newcommand*\mysanitize{%
3  \@makeother\$\ \@makeother\& \@makeother\# \@makeother\^%
   \@makeother\_ \@makeother\% \@makeother\~}
\makeatother
```

次に対象となる `myTT` 環境を定義します。安全のため `trivlist` 環境の中に入れます。さらに、定義の変更を局所的にするために `\begingroup` と `\endgroup` でサンドイッチします。さらに先ほどの `\mysanitize` 命令と `\ttfamily` 命令を使用します。ここで問題にな

るのは改行文字^{^^M}ですが、これはすでに`\obeylines` 命令を使うことで改行文字を改段落`\par` 命令を用います。ただし、改段落にするので`\parindent` には 0pt を代入しておきます。

```

\newenvironment{myTT}{%
  \trivlist\item\relax\begin{group}%
    \ttfamily\mysanitize%
    \obeylines\parindent=0pt}%
5  {\endgroup\endtrivlist}%

```

これで myTT 環境の定義に終了です。

```

\begin{myTT}
$ & # ^ _ % ~ \ ( x \sp{2} \neq x \sb{2} \ )
This line will be in new line.
\end{myTT}

```

さて、実際には alltt パッケージではもう少し丁寧なことをやっているの、興味のある人は見ておいてください。

▼ 4.1.3 afterpage

L^AT_EX の浮動体 (floats) 制御では満足の行く場所に、図表を配置することができないことがしばしばあります。このような場合`\clearpage` を使うことが考えられますが、ページに対する影響が多いためお得とは言いがたいでしょう。この場合とりあえず、次のページから図表を配置してほしいという命令があれば助かります。これを可能にするのが David Carlisle 氏の afterpage パッケージで、

```
1 \afterpage{\clearpage}
```

のように使います。さらにある図を現在のページが組み終わった後に出力したいというならば

```
\afterpage{\clearpage\begin{figure}[!h] ... \end{figure}}
```

のように使うことができます。

実際、この afterpage パッケージは T_EX の`\output` などと関わっているため、この冊子では深く取り扱いません。

▼ 4.1.4 xspace

用語などを統一するという用途で次のような命令を定義することがあります。

```
\newcommand*\pdfTeX{PDF\TeX}
```

これを次のように用いてしまうと、意図しない出力になります。

```
\pdfTeX is a very large system.
```

PDF^TE_X is a very large system.

このような場合は次の文字がスペースの場合は自動的に空白を挿入し、句読点や約物の場合はスペースを入れないようにしてほしいものです。実際に David Carlisle 氏の `xspace` パッケージでは次のような処理をしています。

```
\DeclareRobustCommand\xspace{\futurelet\@let@token\@xspace}
\def\@xspace{%
  \ifx\@let@token\bgroup\else
4  \ifx\@let@token\egroup\else
  ...
  \ifx\@let@token\sptoken\else
  \space
  \fi\fi ... \fi}
```

この中に登録されていないスペースを入れてほしくない文字は、自分で追加することができます。

先ほどの記述は次のようになります。

```
%\usepackage{xspace}
2 \renewcommand*\pdfTeX{PDF\TeX\xspace}
\pdfTeX is a very large system.
He may be a \pdfTeX{}nician.
```

PDF_TE_X is a very large system. He may be a PDF_TE_Xnician.

```
1 \def\@test{[\@nexttoken]}
\let\@nexttoken=\LaTeX
\@test \LaTeX
%
\futurelet\@nexttoken \@test \LaTeX
6 \def\test{\futurelet\@nexttoken \@test}
\test hoge \test gehu
```

```
% 未来の\pp{後続する}トークンがなんだか分からん時に使える。
\def\addspace{\futurelet\@next@token \@addspace}
3 \def\@addspace{(\@next@token)}
\def\tex{\TeX\addspace}
I use \tex all the time and I love \tex.\par
\def\@addspace{\ifx\@next@token.\else\space\fi}
I use \tex all the time and I love \tex.
```

▼ 4.1.5 calc

つぎのような定義をしておくと、L^AT_EX のブール値を扱いやすくなります。

```
\newcommand*\newbool[1]{%
  \expandafter\newif\csname if#1\endcsname\relax}
3 \newcommand{\TorF}[3]{\csname if#1\endcsname\relax #2\else#3\fi}
\newcommand{\ifTrue}[2]{\csname if#1\endcsname\relax #2\fi}
\newcommand{\ifFalse}[2]{\csname if#1\endcsname\else #2\fi}
\newcommand*\{True}[2]{\csname #1true\endcsname\relax}
\newcommand*\{False}[2]{\csname #1false\endcsname\relax}
8 %
\newbool{hoge}
```

```
\TorF{hoge}{真です、}{偽です。}  
\True{hoge} \ifTrue{hoge}{真ですね。}  
\False{hoge} \TorF{真だ。}{偽だ。}
```

さてさて、結果はどうなるか。

しかし、このままでは未定義のブール値も構わず処理します。

```
\newbool{geho}  
\ifFalse{gege}{あれれ}{およよ}
```

そこで、定義済みかどうかを判定する処理を付け加え、もし未定義ならば処理を中断するようにしましょう。

第 5 章

応用と実例

▼ 5.0.6 表紙の作成

さて、次のような学位論文用の表紙を作りたいとしましょう。通常のクラスファイルが提供している表代用の命令には

```
\title \author \date \maketitle
```

などが用意されています。これらと同じような処理をすれば表紙をカスタマイズできそうです。では

```
\def\date#1{\gdef\@date{#1}}  
\gdef\@date{\today}
```

hogehogehoge

5.1 問題集の作成

さて、theorem パッケージを使って、「例題型環境」や「問題型環境」を作成してみましょう。

例えば次のような仕様のマクロを作成しようとしましょう。

問題と解答を同じ場所に執筆し、何らかの切り替えで解答側の文章を別のファイル `kaito.tex` に書き出せるようにし、巻末で解答を表示する。

ここで使用例をまず示します。

```
\begin{Toi}  
\begin{mondai}  
3 問題を作成することは簡単かどうか、10文字以内で答えなさい。  
\end{mondai}  
\begin{kaito}  
非常に簡単ではない。  
\end{kaito}  
8 \end{Toi}
```

問題 5.1 問題を作成することは簡単かどうか、10文字以内で答えなさい。

ここで、別ファイルに解答を書き出すには $\text{T}_{\text{E}}\text{X}$ のファイル出力機能を使います。各問題における解答を一つの別ファイルに書き出し、原稿ファイルの末尾、いわゆる巻末にすべての解答を読み込みます。それをどのように実装するか、丁寧に解説します。

まず、「問題」は章カウンタ (chapter) と連動して問題番号を表示してほしいものです。これには theorem パッケージが流用できそうですから、今回はこのパッケージを使うことにします。今回は「問題」部分に特別な処理をしなくても良いと思いますが、今後のことも考えて環境だけは定義しておきます (このようにすると簡単に将来の拡張に対して対応できます)。

```
%\theorembodyfont{\normalfont}%プリアンプルで
2 %\newtheorem{myToi}{問}[chapter]%プリアンプルで
  \newenvironment{Toi}{\begin{myToi}}{\end{myToi}}%
    \label{toi:\c@myToi}}
  \newenvironment{mondai}{\c@myToi}
```

さて、次は肝心の kaito 環境を定義します。T_EX でファイル出力を使用するためには、まず \newwrite 命令で宣言します。

```
\newwrite{<ファイル名用の制御綴り>}
```

\newwrite には直接ファイル名をしません。あくまで制御綴りでその存在を確保するだけにします。

```
\newwrite\MyKaito
\immediate\openout\MyKaito\jobname.kai
\immediate\write\MyKaito{解答だよけど、改行とかは出力されないよ。}
\immediate\write\MyKaito{解答だよけど、改行とかは出力されないよ。}
5 \immediate\closeout\MyKaito\newpage
  \input{\jobname.kai}
```

上記が簡易な例となります。実際にタイプセットして \jobname.kai の中身を確認してください。このままでは入力原稿の改行などが反映されませんので、改行も解答ファイル \jobname.kai に出力するようにします。

theorem パッケージ等の力を借りずにきちんと自分でカウンタを使って環境を定義しましょう。

```
\documentclass{jarticle}
\makeatletter
\def\NewProblem{%
4  \newwrite\kaitofile\relax
  \immediate\openout\kaitofile\jobname.kai%
  \ifx\chapter\undefined
    \newcounter{monmon}%
  \else
9    \newcounter{monmon}[chapter]%
    \def\themonmon{\thechapter.\c@monmon}%
  \fi
}
\newenvironment{mondai}{%
14  \refstepcounter{monmon}%
  \label{mondai:\themonmon}
  \ifvmode\par\fi {\gtfamily 問~\themonmon}\quad}{\par}
\newcommand\kaito[1]{\immediate\write\kaitofile{%
  \string\par{\string\gtfamily\space 解答\space
19  \string\ref{mondai:\themonmon}}%
  \string\quad\relax#1}}
\newcommand\KAITO{\immediate\closeout\kaitofile\newpage}
```

```

\ifx\chapter\undefined \section*{解答}\else
  \chapter*{解答}\fi\input{\jobname.kai}}
24 \makeatother
   \begin{document}
   \NewProblem
   \begin{mondai}
   この問題を解け。
29 \kai{解答だよけど、改行とかは出力されないよ。}
   \end{mondai}
   \begin{mondai}
   この問題をとけ。
   \kai{解答だよけど、改行とかは出力されないよ。}
34 \end{mondai}
   \KAITO
   \end{document}

```

うーん、これだけでは少しさびしいと思われまますので、もう少し色をつけます。これでも動くのですが、「解答ファイル」には `verbatim` 環境のように、記述した内容がそのまま出力されてほしいものです。そこで、今回はこの改造を施します。

▼ 5.1.1 画像ファイルの読み込みの工夫

別に画像がないときはそのファイルを読み込んでくれなくても良いので、`\IfFileExists` 命令を使って、何か命令を定義しましょう。

```

\newcommand*\Image[1]{\IfFileExists{#1}{あるよ}{ないよ}}
\Image{hoge.eps}
\renewcommand*\Image[2][width=.8\linewidth]{\IfFileExists{#2}{%
4 \includegraphics{#1}{#2}{図版張り込み予定}}
\begin{figure}[htbp]
\begin{center}
\Image[width=.3\linewidth]{hoge.pes}
\caption{ほげほげ}
9 \end{center}
\end{figure}

```

まあ、色々応用できるでしょうから、例題程度に。

5.2 BibTeX で出力した文献一覧 .bib の著者名の重複処理

BibTeX で出力した文献一覧 $\langle file \rangle$.bib、いわゆる `thebibliography` 環境中にある `\bibitem` によるエントリを考慮します。BibTeX の文献形式 $\langle file \rangle$.bst ファイルにおいてこのような処理も実装できそうですが、 $\langle file \rangle$.bst ファイルの編集が面倒だったので、L^AT_EX 側で対応します。下記の記述を `thebibliography` 環境が始まる前に記述すると、著者名が続けて重複している場合、二つ目以降を `three em-dash` で代用します。適宜全角物の 4 倍角ダッシュにすることもできます。著者名が 5 人程度で、その人たちの文献を 10 個以上引用していると、とんでもないことになるので、応急処置的に作製しました。

```

% ----- では途切れることがあるので。
\DeclareRobustCommand\iiidemdash{%

```

```

---\kern-.5em---\kern-.5em---\kern-.5em---\kern-.5em---}
% 著者名が続けて同じならば three em-dash (\iiiemdash) で代替
5 \makeatletter
\let\orig@bibitem\bibitem
\let\temp@str\@empty
\global\def\bibitem#1#2\newblock{%
\orig@bibitem{#1}%
10 \def\temp@str{#2}
\ifx\temp@str\previous@str
\iiiemdash.\space\newblock
\else
#2\newblock
15 \fi
\def\previous@str{#2}%
}
\makeatother

```

5.3 著者名の終わりをピリオドでなくコンマにしたい

3 人以上の著者名を連記できないが BibTeX スタイル $\langle file \rangle$.bst をいじらない方法。

```

\begin{filecontents*}{hoge.bib}
2 @Book{author04,
author = {N. Author},
title = {title},
publisher = {publisher},
year = 2004
7 }
@Book{taro00,
author = {H. Taro and T. Osamu},
title = {title},
publisher = {pub},
12 year = 2000
}
\end{filecontents*}
\documentclass{jsarticle}
\bibliographystyle{jplain}
17 \makeatletter
\let\orig@bibitem\bibitem
\global\def\bibitem#1#2. \newblock{\orig@bibitem{#1} #2, \newblock}
\makeatother
\begin{document}
22 \nocite*
\bibliography{hoge}
\end{document}

```

5.4 参考文献一覧における著者名の重複

由緒正しい書籍では参考文献の著者名の重複にダッシュを使う。これは欧文組版の場合には three em-dash です。でも T_EX でこれをどうやって出力するのだろうかということ。

```

1 \DeclareRobustCommand*\iiiemdash{%
---\kern-.5em---\kern-.5em---\kern-.5em---\kern-.5em---}

```

上記のように定義して、\iiiemdash 程度に使えばよいでしょう。他にもハイフンやダッシュの位置を前後のグリフに合わせてくれる DTP ソフト（もちろん InDesign）があるようだ。明日大学に行ったらちょっとやってみよう。他にも T_EX の世界では Type tools なるものがあるようだ。

5.5 電話番号などのハイフンの位置

電話番号や郵便番号でハイフンを使うのは en-dash を使うのか、忘れてしまいました。が、とりあえず、次のようなマクロを作りました。

```

\documentclass[11pt]{jsarticle}
\makeatletter
3 % 電話番号中に出現するハイフン ‘-’ を字上げする長さ
\newdimen\raiseDim
% 初期値は エックスハイトの 20%
\raiseDim=0.2ex
% 実際にハイフンを上げるコマンド
8 \newcommand*\raise@hyphen[1]{%
% 文字列処理用に \temp@char := ‘-’ を定義
\def\temp@char{-}%
% 一文字一文字処理する
\@tfor\string@temp:=#1\do{%
13 % ある文字 \string@temp が \temp@char と等しい（ハイフンならば）
\ifx\string@temp\temp@char
% 字上げをする
\raise\the\raiseDim\hbox{\temp@char}%
% ハイフンではないならば、なにもしない
18 \else
\string@temp
\fi
}%
23 %
% 市内局番か市外局番かを判定するbool型レジスタ
\newif\if@sinai
% 市内へ切り替え
\newcommand*\sinai{\@sinaitrue}
28 % 市外へ切り替え
\newcommand*\sigai{\@sinaifalse}
% 電話番号を記述する環境 \phoneNum[#1]{#2} = #1 :=市外局番, #2 := 市内局番.
\newcommand*\phoneNum{\@ifnextchar[{\@phoneNum}{\@@phoneNum}}
% 市外局番がある場合 市外・市内を判定して出力する
33 \def\@phoneNum[#1]#2{%
\if@sinai
\raise@hyphen{#2}%
\else
(#1)~\raise@hyphen{#2}%
38 \fi
}
% 市内局番のみの場合
\def\@@phoneNum#1{\raise@hyphen{#1}}
\makeatother
43 \begin{document}
10-1111, (011)~10-3333\par
\sinai

```

```

\phoneNum{10-1111}, \phoneNum[011]{10-3333}\par
\sigai
48 \phoneNum{10-1111}, \phoneNum[011]{10-3333}
\end{document}

```

5.6 p_TE_X と j_TE_X の判別

単純に p_LA_TE_X 2_ε であるかを調べるには

```
1 \ifx\pfmtname\@undefined ?\else \pLaTeX!\fi
```

とすれば、\pfmtname が定義されていれば p_LA_TE_X 2_ε であるとします。同様に j_LA_TE_X 2_ε も

```
\ifx\jfmtname\@undefined ?\else \JLaTeX!\fi
```

として判定できます。もっと厳密に判定する時は次のようにします。

```

\documentclass{jsarticle}
\begin{document}
\makeatletter
4 \ifx\pfmtname\@undefined
  \ifx\jfmtname\@undefined
    \def\temp@str{ptex}
    \ifx\fmtname\temp@str
      \pTeX!
9    \else
      \def\temp@str{jplain}
      \ifx\fmtname\temp@str
        \JTeX!
      \else
14      \def\temp@str{plain}
      \ifx\fmtname\temp@str
        plain \TeX!
      \fi
    \fi
19  \fi
  \else
    \JLaTeXe!
  \fi
\else
24 \pLaTeXe!
\fi
\makeatother
\end{document}

```

この方法を用いた場合は p_LA_TE_X 2_ε, j_LA_TE_X 2_ε, p_TE_X, j_TE_X, plain T_EX の判定ができます。

5.7 文字列の積み重ね

```

\documentclass[11pt,papersize]{jsarticle}
\makeatletter
3 \def\stack{\@ifstar{\s@stack}{\bgroup\@stack}}

```



```

\def\s@stack{\bgroup\scriptsize\@stack}
\def\@stack#1{%
  \ensuremath{%
    \left\{%
      \vcenter{%
        \@for\member:=#1\do{%
          \hbox{\member}%
        }%
      }%
    \right\}%
  }%
\egroup}
\makeatother
\begin{document}
18
\newcommand*\va[1]{\mbox{\mbox{\$\langle$\}\itshape#1\mbox{\$\rangle$\}}
\newcommand*\Vec[1]{\ensuremath{\{#1}_1,\{#1}_2,\ldots,\{#1}_n\}}
The \verb|\stack| command's syntax:
\begin{quote}
23 \verb|\stack*|\verb|{\Vec{\va{member}}}\verb|}|
\end{quote}
Star '*' makes the members in scriptsize.

\verb|The \stack{robust,fragile}-commands| shows below,
28 \begin{center}
The \stack{robust,fragile}-commands.\par
\end{center}

\verb|The \stack*{robust,fragile}-commands| shows below,
33 \begin{center}
The \stack*{robust,fragile}-commands.\par
\end{center}

\verb|Tom loves \stack*{Alice,Becky,Catherine,Mary,Stephanie}|
38 shows below,
\begin{center}
Tom loves \stack*{Alice,Becky,Catherine,Mary,Stephanie}.
\end{center}

43 The \verb|\stack| command's source code is below.
\begin{verbatim}
\def\stack{\@ifstar{\s@stack}{\bgroup\@stack}}
\def\s@stack{\bgroup\scriptsize\@stack}
\def\@stack#1{%
48 \ensuremath{%
\left\{%
\vcenter{%
\@for\member:=#1\do{%
\hbox{\member}%
53 }%
}%
\right\}%
}%
\egroup}
58 \end{verbatim}
\end{document}

```

5.8 MoeTeX logo

```
1 \DeclareRobustCommand*\MoeTeX{M\kern-.07em\lower.5ex\hbox{0}
  \kern-.1667em\lower.5ex\hbox{E}\kern-0.1667em\TeX}
```

M_OE_{TE}X

5.9 ThorTeXTypo Wiki の口ゴ

```
\documentclass[36pt,papersize]{jsarticle}
\pagestyle{empty}
3 \usepackage[T1]{fontenc}\usepackage{pxfonts}
\begin{document}
 $w_{\kappa_i} \neq \omega_{\kappa_i}$ 
\end{document}
```

$w_{\kappa_i} \neq \omega_{\kappa_i}$

5.10 数値の基数変換

▼ 5.10.1 10 進数から 2 進数へ

```
\def\dectobin#1{%
  \@tempcnta=#1\relax \let\@bin@list=\@empty% ダミーノード
  \ifnum \@tempcnta = \z@
4   \edef\@bin@list{0}%
  \else
    \@whilenum \@tempcnta > \z@ \do{%
      \ifodd \@tempcnta
          \edef\@bin@list{1\@bin@list}%
9      \else
          \edef\@bin@list{0\@bin@list}%
      \fi
      \divide \@tempcnta \tw@ % 2 で割る
    }%
14  \fi
  \@bin@list% 変換結果
  \let\@bin@list=\@empty% リストを空にする
}
\def\hoge#1{#1 $\rightarrow$ \dectobin{#1}, }
19 \hoge{0} \hoge{1} \hoge{2} \hoge{19} \hoge{20} \hoge{43} \hoge{128}\par
```

▼ 5.10.2 2 進数から 10 進数へ

```

1 \def\bintodec#1{%
  \let \@dec@list = \@empty% ダミーノードみたいなもん
  \@tfor\@temp@char:=#1\do{\edef\@dec@list{\@temp@char\@dec@list}}%
  \@tempcnta = \@ne \@tempcntb = \z@
  \expandafter\@tfor\expandafter\@temp@char\expandafter:\expandafter=%
6  \@dec@list\do{%
  \ifnum \@temp@char = \@ne \advance \@tempcntb \@tempcnta \fi
  \multiply \@tempcnta \tw@
  }%
  \number\@tempcntb
11 \let \@dec@list = \@empty% リストを空にする
}
\def\hoge#1{#1 $\rightarrow$ \bintodec{#1}, }
\hoge{0} \hoge{1} \hoge{10} \hoge{10011} \hoge{101011} \hoge{10000000}.\par

```

▼ 5.10.3 8/16 進数から 10 進数

```

1 \def\hextodec#1{\@tempcnta=#1\relax \number\@tempcnta} %"
\def\octodec#1{\@tempcnta=#1\relax \number\@tempcnta}
\hextodec{00FF}, \octodec{177}

```

5.11 “a. a and b. a, b and c.” の自動出力

```

1
1 and 2
1, 2 and 3
1, 2, 3 and 4.
:
1, 2, 3, ..., n - 2, n - 1 and n

```

しかし、T_EX 側からすれば、総数 n がいくつあるのかとすることが分かりませんので、まずは総数 n を求めます。さらに問題となるのは、初めから i 番目というのではなく、終わりから $n - i$ 番目ということになります。ここで、もう少し分かりやすいように問題を書き換えます。

総数 \ 順番	n	$n - 1$	$n - 2$...	3	2	1
1							1
2						1 and	2
3					1,	2 and	3
4				1,	2,	3 and	4
:							
n	1,	2,	3,	...,	$n - 2,$	$n - 1$ and	n

すると最後から 1 番目は直接出力し、最後から 2 番目は ‘ $n - 1$ and’ とし、最後から 3 番目以降はカンマ付にする、ということになります。

このように考えるとカウンタが一つ必要になりますし、リストのメンバー数を数える処理が必要になります。

```

\makeatletter
2 \def\andfor#1{%
  \@tempcnta=\z% ここでカウンタをクリア
  \@for\member:=#1\do{\advance\@tempcnta\@ne}% メンバー数の計算
  \@for\member:=#1\do{% 実際の処理
    \ifnum\@tempcnta>2\relax% 3 以上ならばカンマ付
7     \member,\space
    \else
      \ifnum\@tempcnta=2\relax% 最後から二つ目ならば ‘\member~and’
        \member \nobreakspace and\space
    \else
12     \ifnum\@tempcnta=1\relax% 最後の一つなら直接出力
        \member
        \fi% \end{ifnum}
        \fi% \end{ifnum}
        \fi% \end{ifnum}
17     \advance\@tempcnta\m@ne% カウンタを一つ減少させる
    }% \end{for}
  }
\let\andfor\andfor
\makeatother

```

このようにして作成したマクロを

```

\andfor{\TeX}\par
\andfor{\TeX, \LaTeX}\par
\andfor{\TeX, \LaTeX, \LaTeXe}\par
4 \andfor{Knuth, Lamport, Carlisle, Rahtz, Sc\`opf}\par

```

とした場合の出力は

```

1 TeX
  TeX~and LaTeX
  TeX, LaTeX~and LaTeX2e
  Knuth, Lamport, Carlisle, Rahtz~and Sc\`opf

```

となります（実際はカーニングと字上げ字下げが行なわれます）。
色々と応用できそうな感じですよ、活用してみてください。

5.12 PDF アノテーション機能のメモアイコン位置

```

1 \AtBeginDvi{\special{pdf:tounicode 90ms-RKSJ-UCS2}}
  \documentclass[papersize,dvipdfm]{jsarticle}
  \usepackage[bookmarks=false]{hyperref}
  % \pdfAnnote[<見出し>]{<内容>}
  \newcommand\pdfAnnote[2][\marginpar{\mbox{}}%
6   \special{pdf: ann %width 4cm height 2.5cm %
    << /Type /Annot /Subtype /Text /Open true /C [0 0 1] %
    /T (\space#1) /Contents (#2) >>}}
  % \pdfAnnoteXY{x,y}{<内容>} [単位: cm]
  \newcommand\pdfAnnoteXY[3][\marginpar{\mbox{}}%
11  \setlength\unitlength{1truecm}%
  \begin{picture}(0,0)%

```

```

\put(#2){
  \special{pdf: ann %width 4cm height 3cm %
    << /Type /Annot /Subtype /Text /Open true %
16    /C [0 0 1] /T (\space#1) /Contents (#3) >>}}%
  \end{picture}}%
\begin{document}
  日本国憲法\pdfAnnote[憲法]{第 1 章はGHQ ではなく、日本国が独自に
  盛り込んだもの。}において第 9 条で戦争の放棄が宣言されている。 \par
21 日本国憲法\pdfAnnote{第 1 章はGHQ ではなく、日本国が独自に
  盛り込んだもの。}において第 9 条で戦争の放棄が宣言されている。 \par
  日本国憲法\pdfAnnoteXY[憲法]{0,5}{第 1 章はGHQ ではなく、日本国が独自に
  盛り込んだもの。}において第 9 条で戦争の放棄が宣言されている。 \par
  日本国憲法\pdfAnnoteXY{1,5}{第 1 章はGHQ ではなく、日本国が独自に
26 盛り込んだもの。}において第 9 条で戦争の放棄が宣言されている。
  \end{document}

```

柱に出力する方が面倒なので、傍注として出力した方が手っ取り早い気がします。

```

\AtBeginDvi{\special{pdf:tounicode 90ms-RKSJ-UCS2}}
\documentclass[11pt,papersize]{jsbook}
3 \makeatletter
  \newcounter{pdfnote}
  \newcommand*\pdfAnnote[2] []{%
    \stepcounter{pdfnote}%
    \mbox{${}^{\ast}\thepdfnote}$}%
8  \marginpar{\mbox{}}\special{pdf: ann %
    << /Type /Annot /Subtype /Text /Open false /C [0 0 1] %
    /T (*\thepdfnote\space#1) /Contents (#2) >>}}
  \makeatother
  \begin{document}
13 \tableofcontents
  \chapter{これから}
  ほげほげ\pdfAnnote[ほげほげ]{ここはよく見てね。}ほげほげ
  \chapter{それから}
  ほげほげ\pdfAnnote{げほげほ、げほげほ、げほげほ。}。
18 \chapter{どれから}
  日本国憲法第 9 条\pdfAnnote{戦争を放棄する・陸海空一切の軍事力を持たない、
  という事がかかっている、}には云々。
  \clearpage
  日本国憲法第 9 条\pdfAnnote{戦争を放棄する・陸海空一切の軍事力を持たない、
23 という事がかかっている、}には云々。
  \par
  日本国憲法第 9 条\pdfAnnote{戦争を放棄する・陸海空一切の軍事力を持たない、
  という事がかかっている、}には云々。
  \end{document}

```

忘れていましたが、pdfnote カウンタ は適当に chapter カウンタを親にした方がいいのかもしれない。

```

\makeatletter
\newcounter{pdfnote}[chapter]
3 \renewcommand\thepdfnote{\@arabic\c@pdfnote}%
  \@arabic\c@chapter.\@arabic\c@pdfnote
\newcommand*\pdfAnnote[2] []{%
  \stepcounter{pdfnote}%
  \mbox{${}^{\ast}\thepdfnote}$}%
8  \marginpar{\mbox{}}\special{pdf: ann <<

```

```

/Type /Annot /Subtype /Text /Open false /C [0 0 1] %
/T (*\@arabic\c@pdfnote\space#1) /Contents (#2) >>}}%
}
\makeatother

```

5.13 hyperref において hyperlink が nest になる例

例えば、次のようにすると目次のリンクが nest になります。

```

\documentclass[dvipdfm]{jsarticle}
\usepackage{hyperref}
3 \begin{document}
\tableofcontents
\section{図~\ref{figure}について}
\clearpage
\begin{figure}[p]
8 \caption{サンプル}\label{figure}
\end{figure}
\end{document}

```

5.14 marginpar に table を

>>28025

次の例を吟味して下さい。

```

\documentclass{journal}
\usepackage{okumacro}
\begin{document}
{\LaTeX}において傍注 (\texttt{\yen marginpar})は浮動体。
5 table 環境も figure 環境も浮動体。浮動体の
中 (\texttt{\yen marginpar})に浮動体 (table)を入れられても
困るでしょう。 \vskip2em
次のように浮動体にしない表環境を新設します。
これは傍注用に使うようにすれば良いでしょう。
10 \makeatletter
\newenvironment{margintable}%
{\vbox\bgroup\centering\def\@capttype{table}}%
{\egroup}
\makeatother
15 使い方は以下ようになります。 \vskip2em
{\LaTeX}の変遷については表~\ref{tab:hoge}\marginpar{%
\begin{margintable} \centering
\caption{ほげ\label{tab:hoge}}
\begin{tabular}{|c|}
20 \hline\LaTeX2.09\\LaTeXe\\LaTeX3\\hline
\end{tabular}
\end{margintable}}を見てね。 \vskip2em
普通の表(表~\ref{tab:ara})も次のようになります。
\begin{table}[htpb] \centering
25 \caption{普通の表\label{tab:ara}}
\begin{tabular}{|ccc|}
\hline ほげ& げほ& どれ\\hline
\end{tabular}

```

```
\end{table}
30 \vskip2em
傍注の表~\ref{tab:geho}\marginpar{%
  \begin{margintable} \centering
    \caption{げほ\label{tab:geho}}
    \begin{tabular}{|c|}
35     \hline\TeX\\\LaTeX\\何か\\\hline
    \end{tabular}
    \end{margintable}}%
はどどこに旅立つかは分かりませんがね。
\end{document}
```



第 6 章

マクロ作成の基礎知識

6.1 リスト処理

6.2 リスト処理 (簡略化バージョン)

あらかじめコントロールシーケンス `\hoge` にすでに整数値のリストが $\{a_1, a_2, \dots, a_n\}$ が格納されている事を前提とする。

```
1 \documentclass{jarticle}
  \makeatletter
  % リスト #1 が空かどうかを判定する
  \def\list@empty#1{}
  % リスト #1 の先頭に #2 を追加する
6 \def\list@push#1#2{\edef#1{#2,#1}}
  % リスト #1 の末尾に #2 を追加する
  \def\list@append#1#2{\edef#1{#1,#2}}
  % リスト #1 の先頭から要素を一つ取り除く
  \def\list@pop#1{%
11   \@tempcnta=\z@
     \let\list@save@list\@empty
     \@for\member:=#1\do{%
       \ifnum\@tempcnta=\z@
         \member
16       \else
         \ifnum\@tempcnta=\@ne
           \edef\list@save@list{\member}
         \else
           \edef\list@save@list{\list@save@list,\member}%
21       \fi
       \fi
       \advance\@tempcnta\@ne
     }%
     \edef#1{\list@save@list}%
26 }
  % リスト #1 のサイズを求める
  \def\list@size#1{%
    \@tempcnta=\z@
    \@for\member:=#1\do{\advance\@tempcnta\@ne}%
31   \number\@tempcnta
  }
  % リスト #1 を逆順に並び替える
  \def\list@reverse#1{%
    \@tempcnta=\z@
36   \let\list@save@list\@empty
```

```

\@for\member:=#1\do{%
  \ifnum\@tempcnta=\z@
    \edef\list@save@list{\member}%
  \else
41   \edef\list@save@list{\member,\list@save@list}%
  \fi
  \advance\@tempcnta\@ne
}%
\edef#1{\list@save@list}%
46 }
% リスト #1 の直和を表示する
\def\list@sum#1{%
  \@tempcnta=\z@
  \@for\member:=#1\do{%
51   \advance \@tempcnta \member
  }%
  \number\@tempcnta
}
% リスト #1 の直積を表示する
56 \def\list@prod#1{%
  \@tempcnta=\@ne
  \@for\member:=#1\do{%
    \multiply \@tempcnta \member
  }%
61  \number\@tempcnta
}
% リストの要素を表示する
\def\list@show{\@ifnextchar[{\@list@show}{\@list@show[\@empty]}}
% 書式 #1 付で リスト #2 を表示する
66 \def\@list@show[#1]#2{%
  \@tempcnta=\z@
  \@for\member:=#2\do{\advance\@tempcnta\@ne}%
  \@for\member:=#2\do{%
71   \ifnum\@tempcnta>\@ne
    \member#1
  \else
    \member
  \fi
  \advance\@tempcnta\m@ne
76 }%
}
\makeatother
\begin{document}
\makeatletter
81 \def\hoge{9,10,5,7,10,100}
\list@show[,]{\hoge}\par
\list@append{\hoge}{4}
\list@show[,]{\hoge}\par
\list@push{\hoge}{8}
86 \list@push{\hoge}{27}
\list@show[,]{\hoge}\par
ポップ : \list@pop{\hoge}\par
\list@show[,]{\hoge}\par
逆順 : \list@reverse{\hoge}
91 \list@show[,]{\hoge}\par
直和 : \list@sum{\hoge}\par
直積 : \list@prod{\hoge}\par

```

```
\makeatother
\end{document}
```

例えば、あるマクロ `\List` の先頭・末尾にメンバーを追加したい、という場合は

```
\let\List\empty% リスト処理のためのダミーノードに相当する
\def\appendList#1{\def\List{\List#1}}
\def\pushList#1{\def\List{#1\List}}
\appendList{hoge}
5 \appendList{,foo}
\appendList{,bar}
\pushList{hoge,}
\List
```

というのは無限ループでエラーになる。`\List` というマクロを定義するために自分自身を参照しているためである。ここで `\edef` という定義中のマクロを展開するプリミティブが存在する。これを使えば次のように記述できる。

```
\let\List\empty% リスト処理のためのダミーノードに相当する
2 \def\appendList#1{\edef\List{\List#1}}
\def\pushList#1{\edef\List{#1\List}}
\appendList{hoge}
\appendList{,foo}
\appendList{,bar}
7 \pushList{hoge,}
\List
```

`\List` の出力は `hoge,hoge,foo,bar` となっていることから、`\List` を再定義する前に `\edef` は前回の `\List` を展開していることが分かる。

▼ 6.2.1 文字処理

▼ 6.2.2 文字列処理

6.3 展開と置換

▼ 6.3.1 展開順序

▼ 6.3.2 展開の抑制

▼ 6.3.3 閑話休題

6.4 グループ핑

▼ 6.4.1 局所

▼ 6.4.2 大域

▼ 6.4.3 グループの区切り

▼ 6.4.4 入れ子

▼ 6.4.5 波括弧の重み

6.5 モード

垂直モード ページを組むとき等。

内部垂直モード `\vbox` などの中。

水平モード 途中改行を許して行を組む。

限定水平モード `\hbox` などの中で途中改行を許さない。

数式モード 行中に組む数式。

ディスプレイ数式モード 別行に組む数式。

法則を以下にまとめる。

1. 垂直モードのとき、単なる文字列を見つけると水平モードに移行する。
2. 垂直モードのとき、特殊な命令によって水平モードに移行する（たとえば `\indent` や `\leavevmode` など）。
3. 垂直モードのとき、`\hbox` を見つけると限定水平モードに移行する。
4. ...

▼ 6.5.1 水平

▼ 6.5.2 垂直

▼ 6.5.3 内部/限定

▼ 6.5.4 ボックスの内容

```
\ifvoid(数値) (なんにもないのか)
\ifhbox(数値) (水平ボックスを含むのか)
\ifvbox(数値) (垂直ボックスを含むのか)
```

▼ 6.5.5 モードのあれこれ

```
\ifvmode (垂直モードの判定)
\ifhmode (水平モードの判定)
\ifmmode (数式モードの判定)
\ifinner (内部モードの判定)
```

この `\voidb@x` を使って垂直モードから水平モードに移行するコマンド `\leavevmode` が作られます。

```
\def\leavevmode{\unhbox\voidb@x}
```

`\strutbox` が次のように定義されています。数式モードでも使われる。

```
\newbox\strutbox
\def\strut{\relax\ifmmode\copy\strutbox\else\unhcopy\strutbox\fi}
```

▼ 6.5.6 `\framebox` 実装への道 その 1

さてさて、L^AT_EX では `\framebox` などに見られるような、LR 要素の四方を線で囲み、枠を付けるコマンド `\framebox/\fbox` が用意されています。これを T_EX のマクロのみで実装する方法を考えましょう。まずは考え方です。次の図を御覧ください。二通りのアプローチがあります。

$$\begin{array}{l} \vbox \left\{ \begin{array}{c} \hrule \\ \hbox{| \vbox{文章要素} |} \\ \hrule \end{array} \right\} \\ \hbox \left\{ \begin{array}{c} \hrule \\ \hbox{文章要素} \\ \hrule \end{array} \right\} \end{array}$$

さて、このような考え方に基づくと、次のような構成になります。

```
\vbox{%
  \hrule%
  \hbox{\vrule\hbox{\pa{文章要素}}\vrule}%
}
```

```
\hrule%
}
```

ここで、文章要素と枠とを離れさせるための間隔 `\myboxsep` を導入します。初期値は適当に 3pt ということにしておきます。

```
\newlength{\myboxsep}
\setlength\myboxsep{3pt}
```

このようにして、

```
\ifvmode \leavevmode \fi\ vbox{%
  \hrule
3  \par\vskip \myboxsep
  \hbox{\vrule \hskip \myboxsep \pa{文章要素}\hskip \myboxsep \vrule}%
  \par\vskip \myboxsep
  \hrule
}
```

縦の罫線を先に引く方法ではちょっと考え辛いので、横の罫線を先に引く場合を考えてみます。

```
\ifvmode \leavevmode \fi
\hbox{%
3  \vrule
  \vbox{%
    \hrule
    \par \vskip \myboxsep
    \hbox{\hskip \myboxsep\cmd{\hbox}{\pa{文章要素}\hskip \myboxsep}}%
8  \par \vskip \myboxsep
    \hrule
  }%
  \vrule
}
```

上記のようにすると、見事に `\myboxsep` が生きています。さらに、続いて枠の太さを決める `\myboxrule` も導入します。

```
\newlength{\myboxrule}
\setlength\myboxrule{3pt}
```

初期値は適当に 3pt にでもしておきます。すると次のように `\myframebox` が定義できます。

```
\def\myframebox#1{\ifvmode \leavevmode \fi
\hbox{%
3  \vrule width \myboxrule
  \vbox{%
    \hrule height \myboxrule
    \par \vskip \myboxsep
    \hbox{\hskip \myboxsep #1\hskip \myboxsep}%
8  \par \vskip \myboxsep
    \hrule height \myboxrule
  }%
  \vrule width \myboxrule
}%
13 }
```

単に `\vrule` には幅 (width) を、`\hrule` には高さ (height) を指定するだけです。また、次のようにして枠がびったりとくつつく `\myfbox` も作ることが出来ます。

```

\def\myfbox#1{%
2 \begingroup
  \setlength\myboxsep{0pt}%
  \myframebox{#1}%
\endgroup
}

```

使用例は次のようになります。

```

\myframebox{文章要素}, \myfbox{文章要素}, \myframebox{文章要素}\par
\myboxrule=.4pt \myfbox{文章要素}

```

さて、本来 `\framebox` は

```
\framebox[⟨幅⟩][⟨位置指定子⟩]{⟨文章要素⟩}
```

のような使い方が出来るのでまだ不十分ですが、今日はこのへんで終わりにしましょう (眠いのもう寝ます)。

6.6 カウンタ

- ▼ 6.6.1 T_EX のカウンタ
- ▼ 6.6.2 L^AT_EX のカウンタ
- ▼ 6.6.3 親子カウンタ
- ▼ 6.6.4 相互参照

6.7 ボックス・グルー

▼ 6.7.1 グルー

▼ 6.7.2 罫線

`\hrule` は垂直モード、`\vrule` は水平モードで組まれる事になります。水平モードで `\hrule` を使うと、その時点で水平モードを終了し垂直モードに移行します。

水平線を文中に引こうと `\verb|\hrule|` を使おうとしたら `\hrule` 全然うまく逝きません。逆に `\verb|\vrule|` の方が `{\vrule width 10pt}` 良いです。なぜですか？

水平線を文中に引こうと `\hrule` を使おうとしたら全然うまく逝きません。逆に `\vrule` の方が ■ 良いです。なぜですか？

▼ 6.7.3 リータ

```
% 垂直モードで \hrulefill が呼び出されたときのために \leavevmode
% \E{tabular}/\E{array} 環境で使われたときのために \kern\z@
3 \def\hrulefill{\leavevmode\leaders\hrule\hfill\kern\z@}
\def\dotfill{\leavevmode\cleaders\hb@xt@.44em{\hss.\hss}\hfill\kern\z@}
```

▼ 6.7.4 ボックスの仕組み

```
\hbox{(要素)} (限定水平モード)
\vbox{(要素)} (内部垂直モード)
```

`\hbox` は限定水平モードなので (要素) を水平方向に並べようとしますが途中で改行できません。`\vbox` は垂直モードですから、(要素) を垂直方向に並べようとします。

<code>\hbox{\hbox{ほげ}\hbox{ほげ}\hbox{ほげ}}</code>	ほげほげほげ
<code>\vbox{\hbox{ほげ}\hbox{ほげ}\hbox{ほげ}}</code>	ほげ
<code>\vbox{\leavevmode\hbox{ほげ}\hbox{ほげ}}%</code>	ほげ
<code>\hbox{ほげ}}</code>	ほげほげほげ

<code>\newcommand{\tete}{\hbox{ほげ}}</code>	ほげ
<code>\hbox{ほげ}\hbox{ほげ}}</code>	ほげ ほげ
<code>\ldots\vbox{\hrule\tete\hrule}</code>	... ほげ ... ほげ ... <u>ほげ</u> ...
<code>\ldots\$\vcenter{\hrule\tete\hrule}\$</code>	ほげ ほげ
<code>\ldots\vtop{\hrule\tete\hrule}\ldots</code>	ほげ

このことから L^AT_EX の `\fbox` 命令は

```
1 \vbox{ \hrule
      \hbox{\vrule\relax{これ}\vrule}%
      \hrule}%
```

と同じようなことを行っていると考えられるでしょう。

```
{\fboxsep=0pt \fbox{こちら}}は多分 \TeX の
\verb|\hrule|, \verb|\vrule|, \verb|\vbox|,
\verb|\hbox|を次のようにしていると考えられ
ます。さっきのは
\verb|\hrule \hbox{\vrule \relax こちら %
\vrule}\hrule}
```

こちらは多分 T_EX の \hrule, \vrule, \vbox, \hbox を次のようにしていると考えられます。さっきのは
こちら

しかし少し考えてみれば L^AT_EX では \fbox を作成するときに \fboxsep と \fboxrule によって罫線の太さと文字列との間隔を調整できます。もう少し工夫した定義が必要なのはお分かりだと思います。

6.8 ファイル

▼ 6.8.1 補助ファイルの役割

▼ 6.8.2 ファイル入力

```
\newread{<レジスタ名>}
```

▼ 6.8.3 ファイル出力

```
\newwrite{<レジスタ名>}
```

▼ 6.8.4 ファイルに関わるその他のコマンド

あるファイル *<file>.tex* をタイプセットするときに、ページを組む前に *<file>.aux* を読み込まなければなりませんので \document 命令で *<file>.aux* を読み込むように定義されています。

```
\document (本文の始めに一度だけ記述する)
\nofiles (ファイルの書き出しをすべて無効にする)
```

書籍等を作成していて、もうそれ以上ファイルに変更を加えないことが分かっているときに、目次ファイル *<file>.toc* などを固定するとき等に \nofiles が使えます。

現在タイプセットしているファイルで使用されているファイル（プリロードも含む）を調べるには \listfiles 命令を使います。

L^AT_EX 処理を実行した原稿のプリアンブルに

```
\listfiles
```

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

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

のようなファイル `\filename.tex` が存在し、`platex` などタイプセットすれば

```

This is pTeX Version 3.141592-p3.1.2 (sjis)(Web2C 7.5.2)
*File List*
  pldefs.ltx 2000/07/13 v1.2 pLaTeX Kernel
  jy1mc.fd 1997/01/24 v1.3 KANJI font defines
  jy1gt.fd 1997/01/24 v1.3 KANJI font defines
  kinsoku.tex
  plpatch.ltx
  jbook.cls 2001/10/04 v1.3 Standard pLaTeX class
  jbk10.clo 2001/10/04 v1.3 Standard pLaTeX file
*****
Output written on filename.dvi (1 page, 216 bytes).
Transcript written on filename.log.

```

のような表示が出るでしょう。ここで少し疑問に思っていたきたいことは、「原稿には `jbook` を使うことしか宣言していないのに何か別のファイルも一緒に読み込まれている。」ということです。このように L^AT_EX プログラムを実行した段階ですでに読み込まれているファイルを *preload file* と呼びます。

一つの配布ファイル `hoge.tex` から、別のファイルに出力したいときは `\filecontents` 命令を使います。コメントをつけてほしくないときは `\filecontents*` を使います。

```

\begin{filecontents}{hoge.txt}
私はこれから山に芝刈りに行きます。
3 \end{filecontents}
\documentclass{jsarticle}
\begin{document}
\section{芝刈り}
\input{hoge}
8 \section{2度目の芝刈り}
\input{hoge}
\end{document}

```

```

\IfFileExists{<ファイル名>}{<真の時>}{<偽の時>}
\InputIfFileExists{<ファイル名>}{<真の時>}{<偽の時>}

```

6.9 エラー

▼ 6.9.1 T_EX のエラー

▼ 6.9.2 L^AT_EX のエラー

▼ 6.9.3 T_EX の警告

▼ 6.9.4 L^AT_EX の警告

▼ 6.9.5 エラーの出し方

```

\GenericInfo{<>}{<ログファイルへ書き出す内容>}

```

```

\makeatletter
% \jobname.log にしか書き出さない
\GenericInfo{hoge:}{%

```

```

だめだめだめ!\MessageBreak
5 買うな!\MessageBreak
   あんたにはまだ早い!}

あふお。 \par
% \jobname.log に書き出しターミナルにも表示する
10 \GenericWarning{geho:}{だめだぞおそんなことしたらあ、先生
   こまっちゃうなあ、どうしてもお？ あはははは。どーんな問題。}

```

`\PackageWarningpackage{<警告>}` 入力ファイルの行数を表示しない `NoLine`
`\PackageWarningNoLine{<パッケージ>}{<警告>}` `\PackageError{<パッケージ>}{<エラー内容>}{<ヘルプ>}` `\PackageInfo{<パッケージ>}{<情報>}`

`\ClassWarning` `\ClassWarningNoLine` `\ClassError` `\ClassInfo`
`\@latex@error{<エラー内容>}{<ヘルプ>}` `\@latex@warning{<警告>}` `\@latex@warning@no@line@`
`警告}` `\@latex@info{<情報>}` `\@latex@info@no@line{<情報>}`

時折マクロ・クラスの中で `\@warning` などを見掛けますが、これは `\@latex@warning` と同じものです。

```

\let\@warning\@latex@warning
\let\@warning\@latex@warning@no@line
\global\let\@latexerr\@latex@error

```

▼ 6.9.6 脆弱・頑丈

6.10 フォント

- ▼ 6.10.1 NFSS
- ▼ 6.10.2 基本属性
- ▼ 6.10.3 数式モードでのフォント
- ▼ 6.10.4 フォントの定義

6.11 クラスファイルの構築

- ▼ 6.11.1 もっとも短いクラス
- ▼ 6.11.2 ちょっと長いクラス
- ▼ 6.11.3 さらにちょっと長いクラス
- ▼ 6.11.4 版面
- ▼ 6.11.5 柱・ノンブル
- ▼ 6.11.6 表題

▼ 6.11.7 見出し

章節見出し

見出し番号

見出しのカスタマイズ

```
\@startsection{<カウンタ名>}{<深さ>}{<字下げ幅>}{<前空き>}{<後空き>}{<体裁>}
```

最後の〈体裁〉には書体変更・サイズ変更用のコマンド (`\reset@font`, `\Large`, `\sffamily`, `\MakeUppercase`) や揃えを行なうコマンド (`\centering`, `\raggedright`, `\raggedleft`) が使えます。〈後空き〉が 0 より小さければ「文中見出し」として `\paragraph` などのように見出し語の直後に改行できないようにします。

通常は次のような `\hoge` 命令を定義するには `\@ifstar` と `\@ifnextchar` を使うことになります。

```
\documentclass[a4j,11pt,papersize]{jsarticle}
2 \makeatletter
\def\hoge{\@ifstar {\@hoge}{\@hoge}}
\def\@hoge{\@ifnextchar [{\@hoge}{\@hoge[\@empty]}}
\def\@hoge[#1]#2{任意引数は‘#1’, 必須引数は‘#2’}
\makeatother
7 \begin{document}
\hoge{ひ}\par
\hoge*{ひ}\par
\hoge[に]{ひ}\par
\hoge*[に]{ひ}\par
12 \end{document}
```

しかし、毎回これをやっていると付かれるので見出しに関しては `\secdef` なるコマンドが用意されています。また、星付き `*` の場合は目次には書き出さないという暗黙の了解があるので、

```
\hoge*[に]{ひ}
```

というパターンは許容しません。ですから

```
\documentclass[a4j,11pt,papersize]{jsarticle}
\makeatletter
\def\hoge{\secdef {\@hoge}{\star@hoge}}
4 % 任意引数が与えられていないときは自動的に 必須引数が渡される
% \@ifnextchar [{\@hoge}{\one@hoge}
% \def\one@hoge#1{\@hoge[#1]#2}
% と同じこと。
\def\@hoge[#1]#2{任意引数は‘#1’, 必須引数は‘#2’}
9 \def\star@hoge#1{星付で必須引数は‘#1’}
\makeatother
\begin{document}
\hoge{目次にも出力される見出し}\par
\hoge*{目次には出力されない見出し}\par
14 \hoge[目次用のテキスト]{見出し用の長いやつ}\par
\end{document}
```

という例があれば、

任意引数は‘目次にも出力される見出し’, 必須引数は‘目次にも出力される見出し’ 星付で必須引数は‘目次には出力されない見出し’ 任意引数は‘目次用のテキスト’, 必須引数は‘見出し用の長いやつ’

という出力になります。

▼ 6.11.8 図表

▼ 6.11.9 目次

目次の機構

```
\addtocontents{<拡張子>}{<追加する内容>}
\addcontentsline{<拡張子>}{<種類>}{<追加する内容>}
```

から

```
\@writefile{toc}{\contentsline{chapter}{\numberline{第 1章}{\TeX}の基礎}{1}}
```

が *<file>.aux* に書き出されて、これから *<file>.toc* に

```
\contentsline {chapter} {\numberline{第 1章}{\TeX}の基礎} {1}
```

借りに 次のような *<file>.tex* を用意して 2 回程タイプセットを行なえば

```
\documentclass[a4j,11pt,papersize]{jsarticle}
\begin{document}
\tableofcontents
4 \listoffigures
\listoftables
\section{ほげ}
\subsection{どれ}
\subsubsection{ほれ}
9 \paragraph{ありゃ}
\subparagraph{こりゃ}
\begin{table}[htbp]
\caption{ほげ}
\end{table}
14 \begin{figure}[htbp]
\caption{どれ}
\end{figure}
\end{document}
```

それぞれ *<file>.toc* では

```
\contentsline {section}{\numberline {1}ほげ}{1}
\contentsline {subsection}{\numberline {1.1}どれ}{1}
3 \contentsline {subsubsection}{\numberline {1.1.1}ほれ}{1}
\contentsline {paragraph}{ありゃ}{1}
\contentsline {subparagraph}{こりゃ}{1}
```

さらに *<file>.lof* では

```
\contentsline {figure}{\numberline {1}{\ignorespaces どれ}}{1}
```

ついでに `<file>.lot` では

```
\contentsline {table}{\numberline {1}{\ignorespaces ほげ}}{1}
```

極めつけに `<file>.aux` では

```
\relax
\@writefile{toc}{\contentsline{section}{\numberline
  {1}ほげ}{1}}
4 \@writefile{toc}{\contentsline{subsection}{\numberline
  {1.1}どれ}{1}}
\@writefile{toc}{\contentsline{subsubsection}{\numberline
  {1.1.1}ほれ}{1}}
\@writefile{toc}{\contentsline{paragraph}{ありゃ}{1}}
9 \@writefile{lot}{\contentsline{table}{\numberline
  {1}{\ignorespaces ほげ}}{1}}
\@writefile{lof}{\contentsline{figure}{\numberline
  {1}{\ignorespaces どれ}}{1}}
\@writefile{toc}{\contentsline{subparagraph}{こりゃ}{1}}
```

ある範囲の見出しを目次に出したくない `\addcontentsline` と `\addtocontents` を無効化すれば良いので

```
\documentclass{book}
2 \begin{document}
\tableofcontents
\begingroup % ここから見出しを目次に出したくない
\def\addcontentsline#1#2#3{% \@gobble \@gobbletwo
\def\addtocontents#1#2{% \let\addcontents \@gobbletwo
7 \chapter{ほげ}
ほげほげほげ。
\section{ほえ}
ほえほえほえ。
\endgroup% ここまで見出しを目次に出したくない
12 \chapter{うりゃ}
うりゃうりゃうりゃ。
\end{document}
```

目次の体裁

```
1 \@dottedtocline{<>}{<>}{<>}{<>}{<>}
\@pnumwidth
\@tocrmarg
\@dotsep
\l@<なんとか>
```


▼ 6.11.10 文献一覧・索引・用語集

6.12 空白

▼ 6.12.1 自動で入る空白

▼ 6.12.2 手動で入れる空白

```
\nopagebreak[⟨0-4⟩] (改ページを抑制する)
\pagebreak[⟨0-4⟩] (改ページを助長する)
\nolinebreak[⟨0-4⟩] (改行を抑制する)
\linebreak[⟨0-4⟩] (改行を助長する)
```

改行を許さないスペースにチルダ`~`があります、これと同じコマンドが`\nobreakspace`として使用できます。

```
\DeclareRobustCommand{\nobreakspace}{%
  \leavevmode\nobreak\ }
\catcode'\~ =13
\def~{\nobreakspace{}}
5 \expandafter\let\expandafter\@xobeysp\cename nobreakspace \endcsname
```

`\@xobeysp` は `\obeyspaces` などにより空白文字を無視せずに一つのスペースとして扱うときにそれを `\nobreakspace` にするための設定です。

▼ 6.12.3 スペースファクタ

6.13 出力ルーチン・ページの構成法

▼ 6.13.1 出力

▼ 6.13.2 マークの基本

▼ 6.13.3 output

6.14 ページ構成

▼ 6.14.1 パラメータ

```
\pretolerance=100
\tolerance=200
\hbadness=1000
\vbadness=1000
5 \linepenalty=10
\hyphenpenalty=50
\exhyphenpenalty=50
\binoppenalty=500
\clubpenalty=150
10 \windowpenalty=150
\displaywindowpenalty=50
```

```

\brokenpenalty=100
\predisplaypenalty=10000
%\postdisplaypenalty=0
15 %\interlinepenalty=0
%\floatingpenalty=0
%\outputpenalty=0
\doublehyphendemerites=10000
\finalhyphendemerites=5000
20 \adjdemerits=10000
%\looseness=0% , cleared by \TeX after each paragraph
%\pausing=0
%\holdinginserts=0
%\tracingonline=0
25 %\tracingmacros=0
%\tracingstats=0
%\tracingparagraphs=0
%\tracingpages=0
%\tracingoutput=0
30 \tracinglostchars=1
%\tracingcommands=0
%\tracingrestores=0
%\language=0
\uchyph=1

```

```

\showoutput<整数>
\tracingall
\tracingcommands<整数>
\tracingstats<整数>
\tracingpages<整数>
\tracinglostchars<整数>
\tracingmacros<整数>
\tracingparagraphs<整数>
\tracingrestores<整数>

```

`\tracingall` は全ての処理状況を表示するためのマクロ。

```

1 \gdef\tracingall{%
  \tracingcommands\tw@
  \tracingstats\tw@
  \tracingpages\@ne
  \tracinglostchars\@ne
6  \tracingmacros\tw@
  \tracingparagraphs\@ne
  \tracingrestores\@ne
  \errorcontextlines \maxdimen
  \showoutput}
11 \def\tracingall{\@autoerr\tracingall}

```

エラー出力 プリアンブルに

```
\setcounter{errorcontextlines}{1}
```

によってエラーの出力される量を調整することができます。標準は `-1` です。

```

\parindent=20pt
\hangindent=0pt
\hoffset=0pt
4 \voffset=0pt
\parskip=0pt plus 1pt
\adovedisplayskip=12pt plus 3pt minus 9pt
\adovedisplaysshortskip=0pt plus 3pt
\belowdisplayskip=12pt plus 3pt minus 9pt
9 \belowdisplaysshortskip=7pt plus 3pt minus 4pt
%\leftskip=0pt
%\rightskip=0pt
\topskip=10pt
\splittopskip=10pt
14 %\tabskip=0pt
%\spaceskip=0pt
%\xspaceskip=0pt
\parfillskip=0pt plus 1fil

```

```

\newskip\normalbaselineskip \normalbaselineskip=12pt
\newskip\normallineskip \normallineskip=1pt
3 \newdimen\normallineskiplimit \normallineskiplimit=0pt

```

`\normalbaselines` (通常に行送りに戻す)

```

2 \def\normalbaselines{%
  \lineskip=\normallineskip
  \baselineskip=\normalbaselineskip
  \lineskiplimit=\normallineskiplimit
}

```

`\frenchspacing` (フランス語風にスペーシング)

`\nonfrenchspacing` (通常のスペースファクタでスペーシング)

▼ 6.14.2 段落処理

`\rightskip` と `\leftskip` を使うと、ごによごによでできる。ただし、段落が終了した時点で設定が反映されるので、段落の終了を明示的に示す必要もあります。

```

{\ifvmode\leavevmode\fi
\leftskip=3zw \rightskip=3zw
あいうえお、かきくけこ\par
}

```

波括弧は `\leftskip` や `\rightskip` などの大域変数の設定が外側の部分にも影響しないように、有効範囲 (スコープ) を決めています。

インデントを操作する命令として `\hangindent` と `\hangafter` の二つがあります。

`\hangindent` (字下げの幅)


`\hangafter` (どの行から字下げするか決める)

`\parshape = n i1 l1 i2 l2 ⋯ in ln`

```

1 \bgroup\rightskip=3zw \leftskip=\rightskip
  段落を終了した時点で、値が反映される。
  段落の終了を明示的に示さないため。 \par\egroup
  \newenvironment{myquote}[2]%
    {\ifvmode\par\fi\bgroup\leftskip#1\relax\rightskip#2}%
6   {\par\egroup}
   \begin{myquote}
     段落を終了した時点で、値が反映される。
     段落の終了を明示的に示さないため。
   \end{myquote}

```

 それはちょっとやめたほうが良いんじゃないか？だってねえ、これから逝くんでしょ
う？やっぱりやめたほうがいいよなあ。ああ。それはちょっとやめたほうが良いん
じゃないか？だってねえ、これから逝くんでしょ？やっぱりやめたほうがいいよなあ。
ああ。それはちょっとやめたほうが良いんじゃないか？だってねえ、これから逝くんで
しょ？やっぱりやめたほうがいいよなあ。ああ。

▼ 6.14.3 スペースファクタ

▷ 例題 6.1 T_EX では `\TeX` 命令は T_EX と定義されていますが、L^AT_EX 2_ε では T_EX と定義されているのは何故でしょうか。 `\@` はスペースファクタである事を考慮してください。

▼ 6.14.4 ハイフネーション

▼ 6.14.5 ハイフネーション

T_EX のハイフネーションを実現するハイフネーションアルゴリズムはそもそも欧文のハイフネーションはその単語の品詞によって違うことが問題となります。名詞では「贈り物」を意味する ‘present’ は ‘pres-ent’ になりますが、これが「贈呈する」の動詞になると、‘pre-sent’ になり、アクセントも変わります。もし、誤ったハイフネーションを組版してしまった場合は、文の構造までもが変わってしまうので、ハイフネーション一つをとっても難しい問題です。

このような問題に対しては、最低でも構文解析等が必要になり、より完成度を高める場合は意味解析なども要求されます。しかし、それらの処理を実装することは非常に難しいことで、いわゆる自然言語処理の領域に手を染めなければなりません。

T_EX ではもう少し簡易なアルゴリズムでハイフネーションを行っています。簡易といってもそれは Frank Liang 氏によって発見された優秀なアルゴリズムで、精度が 89.3% と極めて高くなっています。これは実際の単語の使用頻度を考えると、正確性は 95% を超えるとも T_EX の作成者の著書で述べられています。

T_EX はハイフネーションをその辞書の中から検索し、パターンが一致したものを採用します。そのため、新語などにもある程度対応します。もし T_EX に登録されていない単語でも `\hyphenation` コマンドで大域的に指定することができます。

```
\hyphenation{〈単語, …〉}
\fussy
\sloppy
```

```
\hyphenation{ho-ge pi-yo ge-ma a-rya}
{\fussy hoge hoge hoge hoge
 hoge hogehogehogehogehoge hoge.}\par
{\sloppy hoge hoge hoge hoge hoge
 hogehogehogehogehoge hoge.}\par
hoge hogehogehogehogehoge hoge.
```

```
hoge hoge hoge hoge hogehogehogehogehogehoge
hoge.
hoge hoge hoge hoge hoge hogehogehogehogehoge
hoge.
hoge hogehogehogehogehoge hoge.
```

▼ 6.14.6 行分割

▼ 6.14.7 ページ分割

▼ 6.14.8 ブレークポイント



第7章 L^AT_EX 2_ε 解体

7.1 予備知識

L^AT_EX 2_ε のマクロを眺めたり手を出す前に知っておくと便利なものがあります。

```
\def\fmtname{LaTeX2e}
\def\fmtversion{2001/06/01}
```

```
\typeout{<ターミナルに警告として表示したい内容>}
\typein[<制御綴>]{<ターミナルに警告として表示したい内容>}
```

```
\newcount\hoge
\typeout{ちょっとお尋ねしたいことがあります。}
3 私の名前は「\typein{あなたの名前はなんですか?}」です。
私は\typein[\tempstr]{あなたの年齢は?} \tempstr 歳になります。
\hoge=\tempstr \advance\hoge 1% インクリメント
来年は \the\hoge 歳になります。 \tempstr
```

```
「
  ちょっとお尋ねしたいことがあります。
  あなたの名前はなんですか?
```

```
\@typein=渡辺徹
あなたの年齢は?
```

```
\tempstr=21
」
```

記号やマクロの保存

とある表紙で何らかの記号をアクティブにしてしまうと、あっちの人になって元の世界に帰って来れなくなるので、ある程度の記号に関してはバックスラッシュと普通の文字で別の参照手段を残しておきます。

```
\def\^^M{\ } % これは改行文字をスペースとして定義する
\let\^^I\^^M % これはタブ文字をスペースとする
3 \def\lq{'}% 左シングルクオート
\def\rq{'}% 右シングルクオート
\def\lblack{[]}% 左角括弧
\def\rblack{[]}% 右角括弧
\let\endgraf=\par% plain TeX の実改段落 \par を保存
8 \let\endline=\cr% plain TeX の実改行 \cr を保存
\def\space{ }% スペースを \space として定義することで展開抑制を狙う
\let\empty\@empty%
```

```

\def\null{\hbox{}}% ページ先頭などで使うことも
\let\bgroup={
13 \let\egroup=}

```

`\@empty` は次のように定義されているものとします。`\space` のほかに `\@spaces` も定義されています。

```

\def\@empty{}
2 \def\@spaces{\space\space\space\space}% 空白四つ

```

```

\def\@height{height}
\def\@depth{depth}
3 \def\@width{width}
\def\@minus{minus}
\def\@plus{plus}

```

```

\def\hbx@t@{\hbox to}

```

TEX プリミティブの保存。

```

\let\@par=\par
\let\@hyph=\-
\let\@dischph=\-
4 \let\@italiccorr=\/

```

```

\obeylines
\obeyspaces

```

`\bgroup`

連続する空白 改行は普通は一つのスペースとして扱われる。`\par`

`\obeylines \obeyspaces%`

連続する空白 改行は普通は一つのスペースとして扱われる。

`\egroup`

連続する空白 改行は普通は一つのスペースとして扱われる。

連続する空白 改行は普通は一つのスペースとして扱われる。

```

\slash

```

改行できるか出来ないか。

```

\break (分割を促す)
\nobreak (分割を抑制する)
\allowbreak (分割を許す)
\smallbreak (-50)
\medbreak (-100)
\bigbreak (-200)

```

`\m@th` は良く見掛けるマクロ。

```

1 \def\m@th{\mathsurround\z@}

```


引数

```

\long\def\@gobble#1{}
\long\def\@gobbletwo#1#2{}
\long\def\@gobblefour#1#2#3#4{}
4 \long\def\@firstofone#1{#1}
\long\def\@firstoftwo#1#2{#1}
\long\def\@secondoftwo#1#2{#2}
\let\@iden \@firstofone
\long\def\@thirdofthree#1#2#3{#3}
9 \long\def\@expandtwoargs#1#2#3{%
  \edef\reserved@a{\noexpand#1{#2}{#}}\reserved@a}
% はてはて
\edef\@backslashchar{\expandafter\@gobble\string\}

```

```

\@gobble
\@gobbletwo

```

テキスト用コマンド

例えば `\texttrademark` で ™ を出力するように、添え字は何かと文中でも使うものです。この場合は `\textsuperscript` を使うとうまく行きます。

```
\let\tsp\textsuperscript
```

詳細は参考文献 `\tsp{(1,3)}` を参照してください。

詳細は参考文献 ^(1,3) を参照してください。

何らかの原因で合字（エンコーディングの問題などで）が使えないときは、それぞれ次のコマンドで代替することができます。

通常の命令	代替命令	合字
---	<code>\textemdash</code>	—
--	<code>\textendash</code>	–
!‘	<code>\textexclamdown</code>	¡
?‘	<code>\textquestiondown</code>	¿
“	<code>\textquotedblleft</code>	“
”	<code>\textquotedblright</code>	”
‘	<code>\textquoteleft</code>	‘
’	<code>\textquoteright</code>	’

たまにドイツ語などで合字を殺したいときは、`\textcompwordmark` 命令を使うこともできます。これはハイフネーションされません。ハイフネーションされても良い場合は `\-` を使うと良いでしょう。

```
shelfful, shelf\textcompwordmark ful, shelf\-\ful\shalf, shelfful, shelfful
```

7.2 リスト系環境

リスト系環境を理解するには T_EX における段落処理を理解する必要があります。まずは次の三つの段落制御用のコマンドを見てください。

改段落 `\par` はリスト系の環境や `array/tabular` 環境などでは適宜都合の良いように変更されます。

```
\def\@setpar#1{\def\par{#1}\def\@par{#1}}
\def\@par{\let\par\@par\par}
3 \def\@restorepar{\def\par{\@par}}
```

T_EX オリジナルの `\par` は `\@par` に保存されています。`\@setpar` によって `\par` と `\@par` の定義内容を変更します。`\@par` には予めオリジナルの `\@par` を定義しておき、元に戻すときは `\@restorepar` を呼び出します。

▼ 7.2.1 リスト型環境

▼ 7.2.2 自作の箇条書き型の環境

箇条書きとはいくつかの段落を改行や余白などを含めた一連の項目のことです。L^AT_EX ではすでにこの箇条書き型の環境が登場しています。主な環境として

- ラベル付きの `itemize` 環境。
- 番号付きの `enumerate` 環境。
- 説明ラベル付きの `description` 環境。
- 項目を中央揃えさせる `center` 環境。

などがあります。箇条書き型の環境では項目の始めにラベルを付けて字下げを挿入します。ラベルや字下げは必ずしも必要ではないので、中央揃えさせるだけの `center` 環境にも使えます。既存の箇条書き型環境で満足できないときは `list` 環境や、少し制限のある `trivlist` 環境のパラメータの変更を行います。

▼ 7.2.3 list 環境

`list` 環境は汎用性の高い箇条書き型環境です。

```
\begin{list}{<標準のラベル>}{<設定>}
<項目>
\end{list}
```

7.3 箇条書き環境の自作

```
\list{<ラベル>}{<命令>} <項目> \endlist
\item
```

図 7.1 list 環境で設定できる値

```
\begin{trivlist} <項目> \end{endlist}
```

図 7.2 リスト環境の形式

```
\leftmargin (0)
\labelwidth (0)
\itemindent (0)
\linewidth (\hsize)
```

```
\makelabel
\usecounter{<hoge>} \boxlabels
```

垂直空白 (スキップ)

```
\topsep (0)
\partopsep (0)
\itemsep (0)
\parsep (\parskip)
```

水平空白 (寸法)

```
\leftmargin ()
\rightmargin (0pt)
\listparindent (0pt)
\itemindent (0pt) \labelwidth (0) \labelsep (0)
```

```
\leftmargini \leftmarginii \leftmarginiii \leftmarginiv \leftmarginv
\leftmarginvi
```

\itemize と \enumerate カウンタ enumi, enumii, enumiii, enumiv,

▼ 7.3.1 trivlist 環境

▼ 7.3.2 enumerate など

7.4 数式

▼ 7.4.1 揃える

▼ 7.4.2 揃えない

▼ 7.4.3 複数行

▼ 7.4.4 数式番号

まあ、てきとうに改行して `\displaystyle` で表示させる、なんてこともできるわけで、それでいて equation カウンタを増分させて、てきとうに右端にでも表示させればいいわけで。

```

\def\equation{\!\!\nopagebreak%
2 \refstepcounter{equation}%
  \hfill\bgroup$\displaystyle}%$
\def\endequation{${\egroup\hfill(\theequation)\!\!\nopagebreak
  \ignorespacesafterend}}%$

```

似非ですけど。

```

\begin{equation}
f(x) = ax + b
\end{equation}
hogehoge

```

とした場合、`\end{equation}` の後の改行がホワイトスペースとなり、`'hogehoge'` の前にスペースとして表れるので `\ignorespacesafterend` が必要となります。

▼ 7.4.5 各種環境

7.5 表組み

```

1 \def\ialign{\everycr{}\tabskip\z@skip\halign}
  \def\oalign#1{\leavevmode\vtop{\baselineskip\z@skip \lineskip.25ex%
    \ialign{##\crcr#1\crcr}}}
```

```

\def\o@lign{\lineskiplimit\z@ \oalign}
\def\oalign{\lineskiplimit-\maxdimen \oalign}
6 \def\sh@ft#1{\dimen@.00#1ex\multiply\dimen@\fontdimen1\font
  \kern-.0156\dimen@} % compensate for slant in lowered accents
```

▼ 7.5.1 tabbing 環境

▼ 7.5.2 array 環境

▼ 7.5.3 tabular 環境

▼ 7.5.4 既存環境の拡張

7.6 浮動体

▼ 7.6.1 制御用パラメータ

▼ 7.6.2 図表見出し

▼ 7.6.3 図表目次

▼ 7.6.4 浮動体制御

7.7 参考文献

▼ 7.7.1 引用形式

▼ 7.7.2 一覧形式

7.8 ページスタイル

▼ 7.8.1 既存のスタイル

L^AT_EX の標準では

```
\thispagestyle{<スタイル>}
\pagestyle{<スタイル>}
```

とすることにより、特定ページだけのページスタイルの変更や、それ以降のページスタイルの変更が行なえます。標準的なクラスファイルで提供されているスタイルには

`empty` 左右上下、何も表示しない。

`plain` 左右ページ下端中央にページ番号のみを出力する。

`myheadings` `\markright{<偶数頁に出力する要素>}` か `\markboth` のいずれかにより
ヘッダーのマーク (`\leftmark`, `\rightmark`) を決める。

`headings` `\leftmark` (偶数頁), `\rightmark` (奇数頁) とページ番号 (両頁) をヘッダー
に出力する。

`book/report` 系クラスファイルでは

```
\leftmark := \chaptermark
\rightmark := \sectionmark
```

となり、`article` 系クラスでは

```
\leftmark := \sectionmark
\rightmark := \subsectionmark
```

等となることが多い。

▼ 7.8.2 ページスタイルのしくみ

L^AT_EX では

```
\pagestyle{headings}
```

などとすると

```
\def\ps@headings{%
  <ページスタイルの定義内容>%
}
```

が参照されることとなります。ここでは

`\@oddhead` 奇数ヘッダ

`\@oddfoot` 奇数フッタ

`\@evenhead` 偶数ヘッダ

`\@evenfoot` 偶数フッタ

の四つを最低でも定義しなければなりません。もっともシンプルな `empty` スタイルは次のように定義されています。

```
\def\ps@empty{%
2   \let\@mkboth \@gobbletwo
   \let\@oddhead \@empty
   \let\@oddfoot \@empty
   \let\@evenhead \@empty
   \let\@evenfoot \@empty
7 }
```

これらはおおよそすべてのページスタイルに関わるコマンドを空にします。 `\@empty` は波括弧 `{}` に展開されるので、

```
\def\@oddhead{}
```

と定義したことと同じ意味になります。 `\@mkboth` は

```
\let\@mkboth\@gobbletwo
```

として引数を二つ無効にするマクロとして代入されています。`\@mkboth` はユーザーが指定するものではなく、クラスファイルのなかなどで使われています。これは例えばユーザが

```
\markboth{ほげほげ}{ほげほげ}
\markright{ほげ}
```

とした場合は有効になります。

次に `plain` スタイルを見てみましょう。これは `\@oddfoot` と `\@evenfoot` の中央にページ番号を出力すれば良いだけなので、`empty` スタイルを少し書き換えるだけで良いことになります。

```
\def\ps@plain{%
  \let\@mkboth \@gobbletwo
3  \let\@oddhead \@empty
  \let\@evenhead \@empty
  \def\@oddfoot{\reset@font \hfil \thepage \hfil}%
  \let\@evenfoot \@oddfoot
}
```

`empty` と同様に空にするところは空にします。問題となる `\@oddfoot` は

```
\def\@oddfoot{\reset@font \hfil\thepage\hfil}%
```

として `\hfil` でサンドイッチにすることで中央にします。ついでに書体を標準の設定にするために `\reset@font` をここに追加します。このように `\@oddfoot` を定義し、これを `\@evenfoot` にも同様に代入します。

```
\let\@evenfoot \@oddfoot
```

これで `plain` が完成します。

次は `myheadings` を考えてみます。これはユーザーが `\leftmark` と `\rightmark` を適宜設定するということを前提とするため、案外簡単に定義できます。常識的・慣習的・規則的に、普通は奇数頁が右側、偶数頁が左側にくる事になります(左綴じの場合です、右綴じの場合は逆です)。`headings` なのでフッターは空にします。またページ番号はノドではなく小口に出すようにするのが古くからの慣習です。`\leftmark` と `\rightmark` はノドに出します。

```
\def\ps@myheadings{%
  \let\@oddfoot \@empty
  \let\@evenfoot \@empty
4  \def\@evenhead{\reset@font \thepage \hfil \leftmark}%
  \def\@oddhead{\reset@font\rightmark \hfil \thepage}%
  \let\@mkboth \@gobbletwo
  \let\chaptermark \@gobble
  \let\sectionmark \@gobble
9  \let\subsectionmark \@gobble % article 系
}
```


実は `\leftmark` や `\rightmark` は `\section` や `\chapter` などの特定の見出し命令が呼び出されたときに自動的に

```
\chaptermark{\chapter の必須引数}
\sectionmark{\section の必須引数}
```

などが実行されてマークが設定されてしまいます。これを無効化するために `\chaptermark` や `\sectionmark` (article 系の場合は `\chaptermark` はない) を引数を一つ無効にするという命令を代入します (`\@gobble`)。

これでとりあえずは `myheadings` が定義できます。適宜、ヘッダーの書体を変更するあり `\reset@font` で書体を標準にするなどが考えられます。欧文の場合は `\slshape` でスラント体に変更することが多いようです (本文とヘッダを明確に分離するために賢い方法)。

さて、いよいよ本命の `headings` を定義します。今回は `report/book` 系のクラスで用いることを前提とします (さらに `twoside` で `openright` というオプション付き)。article 系で用いる場合は

```
\chaptermark (0) -> \sectionmark (1)
\sectionmark (1)-> \subsectionmark (2)
```

などに変更するなどの細かい部分だけですので、すぐに応用できるでしょう。まずは特に難しく考えなくてもできるものから定義しましょう。 `\@mkboth` は今回 `\markboth`, `\markright` がユーザ支配ではないのでクラスファイル側支配とするため `\markboth` を代入します。フッターも空にします。 `\@evenhead` と `\@oddhead` は `myheadings` の定義と同様に構いません。

```
\def\ps@headings{%
  \let\@oddfont \@empty
  \let\@evenfont \@empty
  \let\@evenhead{\reset@font \thepage \hfil \leftmark}%
  \let\@oddhead{\reset@font \rightmark \hfil \thepage}%
  \let\mkboth \markboth
```

さて、ここまでは簡単ですが、問題の `\chaptermark` と `\sectionmark` を定義していません。 `\chaptermark` でやるべき事は `\leftmark` を適切に設定することです。この場合、前付・後付以外では「第 3 章 章見出しタイトル」となるようにします。さらに通常は `secnumdepth` によってユーザがカウンタをつけるか否かを制御するため、この条件判断も必要になります。 `\sectionmark` も同様な方法で判定します。

```
\def\chaptermark##1{%
  \markboth{%
    \ifnum \c@secnumdepth >\m@ne
    \if@mainmatter
      第 \thechapter 章\hskip 1zw
    \fi
    \fi
    ##1}{}%
  }%
\let\@presectionname\@empty
\let\@postsectionname\@empty
```

```

\def\sectionmark##1{%
  \markright{%
14     \ifnum \c@secnumdepth >\z@
        \if@mainmatter
            \@presectionname \thesection \@postsectionname \hskip 1zw
        \fi
        \fi
19     ##1}%
}%

```

`\chaptermark` も `\sectionarmk` の定義も `\def` の中の `\def` なので `##` で井桁をエスケープさせます。また、`secnumdeth` の裸のカウント数を参照するために `\c@secnumdepth` を `\ifnum` で使用していますし、`\@mainmatter` というブール値も本文かどうかを判定するために使用しています。大抵の `report/book` 系のクラスファイルは

```

表紙
\frontmatter
前付け
\mainmatter
5 本文
\backmatter
後付

```

という構造にするように設計されているので、`\mainmatter` が実行されたときに `\@mainmatter` は `'true'` になっています。欧文の場合は見出しのタイトルを大文字にするとか色々慣習があるので、引数全体を `\MakeUppercase` で括るということもあります。

しかし、このままでは実は問題があります。それは `\chapter` など、章見出しが存在するページのページスタイルです。これは `\chapter` 命令が呼び出された段階で

```
\thispagestyle{plain}
```

等のようにページ下端中央にページ番号だけを表示する仕様になっています。章見出しがあるページはそれだけで要素としての強度があるので、ページ番号を柱(ヘッダー)に出力すべきではない、という考え方もあります。これはヘッダに下線を引いた場合などに実感できます(このような思想の詳しい事は組版系の書籍を参照してください)。そこで、章見出しのあるページだけ、ページ番号を出力するシンプルなページスタイル `plain` を採用します。

大抵の場合は `headings` を通常のページスタイルとして `plain` は使っていません(書籍などでは)。そこで `plain` ページスタイルを再定義することで、一時凌ぎをします。

```

\def\ps@plain{%
  \let\@mkboth \@gobbletwo
  \def\@oddhead{\reset@font \hfil \thepage}%
4  \let\@evenhead \@empty
  \let\@oddfoot \@empty
  \let\@evenfoot \@oddfoot
}

```

これで、とりあえず、ページスタイルの基本は終了です。

▼ 7.8.3 自作のスタイル

さて、前回と前々回はページスタイルの基本的な事を学習しました。まだ、`\mark`, `\leftmark`, `\rightmark`, `\themark` の中身には触れていませんが、ある程度の体裁の調整は出来るようになっていきます。例えば、小口部分に `\leftmark`, `\rightmark` を出力せずにページ番号のそばにおくためには、次のように `\@oddhead`, `\@evenhead` を定義すれば良いことになります。

```
\def\@evenhead{\reset@font \thepage \hskip 1zw \leftmark \hfil}%
\def\@oddhead{\reset@font \hfil \rightmark \hskip 1zw \thepage}%
```

ヘッダーのノド部分にあるタイトル `\@headtitle` を、小口部分には `\leftmark`, `\rightmark` を表示させるようにします。ページ番号はフッターの小口に出すには次のようにします。

```
\gdef\@headtitle{好き好き \LaTeXe 初級編}%
\def\@evenhead{\reset@font \leftmark \hfil \@headtitle}%
3 \def\@oddhead{\reset@font \@headtitle \hfil \rightmark}%
\def\@evenfoot{\reset@font \bfseries \thepage \hfil}%
\def\@oddfont{\reset@font \bfseries \hfil \thepage}%
```

多くの書籍で見掛けることが出来ますが、ヘッダ部分に下線を付加する方法の一つを紹介します。`\textwidth` の幅を持つヘッダ用の箱 (`\hbox`) に対して下線を引けば良いので、次のように `\@evenhead` と `\@oddhead` を定義します。

```
\def\@evenhead{\underline{%
\hb@xt@ \textwidth{\thepage \hfil \leftmark}}}
\def\@oddhead{\underline{%
\hb@xt@ \textwidth{\rightmark \hfil \thepage}}}
```

`\hb@xt@` は `\hbox to` の事です。

おまけとして爪掛け (thumb-index) のあるページスタイルを考えてみます。爪は奇数ページだけに出力するものとし、爪部分は白色で章番号を、爪の背景は黒で、章番号に応じて爪の位置を下げるという事を実現することを考えます。簡単のために `picture` 環境を使って領域 0 の要素を追加します。今回は裁断における綴じの誤差が出ることを予想して、偶数ページには爪を表示させません。まずは爪を表示させる汎用的なマクロ `\page@tume` を次のように定義します。爪は本文 (`\mainmatter` の後) にだけ表示すれば良いので `\if@mainmatter` で判断をします。

```
1 \def\page@tume{%
  \if@mainmatter
    \setlength\unitlength{1truecm}%
    \begin{picture}(0,0)%
      \put(1,-\value{chapter}){%
6      \makebox(0,0)[t1]{\rule{1truecm}{1truecm}}}%
      \put(1,-\value{chapter}){%
          \makebox(0,-.4)[t1]{%
            \textcolor{white}{\hb@xt@ 1truecm{%
              \hfil \sffamily \thechapter \hfil}}}}}%
11 \end{picture}%
```

```
\fi
}
```

しかし、このままでは章番号がどんどん増えて行くとページの下端をはみ出して最終的にはあっちの世界に旅だってしまうので、適当な場所で折り返します。これは $(\text{textwidth} - 1\text{cm}) / 1\text{cm}$ で等によって算出することが出来るの (今回の場合は一つの爪が 1cm で構成される) で、

```
\newcount\cnt@tume
2 \AtBeginDocument{%
  \cnt@tume=\textheight
  \setbox0=\hbox{\vrule width 0pt height 1truecm}%
  \@tempcnta=\ht0
  \advance\cnt@tume -\@tempcnta
7 \divide \cnt@tume \@tempcnta
}
```

として `\cnt@tume` に適当な上限値を与えます。とりあえず、上限値を越える章番号 (`\c@chapter`) に関しては、上限値 (`\cnt@tume`) で割った余りで出力すれば良いので、

```
\newcount\cnt@tume@no
2 \def\get@tume@num{%
  \@tempcnta=\c@chapter
  \@whilenum \@tempcnta>\cnt@tume \do{%
    \advance\@tempcnta -\cnt@tume}%
  \cnt@tume@no=\@tempcnta
7 }
```

として現在の章に応じた余り (`\cnt@tume@no`) を求めるマクロ `\get@tume@num` を定義します。このようにして `\cnt@tume` (これはプリアンブルなどに配置すると良いが、ページレイアウトを変更するコマンドと `\cnt@tume@no` を求める準備が出来たならば、先程の `\page@tume` を次のように書き換えます。

```
\def\page@tume{%
  \if@mainmatter
3 \setlength\unitlength{1truecm}%
  \begin{picture}(0,0)%
  \get@tume@num % ここで \cnt@tume@no を求める
  \put(1,-\the\cnt@tume@no){%
    \makebox(0,0)[t1]{\rule{1truecm}{1truecm}}}%
8 \put(1,-\the\cnt@tume@no){%
  \makebox(0,-.4)[t1]{%
    \textcolor{white}{\hb@xt@ 1truecm{%
      \hfil \thechapter \hfil}}}}}%
  \end{picture}%
13 \fi
```

ヘッダが書き出される度に `\get@tume@num` が呼び出されて不経済ですが、今回はこれで妥協しましょう。全体をまとめると次のようになります。

```
\documentclass{book}
2 \usepackage[dvips]{color}
%
\makeatletter
```

```

\newcount\cnt@tume % 折り返し地点 + 1
\newcount\cnt@tume@no % 実際に爪を出力すべき位置
7 % chapter
\def\@prechaptername{第}
\def\@postchaptername{章}
% section
\let\@presectionname\@empty
12 \let\@postsectionname\@empty
% \cnt@tume を \textheight により求める
\AtBeginDocument{%
  \cnt@tume=\textheight
  \setbox0=\hbox{\vrule width 0pt height 1truecm}%
17 \@tempcnta=\ht0
  \advance\cnt@tume -\@tempcnta
  \divide \cnt@tume \@tempcnta
}
% \cnt@tume@no をそのときの \c@chapter により求める \get@tume@no
22 \def\get@tume@num{%
  \@tempcnta=\c@chapter
  \@whilenum \@tempcnta>\cnt@tume \do{\advance\@tempcnta -\cnt@tume}%
  \cnt@tume@no=\@tempcnta
}
27 % 実際に爪を表示するためのマクロ \page@tume
\def\page@tume{%
  \if@mainmatter
    \setlength\unitlength{1truecm}%
    \begin{picture}(0,0)%
32 \get@tume@num
    \put(1,-\the\cnt@tume@no){%
      \makebox(0,0)[t1]{\rule{1truecm}{1truecm}}}%
    \put(1,-\the\cnt@tume@no){%
37 \makebox(0,-.4)[t1]{%
      \textcolor{white}{\hb@xt@ 1truecm{%
        \hfil\sffamily\thechapter\hfil}}}}%
    \end{picture}%
  \fi
}
42 % \chapter のあるページにおけるスタイル \page@tume を \@oddhead に追加
\def\ps@plain{%
  \let\@mkboth \@gobbletwo
  \def\@oddhead{\reset@font \hfil \thepage\page@tume}%
  \let\@oddfoot \@empty
47 \let\@evenhead \@empty
  \let\@evenfoot \@oddfoot
}
% 主となるページスタイル
\def\ps@headings{%
52 \let\@oddfoot=\@empty
  \let\@evenfoot=\@empty
  \def\@evenhead{\underline{\hb@xt@ \textwidth{%
    \thepage \hfil \leftmark}}}%
  \def\@oddhead{\underline{%
57 \hb@xt@ \textwidth{\rightmark \hfil \thepage}}%
    \page@tume}% \@oddhead の最後に爪を追加
  \let\@mkboth=\markboth
  \def\chaptermark##1{%
    \markboth{%

```

```
62     \ifnum \c@secnumdepth >\m@ne
        \if@mainmatter
            \@prechaptername \thechapter \@postchaptername
            \hskip 1zw
        \fi
67     \fi
        ##1}{}%
}%
\def\sectionmark##1{%
    \markright{%
72     \ifnum \c@secnumdepth >\z@
        \if@mainmatter
            \@presectionname \thesection \@postsectionname \hskip 1zw
        \fi
    \fi
77     ##1}%
}%
}
\makeatother
\pagestyle{headings}
82 \begin{document}
```

第 8 章

L^AT_EX メモ帳

8.1 汎用的なくり返し式

通常のプログラミング言語の場合は、次のような for 文による繰り返し式を使う場面がしばしばあります。

```
int i, sum = 0;
for (i = 1; i < 11; i++){
3   sum := sum + i;
}
```

しかし、これは次のように while 文でも同様な処理を行なえます。

```
1 int i, sum;
  sum = 0;
  i = 1;
  while (i < 11){
    sum := sum + i;
6 }
```

T_EX は必要最低限の繰り返し式である `\loop`/`\repeat` なるコマンドを用意しています (これがあれば for 文や while 文を簡単に用意することが可能)。

```
{(初期化文) \loop (繰り返し文) (条件文)(実行文) \repeat }
```

別に初期化文とかはいらんです。上記の 1 から 10 の総和を求めるためには次のようにします。

```
\makeatletter
\@tempcnta=\z@
\@tempcntb=\z@
4 \loop \advance \@tempcntb \@ne \ifnum \@tempcntb < 11\relax
  \advance \@tempcnta \@tempcntb \repeat
  総和: \@arabic\@tempcnta
\makeatother
```

`\loop` の中における (繰り返し文) は (実行文) よりも先に実行されます。これがいやなときは do-while 文などを定義します。

▷ 例題 8.1 次の実行結果から `\repeat` と `\fi` の役割について考えてください。

```
\makeatletter
\@tempcnta=\z@
3 \loop \ifnum \@tempcnta<10
```

```

\the\@tempcnta\space
\advance \@tempcnta \@ne
\repeat
\def\check@tempcnta{%
8 \ifnum \@tempcnta<10
\the\@tempcnta\space
\advance \@tempcnta \@ne
% 再帰的に自分自身を呼び出す
\check@tempcnta
13 \fi
}
\@tempcnta=\z@
\check@tempcnta
\makeatother

```

実行結果は「0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9」となることから、`\loop/\repeat` と `\ifnum/\fi` による方法はかなり似ているという事が分かると思います。

`\loop/\repeat` では質素過ぎるので L^AT_EX には `\@whilenum` と `\@whiledim` が用意されています。

```

\@whilenum <整数値の比較式> \do<処理内容>
\@whiledim <長さの比較式> \do<処理内容>

```

まあ、とりあえずは以下の使用例を吟味してくださいな。

```

\makeatletter
\@tempcntb=\z@
3 \@tempcnta=\z@
\@whilenum \@tempcntb <11\relax \do{%
\advance \@tempcnta \@tempcntb
\advance \@tempcntb \@ne
}総和: \@arabic\@tempcnta\par
8 \@tempdima=.4pt
\@whiledim \@tempdima <1em\relax \do{%
\vrule width \@tempdima height 1em \kern \@tempdima
\advance \@tempdima \p@
}\par
13 \@tempdima=.4pt
\@whiledim \@tempdima <.5\linewidth\relax \do{%
\hrule width \@tempdima height .4pt \par
\multiply \@tempdima \tw@
}\par
18 \makeatother

```

```

\@whilesw <スイッチ> \fi <処理内容>

```

```

\newif\ifDOUYO
2 \DOUYOtrue
\@whilesw \ifDOUYO \fi {ほげほげ\DOUYOfalse}
\@whilesw \ifDOUYO \fi {げほげほ\DOUYOtrue}

```

8.2 jsclasses で図表番号の後にコロンを表示したいとか

図表番号のスタイルは `\thefigure`, `\thetable` で再定義できる。


```
1 \renewcommand*\thefigure{\@arabic\c@figure}
  \renewcommand*\thetable{\@arabic\c@table}
```

スタイルに関しては `\fnum@figure`, `\fnum@table` で決められる。`\figurename`, `\tablename` はそれぞれクラスファイルで定義されているものとする。

```
\documentclass[a4j]{jsarticle}
\makeatletter
3 \def\fnum@figure{\figurename\nobreak\thefigure:}
  \def\fnum@table{\tablename\nobreak\thetable:}
  \makeatother
  \begin{document}
  \begin{figure}[htbp]
8 \caption{ほげほげ}
  \end{figure}
  \begin{table}[htbp]
  \caption{どれどれ}
  \end{table}
13 \end{document}
```

スタイルを「第 8 図」などとしたければ

```
\def\prefigurename{第}
2 \def\postfigurename{図}
  \def\fnum@figure{\prefigurename \nobreak \thefigure \nobreak
  \postfigurename}
```

とすれば良いだろう。

8.3 T_EX 解剖

T_EX は生真面目過ぎて一般ユーザには分かりにくいプログラムなので消化器官で類推するという伝統的な方法が取られていました。私もこれ以上の類推を思い付かないので、これで解説しましょう。これは T_EX を解剖学的に目、口、食道、胃、腸の五つに準える事になります。T_EX が原稿ファイル $\langle file \rangle.tex$ に喰らいついた時から出力ファイル $\langle file \rangle.dvi$ が作成されるまでに、いったいどのような過程を経ているのか、以下の五つの流れを御覧になってください。

1. 「目 (eye*¹)」は入力ファイル $\langle file \rangle.tex$ を発見すると好物のそれを食べる（おやおや、私は T_EX の好みは熟知していますよ）。そうするとすばやく胃まで運ぶように命令する。
2. 「口 (mouth)」は $\langle file \rangle.tex$ に含まれている文字を食道を通りやすいようにトークンに噛み砕く。
3. 「食道 (gullet)」は口から運ばれてきたトークンのマクロや条件判断を可能な限り展開・置換して胃で吸収しやすいようにトークンリストに変換する。
4. 「胃 (stomach)」ははるばる口から運ばれてきたトークンリストをさらに消化して一つのコマンドで構成される小さなブロックを作る。ブロックは予め腸で消化しやすいように改段落、改ページをしておく。
5. 「腸 (intestine)」は胃で組み上げられたページをデバイスドライバが解釈できる DVI 形式に変換し出力ファイル $\langle file \rangle.dvi$ に書き込む。

これをざっくり理解するためには、すでにある程度 T_EX の組版のしくみを理解している必要があるというのは残念なことであるが、とりあえず以下の記述を云々。

```

1 \tracingall% 解剖するヨ
  \nopagenumbers % ノンブルはなしよ
  V\kern-.4em V% V が二つで W もどき
  \vfill\ eject% 改ページする
  \def\hoge{V\hfill V}% マクロ \hoge を定義する
6 \hoge\par% 段落終了
  \bye% 終わり

```

上記のファイル $\langle file \rangle.tex$ を `tex file` とでもすれば、 $\langle file \rangle.log$ に色々とデータが書き出されることとなります。その前に、焦らずに入力ファイル $\langle file \rangle.tex$ が T_EX の「目」で見られたときにはどのようなになっているのか確認してください。さらに「口」に入ったとき、次のように噛み砕かれます。

```

tracingall
nopagenumbers
V11 kern -12 .12 411 e12 m12 10 V11 vfill eject
def hoge {1 V11 hfill V11 }2

```

*1 一度に一文字しか見ることはないので、目は一つあれば十分でしょう。

```
hoge par bye
```

これらからページを構成するための要素が決定されます。(\\tracingall とか \\nopagenumbers はページを直接構成するための要素ではありません)。

```
{vertical mode: the letter V}
{horizontal mode: the letter V}
3 {\kern
  {the letter V}
  {\vfill}
  {\par}
  {vertical mode: \vfill}
8 {\par}
  {\penalty}
```

垂直モードでの文字 ‘V’ は「水平モードに移行する」という役割りもあるようです。そうしてから文字 ‘V’ が組まれるようです。\\par の後も垂直モードに移行しているのが分かります。

「食道」ではマクロなどがバラバラにされてしまいます。

```
1 \nopagenumbers ->\footline{\hfil}
  \eject ->\par \break
  \break ->\penalty -\@M
  \hoge ->V\hfill V
  \bye ->\par \vfill \supereject \end
6 \supereject ->\par \penalty -\@MM
```

「胃」で最終的に組み上げられた 1 ページ目は次のようになります。

```
Completed box being shipped out [1]
\vbox(667.20255+0.0)x469.75499
.\vbox(0.0+0.0)x469.75499, glue set 14.0fil
4 ..\glue -22.5
  ..\hbox(8.5+0.0)x469.75499, glue set 469.75499fil
  ... \vbox(8.5+0.0)x0.0
  ... \glue 0.0 plus 1.0fil
  ..\glue 0.0 plus 1.0fil minus 1.0fil
9 .\vbox(643.20255+0.0)x469.75499, glue set 633.20255fill
  ..\glue(\topskip) 3.16669
  ..\hbox(6.83331+0.0)x469.75499, glue set 438.75499fil
  ... \hbox(0.0+0.0)x20.0
  ... \tenrm V
14 ... \kern -3.99994
  ... \tenrm V
  ... \penalty 10000
  ... etc.
  ..\glue 0.0 plus 1.0fill
19 .\glue(\baselineskip) 24.0
  .\hbox(0.0+0.0)x469.75499, glue set 469.75499fil
  ..\glue 0.0 plus 1.0fil
```

先頭のピリオドはボックスの入れ子の深さを表しています。一番最初の \\vbox がページ全体をいれるための大きなボックスです。2 ページ目も同じ様に組まれます。



GNU Free Documentation License

Version 1.2, November 2002

Copyright © 2000,2001,2002 Free Software Foundation, Inc.

59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "**Document**", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "**you**". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "**Modified Version**" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "**Secondary Section**" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related mat-

ters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "**Invariant Sections**" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "**Cover Texts**" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "**Transparent**" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "**Opaque**".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "**Title Page**" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "**Entitled XYZ**" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "**Acknowledgements**", "**Dedications**", "**Endorsements**", or "**History**".) To "**Preserve the Title**" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the

- same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
 - C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
 - D. Preserve all the copyright notices of the Document.
 - E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
 - F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
 - G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
 - H. Include an unaltered copy of this License.
 - I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
 - J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
 - K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
 - L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
 - M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
 - N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
 - O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover

Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggre-

gate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright ©YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-

Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.



参考文献

- [1] Johannes Braams, David Carlisle, Alan Jeffrey, Leslie Lamport, Frank Mittelbach, Chris Rowley, and Rainer Shopf. *L^AT_EX 2_ε Sources*, 2001
- [2] 江口庄英. *Ghostscript Another Manual*. ソフトバンク, 1997
- [3] Donald E. Knuth. 改訂新版 T_EX ブック. アスキー, 1991. 斎藤信男監修. 鷺谷好輝訳
- [4] ———. METAFONT ブック. アスキー, 1994. 鷺谷好輝訳. 007.64/Kn
- [5] 乙部巖己, 江口庄英. *pL^AT_EX 2_ε for Windows Another Manual Vol. 2 Extended Kit*. ソフトバンク, 1997
- [6] ———. *pL^AT_EX 2_ε for Windows Another Manual Vol. 1 Basic Kit 1999*. ソフトバンク, 1998
- [7] ページ・エンタープライゼズ. L^AT_EX 2_ε 【マクロ&クラス】プログラミング 基礎編. ページエンタープライゼズ, 2002
- [8] L^AT_EX3 Project Team. *Modifying L^AT_EX*, 1995
- [9] ———. *L^AT_EX 2_ε for class and package writers*, 1999
- [10] ———. *L^AT_EX 2_ε font selection*, 2000
- [11] ———. *L^AT_EX 2_ε for authors*, 2001
- [12] ———. *Configuration options for L^AT_EX 2_ε*, 2001
- [13] Vlandimir Volovich, Werner Lemberg, and L^AT_EX3 Project Team. *Cyrillic languages support in L^AT_EX*, 1999
- [14] 吉永徹美. L^AT_EX 2_ε 【マクロ&クラス】プログラミング 実践編. ページエンタープライゼズ, 2003



命令索引

数字/記号

\(32
\)	32
\-	91
\@	86
\@hoge	7
\@hyph	90
\@italiccorr	90
\@par	90, 92
\@alph	40, 41
\@arabic	22, 40
\@backslashchar	91
\@bin@list	60
\@car	39
\@carcube	39
\@cclv	25
\@cclvi	25
\@cdr	39
\@charlb	14
\@charrb	14
\@cons	39
\@dblarg	37
\@dec@list	60
\@depth	90
\@dischyph	90
\@empty	38, 89, 90, 97
\@evenfoot	97, 98
\@evenhead	97, 99, 101
\@expandtwoargs	91
\@firstofone	91
\@firstoftwo	91
\@fnsymbol	40
\@gobble	91, 99
\@gobblefour	91
\@gobbletwo	91
\@headtitle	101
\@height	90
\@hoge	7
\@iden	91
\@ifdefinable	38
\@ifnextchar	37, 80
\@ifstar	37, 38, 40, 80
\@ifundefined	38
\@latex@error	78
\@latex@info	78
\@latex@info@no@line	78
\@latex@warning	78
\@latex@warning@no@line	78
\@let@token	51
\@M	25
\@m	25
\@mainmatter	100
\@makeother	13
\@Mi	25

\@Mii	25
\@Miii	25
\@minus	90
\@Miv	25
\@mkboth	97-99
\@MM	25
\@namedef	37
\@nameuse	37
\@ne	25
\@oddfoot	97, 98
\@oddhead	97, 99, 101
\@par	92
\@plus	90
\@restorepar	92
\@roman	40, 41
\@sanitize	13, 49
\@secondoftwo	91
\@setpar	92
\@spaces	90
\@star@or@long	40
\@startsection	33, 80
\@temp@char	60
\@tempcnta	16, 17, 29
\@tempcntb	16
\@tempdima	17, 23, 33, 43
\@tempdimb	17
\@tempdimc	17
\@tempskipa	19
\@tempskipb	19
\@temptokena	21
\@tfor	43, 60
\@themark	101
\@thirdofthree	91
\@warning	78
\@whiledim	106
\@whilenum	60, 106
\@whilesw	106
\@width	90
\@xobeysp	83
\@xxxii	25

A

\active	11, 14
\addcontentsline	81, 82
\addtocontents	81, 82
\advance	25
\allowbreak	90
alltt 環境	49
\and	32
\arabic	40
array 環境	92, 96
\author	53

B

\baselineskip	46
\begin	42
\begingroup	7, 49
\beginroup	7
\bgroup	7, 89
\bibitem	55
\bigbreak	90
\boolean	31
\box	15, 20, 23, 24, 34
\boxlabels	94
\break	90

C

\c@chapter	102
\c@page	41
\c@secnumdepth	100
\caption	43
\catcode	13, 14
\centering	80
center 環境	2, 92
\chapter	49, 99, 100
\chaptermark	99, 100
\char	10
\chardef	25
\check@meaning	40
\CheckDefine	39
\ClassError	78
\ClassInfo	78
\ClassWarning	78
\ClassWarningNoLine	78
\clearpage	50
\cnt@lines	22
\cnt@tume	102
\cnt@tume@no	102
comment 環境	31
\count	15
\count@	16, 27
\count2	27
\countdef	25

D

\date	53
\day	27
\DeclareOption	47
\DeclareRobustCommand	39
\dectobin	60
\def	3, 4, 6, 8, 100
description 環境	92
\dimen	15, 34
\dimen@	17
\dimen@i	17
\dimen@ii	17
\displaystyle	95

\divide 25, 26
 \do 106
 \document 76
 document 環境 2
 \dp 20

E

\edef 3, 4, 6, 11, 60, 69
 \egroup 7, 89
 \else 22, 27, 29
 \empty 35, 38, 89
 \end 42
 \endfigure 8
 \endgraf 89
 \endgroup 7, 49
 \endline 89
 \enumerate 94
 enumerate 環境 92, 95
 \equal 31
 \everypar 21, 22, 33
 \expandafter 29

F

\fbox 41, 71, 75, 76
 \fboxrule 76
 \fboxsep 76
 \fi 22, 27, 29, 105, 106
 \figure 8
 \figurename 107
 figure 環境 8
 \filecontents 77
 \filecontents* 77
 \fmtname 29, 89
 \fmtversion 89
 \fnum@figure 107
 \fnum@table 107
 \framebox 21, 41, 71, 73
 \frenchspacing 85
 \fussy 87
 \futurelet 51

G

\gdef 3, 6, 7, 14
 \geho 4
 \GenericInfo 77
 \get@tume@num 102
 \global 3, 6, 8, 41
 \group 7

H

\hangafter 85
 \hangindent 85
 \hb@xt@ 90, 101
 \hbox 23, 41, 42, 70, 75, 101
 \hboxto 101
 \heartsuit 21
 \hfil 98
 \hfill 22
 \hoge 4, 7, 33
 \hrule 41, 73, 75
 \hsize 94
 \ht 20
 \hyphenation 86, 87

I

\if 27, 28, 47

\if@afterindent 49
 \if@mainmatter 101
 \if@tempswa 22
 \ifcase 29
 \ifcat 28
 \ifdim 30
 \iffalse 30
 \ifFileExists 55, 77
 \ifhbox 71
 \ifhmode 71
 \ifinner 71
 \ifmmode 14, 71
 \ifnum 29, 100, 106
 \ifodd 29
 \ifSQ 14
 \ifthenelse 31
 \iftrue 30
 \ifvbox 71
 \ifvmode 71
 \ifvoid 71
 \ifx 28, 29, 39
 \ignorespacesafterend 95
 \iiemdash 57
 \includegraphics 2
 \indent 70
 \index 13
 inputex 環境 49
 \InputIfFileExists 77
 \isodd 31
 \item 92
 \itemindent 94
 \itemize 94
 itemize 環境 92
 \itemsep 94

J

\jfmtname 29
 \jobname 43

K

kaito 環境 54

L

\l@ngrel@x 40
 \l@abelsep 94
 \l@abelwidth 94
 \l@angle 14
 \Large 80
 \LaTeX 34, 39
 \l@black 89
 \leavevmode 70, 71
 \leftmargin 94
 \leftmargini 94
 \leftmarginii 94
 \leftmarginiii 94
 \leftmarginiv 94
 \leftmarginv 94
 \leftmarginvi 94
 \leftmark 97–99, 101
 \leftskip 85
 \lengthtest 31
 \let 8, 28, 30, 34
 \linebreak 83
 \linewidth 94
 \list 92
 \listfiles 76

\listparindent 94
 list 環境 92
 \llap 22
 \long 3, 4, 33, 40
 \loop 105, 106
 \lq 89

M

\m@th 90
 \magstep 46
 \mainmatter 100, 101
 \makeatletter 13, 47
 \makeatother 13, 47
 \makelabel 94
 \maketitle 53
 \MakeUppercase 80, 100
 \mark 101
 \markboth 97, 99
 \markright 97, 99
 \mathchardef 25
 \meaning 32, 34
 \medbreak 90
 \month 27
 \multiply 25, 26, 60
 \muskip 15
 \myboxrule 72
 \myboxsep 72
 \myfbox 73
 \myframebox 72
 \mysanitize 49
 myTT 環境 49, 50

N

\NeedsTeXFormat 45, 46
 \newboolean 31
 \newbox 20
 \newcommand 3, 6, 40
 \newcount 15, 16
 \newcounter 40
 \newdimen 17
 \newenvironment 3
 \newif 22, 47
 \newmuskip 19
 \newread 76
 \newskip 19
 \newtoks 21
 \newwrite 54, 76
 \nobreak 83, 90
 \nobreakspace 83
 \nofiles 76
 \nolinebreak 83
 \nonfrenchspacing 85
 \nopagebreak 83
 \nopagenumbers 109
 \normalbaselines 85
 \not 32
 \null 89
 \number 16, 17, 29, 60

O

\obeylines 50, 90
 \obeyspaces 83, 90
 \or 29, 32
 \orig@par 22
 \outer 3
 \output 21, 45, 50

P

`\p@` 18
`\PackageError` 78
`\PackageInfo` 78
`\PackageWarning` 78
`\PackageWarningNoLine` 78
`\page@tume` 101, 102
`\pagebreak` 83
`\pagenokazu` 41
`\pagenumbering` 41
`\pagestyle` 96
`\par` 3, 4, 22, 28, 50, 92, 109
`\paragraph` 80
`\parindent` 17, 50
`\parsep` 94
`\parshape` 85
`\parskip` 94
`\partopsep` 94
`\pfmtname` 29, 58
`picture` 環境 101
`\ProvidesClass` 46
`\protect` 34, 35, 39
`\protext` 34
`\ProvidesClass` 45, 46

R

`\raggedleft` 80
`\raggedright` 80
`\rangle` 14
`\rblack` 89
`\relax` 17, 33, 35, 39
`\renewcommand` 45
`\renewenvironment` 31
`\repeat` 105, 106
`\reset@font` 80, 98, 99
`\rightmargin` 94
`\rightmark` 97–99, 101
`\rightskip` 85
`\ronannumeral` 16, 17
`\rq` 89

S

`\sb` 14
`\scriptsize` 22
`\secdef` 80
`\section` 33, 43, 49, 99
`\sectionarmk` 100
`\sectionmark` 99

`\set` 17
`\setboolean` 31
`\setbox` 20
`\sffamily` 80
`\show` 32–34
`\showoutput` 84
`\showthe` 33, 34
`\sixt@n` 25
`\skip` 15, 34
`\skip@` 19
`\slash` 90
`\sloppy` 87
`\slshape` 99
`\smallbreak` 90
`\sp` 14
`\space` 89, 90
`\string` 35
`\strut` 71
`\strutbox` 71
`\symbol` 10

T

`tabbing` 環境 96
`\tablename` 107
`\tableofcontents` 43
`tabular` 環境 92, 96
`\tempboxa` 20
`\TeX` 86
`\textcompwordmark` 91
`\textemdash` 91
`\textendash` 91
`\textexclamdown` 91
`\textheight` 46
`\textquestiondown` 91
`\textquotedblleft` 91
`\textquotedblright` 91
`\textquoteleft` 91
`\textquoteright` 91
`\textsuperscript` 91
`\texttrademark` 91
`\texttt` 14
`\textwidth` 101, 102
`\the` 16, 17, 20, 21, 24, 27, 34
`thebibliography` 環境 55
`\thefigure` 106
`\thepage` 41
`\thetable` 106
`\thispagestyle` 96

`\thr@@` 25
`\three@digits` 29
`\time` 27
`\title` 53
`\toks` 15, 34
`\topsep` 94
`\topskip` 46
`\tracingall` 84, 109
`\tracingcommands` 84
`\tracinglostchars` 84
`\tracingmacros` 84
`\tracingpages` 84
`\tracingparagraphs` 84
`\tracingrestores` 84
`\tracingstats` 84
`trivlist` 環境 49, 92, 95
`\ttfamily` 49
`\tw@` 25
`\two@digits` 27, 29
`\typein` 89
`\typeout` 89

U

`\undefined` 38
`\unhbox` 71
`\unhcopy` 71
`\usecounter` 94

V

`\vbox` ... 41, 42, 45, 46, 70, 75, 109
`verbatim` 環境 22, 55
`\voidb@x` 20, 71
`\vrule` 41, 73, 75

W

`\wd` 20, 23
`\whiledo` 31

X

`\xdef` 3, 6, 7

Y

`\year` 27

Z

`\z@` 18
`\z@skip` 19



索引

数字/記号

16 進数 9
8 進数 9

A

afterpage 50
alltt 49, 50
article 49, 97, 99
ASCII 9

B

book 49, 97, 99, 100

C

calc 27, 51
Computer Modern 11

D

date 5
David Carlisle 49–51
depth 1
DVI 108

E

enumi (カウンタ) 94
enumii (カウンタ) 94
enumiii (カウンタ) 94
enumiv (カウンタ) 94
equation (カウンタ) 95
eye 108

G

gullet 108

H

height 1

I

ifthen 31
indentfirst 49
intestine 108

J

jbook 77
Johannes Braams 49
jsarticle 46

K

kaito.tex 53

L

ltpageno.dtx 41
ltpplain.dtx 13, 25

M

Make 6
mouth 108

N

NULL 文字 13

O

openright 99
OT1 11, 12

P

pdftex 42
preload file 77
ptex 42

R

reference point 1
report 49, 97, 99, 100

S

secnumdepth (カウンタ) ... 99
secnumpdeth (カウンタ) .. 100
SGML 2
source2e.tex 1, 34
stomach 108

T

T1 11, 12
tex 42
theorem 53, 54
thumb-index 101
twoside 99

U

undefined 33

W

width 1

X

xspace 50, 51

あ

アセンブリ言語 2
アットマーク 13
アロケーション 2, 23
アンダーバー 13
アンパサンド 13

い

胃 108
井桁 13
入れ子 9
インチ 18

う

うごう 30
上付き添字 13

え

エスケープ 6
エスケープ文字 13
演算子 2

お

オプション
openright 99
OT1 11, 12
T1 11, 12
twoside 99

か

改行文字 13
改段落 108
改ページ 108
解剖学 108
カウンタ 15
enumi 94
enumii 94
enumiii 94
enumiv 94
equation 95
secnumdepth 99
secnumpdeth 100
カテゴリコード 2, 9, 28
環境
alltt 49
array 92, 96
center 2, 92

comment 31
 description 92
 document 2
 enumerate 92, 95
 figure 8
 inputex 49
 itemize 92
 kaito 54
 list 92
 myTT 49, 50
 picture 101
 tabbing 96
 tabular 92, 96
 thebibliography 55
 trivlist 49, 92, 95
 verbatim 22, 55

き

記号 9
 基準点 1

く

口 108
 クラス
 jbook 77
 jsarticle 46
 クラスオプション 47
 グルー 2
 グルーピング 9
 グループの終わり 13
 グループの開始 13

こ

コマンド 2, 9
 コメント 13

さ

索引 13

し

シセロ 18
 下付き添字 13
 消化器官 108
 食道 108
 人名

Alan Jeffrey iii
 Cho Jin-Hwan iii
 Chris Rowley iii
 David Carlisle iii, 31, 49–51
 Donald E. Knuth iii, 2, 11
 Frank Liang 86
 Frank Mittelbach iii
 Johannes Braams iii, 49
 Leslie Lamport iii, 2, 31
 Mark Wicks iii
 Michael Downes iii
 Nelson Beebe iii
 Oren Patashnik iii

Pehong Chen iii
 Rainer Schöpf iii
 Sebastian Rahtz iii
 大島利雄 iii
 奥村晴彦 46
 乙部巖己 iii
 角藤亮 iii
 中野賢 iii
 平田俊作 iii

す

垂直モード 71
 スイッチ 22
 水平モード 71
 数式モードへの移行 13
 スキップ 19
 スケールポイント 18
 スコープ 9, 85
 スペース 13

せ

整列区切り 13
 宣言 9
 宣言型 2
 宣言型命令 42
 センチメートル 18

た

大域変数 85
 高さ 1
 タブ文字 13
 単位 17

ち

腸 108
 チルダ 13

つ

爪掛け 101

て

デイドーポイント 18
 デリート文字 13
 伝統 108

と

トークン 21, 108
 トークンリスト 108
 トークンリストパラメータ 21
 トークンリストレジスタ 21
 ドル 13

な

内部コード 2, 21
 長さ 17

に

入力ファイルの行の終わり 13

は

パーセント 13
 パイカ 18
 パターンマッチ 4
 バックスラッシュ 13
 パッケージ
 afterpage 50
 alltt 49, 50
 article 49, 97, 99
 book 49, 97, 99, 100
 calc 27, 51
 ifthen 31
 indentfirst 49
 report 49, 97, 99, 100
 theorem 53, 54
 xspace 50, 51

ハット 13
 幅 1

ひ

左波括弧 13
 ビッグポイント 18
 日付 5

ふ

ファイル
 kaito.tex 53
 ltpageno.dtx 41
 ltplain.dtx 13, 25
 source2e.tex 1, 34
 フォントサイズ 47
 フォントメトリクス 2
 深さ 1
 複合条件判断 2, 32
 普通の文字 13
 フラグ 22
 プリミティブ 1
 プログラム
 date 5
 Make 6
 pdftex 42
 ptex 42
 tex 42
 ブロック 108
 文中見出し 80

ほ

ポイント 18
 ボックス 1, 20
 ボックス・グルーモデル 1

ま

マークアップ 2
 マクロの変数 13

み

右波括弧 13
 ミリメートル 18

む			
無効文字	13	垂直——	70
も		水平——	70
モード		数式——	70
限定水平——	70	ディスプレイ数式——	70
		内部垂直——	70
		れ	
		レジスタ	15
		目	108
		文字コード	9, 27, 28

好き好き L^AT_EX 2_ε 中級編

© 渡辺徹 2005, 2007

発行日 2005 年 03 月 23 日 第 0.0.1 版 配布

発行日 2007 年 06 月 30 日 第 0.0.2 版 配布

編集 渡辺徹
