コンピュータガイド

– アプリケーション編 –

–cc 環境の利用者へ–

京都産業大学
計算機センター事務室

初版'94. 3.23
改定'96. 4. 1

緒　　言

本学でコンピュータ関係の出版物を最初に発刊したのは１９６８年である。研究報告を主としたものであったが、計算機科学研究所彙報として刊行された。当時はＴＯＳＢＡＣ３４００というバッチ処理のみの計算機であった。その後、１９８３年からは計算機科学研究所所報として継続的に発刊されている。その頃、計算機学習コースという科目が設けられ、受講者対象であるが、計算機利用マニュアルが作られている。当時のＦＡＣＯＭ　Ｍ１８０というＴＳＳ処理の計算機のために、端末機などの利用部分はセンタのメンバーが、プログラム言語の部分は各担当の先生が書いたものをまとめ、毎年改訂していた。　そして、１９９４年に全学の利用者のために、ＵＮＩＸガイドが発刊された。これは、内容と紙面の斬新さのために好評となり、９月にはたちまちＵＮＩＸガイド増補版が出された。１９９５年にその改訂版を出版し、１９９６年の本書となった。

コンピュータの発展は本当に速い。本書の初版はＵＮＩＸワークステーションの利用のためであった。本学にＵＮＩＸが導入されたのは１９８５年の立石電機製のスーパメイトである。なんとＵＮＩＸはシステムＶ１．０であった。その後１９８９年に工学部開設とともに、ワークステーションが急激に増加した。さらに１９９４年にはＵＮＩＸサーバとワークステーション教室が設置され、インターネット利用のワークステーションがなだれ込むようになった。また、１９９５年には１００台近いＭＡＣ機の教室が増設され、ネットワークの利用者が飛躍的にのびている。

ガイドブックの内容も変化せざるを得ない。プログラム言語から応用ソフトへの内容変化はすでに終わっている。そして、コンピュータ利用のためのガイドブックからネットワーク利用のためのガイドブックに変りつつある。この内容の変化に即時に対応できるのは学内の多大なユーザサポートとセンタメンバーのたゆまない努力の結果であると確信している。これからも、本書の充実に期待したい。

最後に、学内のユーザにお願いしたいことをひとつ。ネットワークに接続されたコンピュータは世界中のコンピュータを使い、また使われる可能性がある。つまり、本学のネットワークとコンピュータ利用のモラルは世界から注目されうることを知ってもらいたい。やって良いこと、やってはいけないことを知り、また我流ではなく、スマートに使いこなすために、おおいに本書を活用していただきたい。

工学部　黒住 祥祐

コンピュータガイドの読み方

コンピュータガイドは初心者が京都産業大学の共用 UNIX コンピュータを利用して生活する為の手引書として書かれています。

インターネット編では UNIX コンピュータを利用してインターネットの主なサービスである、電子メールやニュース、World Wide Web などが使えるようになるまでを説明しています。ここまでは全ての UNIX 利用者に習得して欲しいと思います。コンピュータの事をほとんど何も知らなくても、何とかここまではたどりつけるように書いたつもりです。頑張って下さい。

本冊子、アプリケーション編では UNIX コンピュータを利用してもっと色々なことができるように、cc 環境で利用できる様々なアプリケーションの使い方を紹介しています。第 1 章では自分だけの World Wide Web のページの作成の方法、第 2 章では、きれいに製本ができる TEX など、知れば知るほど UNIX の世界が広がること請け合いの内容です。インターネット編を読み終えたら是非 1 度チャレンジして下さい。

基本的にリファレンス要素が強い冊子です。必要な時に見られるよう、是非いつも携帯して下さい。

各章ではそれぞれ以下の事について説明しています。是非読んでみて下さい。

第 1 章	HTML	World Wide Web のページ作成について
第 2 章	LaTeX	文書処理システム LaTeX について
第 3 章	AUCTeX	LaTeX を Mule と組み合わせてより便利にする AUCTeX について
第 4 章	レポートシステム	レポートを電子メイルで提出するシステムについて
第 5 章	Mathematica	数式処理システム Mathematica について
第 6 章	GNUPLOT	グラフ作成システム GNUPLOT について
第 7 章	tgif	画像作成システム tgif について
第 8 章	*xv*	画像処理システム *xv* について
第 9 章	xpaint	画像作成システム xpaint について
第 10 章	NQS	バッチ処理を実現するシステム NQS について

別冊、コンピュータガイド －インターネット入門編－ の見出し一覧

はじめに	全ての利用者に
UNIN はいかが	UNIX の紹介
UNIX への道	教室別に UNIX を使うための方法について
UNIX それから	UNIX の入門
ネットワークの世界へようこそ	電子メール、ニュース、WWW、FTP について
UNIX もっともっと	UNIX のより進んだ使い方について
Mule	テキストエディタ Mule について
ターミナル接続	自宅から学校のコンピュータを利用する手続きについて
NeXT	NeXT コンピュータの使い方について

また、巻末の付録にはリファレンスとして各種コマンドや記号などの一覧表、情報処理教室を利用する際の注意、著作権法の抜粋などがまとめられています。参考文献の一覧も付けておきました。

目 次

第 1 章

HTML:
おいしいホームページの作り方

1.1 はじめに

Netscape Navigator (以下「Netscape」) のようなホームページを見るためのブラウザ（閲覧ソフト）であちこちのホームページを巡っていると（つまり「ネットサーフィン」していると)、「おーっ！」と思うようなカッコいいページに出会うことがありますよね。そういうページを見ると、自分でもホームページを作りたくなってきませんか？ 実は、京都産業大学の cc 環境では、ホームページは思ったよりも簡単に作ることができます。ここでは、ホームページの基本的な作り方を記します。あなたも、ホームページを作って、魅力的な情報を世界に発信しましょう！

さて、ホームページは思ったよりも簡単にできると書きましたが、ホームページを書くための言語があるので、それに従って書く必要があります。いえ、言語と言ってもそれほど難しくはありません。ホームページを書くための言語のことを **HTML** と言います。HTML とは、**HyperText Markup Language** の略です。なんだか難しそうですが、分かりやすく言えばこういうことです。WWW でネットサーフィンしていると[1]、あるページからあるページにジャンプしているように見えますが (**リンク「」している**と言います)、そのような仕組みになっているテキストを**ハイパーテキスト (hypertext)** と言います。そして、ハイパーテキストを可能にするように文書の中で指定することを**マークアップ (markup) する**と言います[2]。そのようにハイパーテキストをマークアップする言語のことを HTML と言います。

1.2 HTML を書く！その前に：とっても大事な著作権

いよいよホームページを書くことになります。しかしながら、自分の書きたいこと、自分の入れたいものをなんでもかんでも入れて良いというわけではありません。一つ気を付けておかなければならないことがあります。それは、**著作権法を侵さないようにする**ということです。「著作権って何？」という人もいると思いますので、ホームページに入れても良いものと入れてもいけないものを少し具体的に書いておきます[3]。ただし、これは一例ですから、分からない場合には法律に詳しい人に尋ねると良いでしょう。

[1] WWW (= World Wide Web) については、「WWW」の章をご覧下さい。
[2] LaTeX も本文以外に色々な命令を埋め込まなければいけないので、マークアップする言語の一種ですね。
[3] 著作権法は、巻末にも掲載されていますので、ご覧ください。

1.2.1　ホームページに載せても良いもの

ホームページに載せても良いものとしては、例えば次のものが考えられるでしょう。

- 自分で書いたエッセイやレポートや論文
- 映画やコンサートや CD などの感想
- 自分で撮った写真、自分で撮った映画、自分で描いた絵
- 自分で作った音
- 自分で作ったプログラム

など、要するに自分で作ったオリジナルなものはどんどん入れましょう。ただし、自分が撮ったものでも写真や映画などは**肖像権**の問題もありますから、**自分以外の人が写っている場合には、その人にホームページに掲載しても良いかどうかを聞いて許可を得た上で載せましょう。**そうしないと、訴えられることもあるかもしれませんよ。;-)

1.2.2　ホームページに載せてはいけないもの

ホームページに載せてはいけないものとしては、例えば次のものが考えられます。

- **雑誌や本のグラビアなどから取った写真や絵**
- **アイドル歌手などの CD などからサンプリングした音**
- **他人の曲の歌詞**
- **他人が書いた記事やホームページ**

など、要するに**他人が作ったものを無断で自分のホームページに入れてはいけません！**これらは著作権法違反という立派な法律違反で、当然罰則もあります。

残念ながら、色々なホームページを見てみると、この著作権法が守られていないページも数多くあるようですこのようなページを作ることは、本当は許されていません！ 京都産業大学の中でその様なページが見つかった場合には、まず警告などがされます。それでも改善される様子が見られない場合には、ホームページが強制的に削除されたり、ユーザ名の抹消などもあり得ますので、十分気を付けてください。

1.3　HTML を書くための準備

1.3.1　ホームページ用のディレクトリはどこ？

このことに関しては、「WWW」の章や、京都産業大学のホームページからたどれる「各種ドキュメント」の中の「WWW 個人ページを作ろう」に詳しく書いてありますので、そちらを見て下さい。URL[4]は、

`http://www.kyoto-su.ac.jp/information/Howtomake_Yourpage.html`

[4]「WWW」の章に説明がありますが、Uniform Resource Locator の略です。とりあえず、ホームページなどの住所だと思っておいたらよいでしょう。

です。

しかし、少しだけ説明しておきます。

あなたのホームページ用のディレクトリはどこにありますか？ 何ですって？まだない？ それでは、作らなければいけないですね。cc2000 に login した後に、`wwwmkdir` コマンドを実行して下さい (1996 年 2 月時点)。

```
cc2000(83)% wwwmkdir
```

これを実行すると、ホームページ作成に関する注意書きが出てきますので、しっかりと読んで下さい。そこには、**著作権を侵さないように**ということが書いてあります。それに同意できれば、`y` を押してリターンを押します。成功したならば、しばらく待つと、どこのディレクトリに、どういう名前のファイルができたかをお知らせしてくれます。できるファイルの名前は、**index-j.html** です。外国語学部の言語学科のユーザ名 `atake` 君の場合には、index-j.html というファイルが、例えばですが、次のディレクトリにできたと教えてくれます (ユーザ名の `atake` の部分は、あなたのユーザ名に置き換えて下さいね)[5]。

```
/NF/local/general/WWW/http/people/ll-student/atake/index-j.html
```

`wwwmkdir` で作成したページは、一晩たたないと所定のページに登録されません。ですので、`wwwmkdir` を実行した直後は、あなたのページはどこからもリンクされていないので、Netscape などを使って京都産業大学のどこかのページからリンクして見ることはできません。しかし、安心して下さい。:-) 直接行き先を指定すれば良いのです。Netscape で見たい場合には、「`Open`」ボダンを押して、例えばですが、次のような URL を入れて下さい。

```
http://www.kyoto-su.ac.jp/people/ll-student/atake/index-j.html
```

どうですか？ あなたの名前が入ったページがきちんと表示されましたか？ これで、あなたもホームページのオーナーです！ :-)

1.3.2 Mule は使える？

さて、`wwwmkdir` が作ってくれた `index-j.html` というファイルの中身を見てみましょう。

まずは、そのディレクトリまで `cd` コマンドで移動しなくてはなりません。先ほどの言語学科の `atake` 君の場合には、現時点では次のコマンドで移動します[6]。

```
cc2000(86)% cd /NF/local/general/WWW/http/people/ll-student/atake
```

そして、それを Mule で読み込みましょう。読み込み方は、「Mule」の章を参照して下さい。

これで第 1 の準備は整いました。もっとも、この `index-j.html` ファイルも見ているだけでは足らなくなって、書き加えたりしたくなるかもしれませんね。書き加えていきましょう。しかし、後の 1.4で述べるように、具体的にホームページを書いていく段階になると、ある程度 Mule が使えないと書くのにものす

[5] `ll-student` などのディレクトリの名前や、ディレクトリの変更が行なわれた場合の新しいディレクトリ名などの詳しいことについては、先ほど紹介した `Howtomake_Yourpage.html` をご覧下さい。

[6] 長いコマンドですね。:-(毎回このようなコマンドを入力するのが面倒だと思われる方は、もし先ほどのディレクトリに `index-j.html` ファイルができているとすれば、

```
cc2000(84)% ln -s /NF/local/general/WWW/http/people/ll-student/atake ~/www
```

のように、シンボリックリンクを張っておくと、どのディレクトリにいても

```
cc2000(85)% cd ~/www
```

だけで移動できるようになるので便利です。このシンボリックリンクについては、この『コンピュータガイド』の巻末にある「FAQ」などをご覧下さい。

ごく時間がかかるかもしれません。ですので、「Mule」の章を見て、Mule を使いこなせるようにしておくと、ページ作りが楽になります。[7]。

1.3.3 Netscape などのブラウザは使える？

第 2 の準備は、Netscape などのブラウザを立ち上げて、ホームページを読み込むことです。そうしないと、本当に修正されているかどうかが分かりません。肝心の Netscape などのブラウザを使うことができますか？ Netscape であれば、cc2000 の UNIX 版でも、31 教室などの Mac 版でもどれでも良いですので、立ち上げておきましょう。立ち上げ方は、「WWW」の章を見て下さい。

1.3.4 忘れずに！ Mule で Save した後にブラウザで Reload

これで、準備は整いました。:-) Mule で `index-j.html` を読み込んで、Netscape などでホームページを読み込んでいますね？ 後は、この後に記す方法を色々使って、自分のプロフィールなどを書き足していけばよいわけです。

そして、Mule で書き込んだり修正したら、必ず `C-x C-s` で Save (セーブ、**「保存」**ですね) して、その後で Netscape などの「`Reload`」(**「再読み込み」**のことです) ボタンを押して、もう一度読み込み直します。そうすると、 Mule で変更した部分がきちんと変更されて表示されるはずです。これを繰り返すことになります。人によりますが、満足のいく出来上がりになるまで、結構何度もこの作業を繰り返すことになると思います。

ところが、こういうこともあります。Mule でファイルを変更したにも関わらず Save するのを忘れて、Netscape だけ一所懸命 Reload して、「あれ？ 変わってへん。何でや？」とつぶやいている人を見かけることが時々あります。しかし、Mule で Save しない限り、何度 Reload をかけてもいつまでたっても変わらないままでしょう。;-) ですから、**「Reload の前に、まず Save」**を忘れずに、ホームページを書いていきましょう。

1.4 これだけでホームページは書ける：基本編

これから、ホームページの基本的な書き方を記していきますが、大きく 2 つに分けてあります。「基本編」と「Netscape 編」です。「基本編」では、Mosaic をはじめどのブラウザでもきちんと表示される書き方を紹介します。はじめは、これだけで十分立派なホームページを作ることができるでしょう。次の「Netscape 編」では、Netscape だけで表示できる書き方を紹介します。

なお、表示のサンプルには、Mac 版の Netscape バージョン 2.0 の画面を使っています。Mac 版の Netscape といってもバージョン 1.1 の画面とは異なる場合がありますし、バージョン 2.0 といっても UNIX 版や Windows 版などの画面とは違う場合があることを、あらかじめお断りしておきます。

1.4.1 ファイル名の付け方

先ほど、`wwwmkdir` を実行した時に、`index-j.html` という名前のファイルができました。この最後に付いている `.html` というのは、おまけで付いているわけではありません。これは**拡張子**と言い、そのファイルがどういう種類のファイルであるかを示しています。`html` 自体の意味はもうお分かりですね？ そして、この `.html` が付いていることによって、Netscape などのブラウザはそのファイルが HTML のファイルで

[7] 必ずしも cc2000 の Mule を使わなければならないということはありません。自分で Mac とかを持っている人は、エディタで作ってテキストファイルにさえしておけば使い回しが効きます。しかし、Mac から cc2000 に送ったり、色々と変換したりする手間が必要ですので、京都産業大学では最初から cc2000 の Mule で書く方が楽でしょう。

あることが認識できるわけです[8]。ですので、新しいファイルを作りたい場合にも、例えば`newfile.html`のように必ず`.html`を付けて下さい[9]。

1.4.2 「タグ」って何？

`index-j.html`ファイルをMuleで読み込んで見てみると、`<HEAD>`のように`<`と`>`で囲まれたものがたくさん出てきます。このようなものを、**タグ (名札)** と言います。このタグは、Netscapeなどのブラウザで見ると表示されていません。表示されませんが、その代わりに「字を大きくせよ」「改行せよ」などの命令や様々な設定を行なう、いわば「黒子」の役割をしています。しかし、もしもこのタグが全くなければ、全部の文章が一続きに表示されてしまい、とても見にくくなるでしょう。そういう意味で、非常に重要な裏方さんなのです。そして、このタグをいかにうまく使いこなすかによって、あなたのページが見やすくも見にくくもなります。;-)

このタグには、2種類あります。1) ペアで使うタグと、2) 単独で使うタグです。ペアで使うタグとは、

```
<B>太字になる</B>
```

のように文字列を囲むタグです。ペアで使うタグは、`<atake>`で始まったら、`</atake>`のように斜め線(スラッシュ)付きのタグで終らなければなりません。そうしないで、`</atake>`を書き忘れてしまうと、最初のタグに含まれている命令がずっと続いて、変なことになってしまいますので、忘れないようにしましょう[10]。一方、単独で使うタグ(「空タグ」とも言います)とは、

```
<BR>
```

のようなタグです。そして、非常に大事なことは、

タグは半角 (つまり英数モード) で書かなければいけない！

ということです[11]。本文を日本語で(つまり全角で)書いている時にタグを入れようとして、そのまま日本語モードで入れると、＜ HR ＞のようなおかしなタグになってしまい(どこがおかしいか分かりますか？)、せっかく書いたつもりでもブラウザは言うことを聞いてくれません。本来裏方さんなのに、全部表示されてしまいます。**必ず半角 (英数モード) で入力しましょう！**[12]

1.4.3 絶対なければいけないタグ

さて、`index-j.html`ファイルをMuleで見てみると、Netscapeなどでは見えなかったタグが色々出てくることが分かります[13]。その中でも、以下に記してあるタグは、絶対に必要なタグです。ですから、消したりしないようにしましょう。まずは見てください。その後で、一つ一つ見ていきましょう。

---

[8] また、`.html`という拡張子を持ったファイルをMuleで読み込むと、1.6に記す **html-helper-mode** に自動的に入るので、それを使ってページを楽に書くことができるようになります。

[9] なお、MS-DOSのように拡張子が3文字に制限されているような場合には、`.htm`しか付けられませんが、それでもきちんと認識されます。しかし、cc環境でHTMLのファイルを作っているならば、`.html`を付けておきましょう。

[10] Muleでは、1.6に述べる **html-helper-mode** を使うと、タグの閉じ忘れをかなり防ぐことができます。:-)

[11] ただし、タグの中でも`"`と`"`の間には、日本語を入れることはあり得ます。

[12] 1.6に記す **html-helper-mode** を使うと、日本語モードに入っていても、ある特定のキー操作だけでタグが半角で入るので非常に便利です。:-)

[13] Netscapeなどでも、「`View`」メニューの中の「`Document Source`」を使えば別ですよ。;-) これなら、Muleで読み込んだ時と同じように見ることができます。ただし、編集はできませんけれども。

入　力

```
<HTML>
<HEAD>
<TITLE> This is my cool homepage :-) </TITLE>
</HEAD>
<BODY>
ここから本文。書きたいことはここに書く。
</BODY>
</HTML>
```

<HTML>と</HTML>

<HTML>と</HTML>は、ファイルの最初と最後に置きます。これは、このファイルがHTMLファイルであることを示すものです。

<HEAD>と</HEAD>

<HEAD>と</HEAD>で囲まれた部分は、本文には現れませんが、次に書くタイトルを表すタグを入れます。

<TITLE>と</TITLE>

<TITLE>と</TITLE>で囲まれた部分には、そのファイルのタイトルを書きます。ここで書かれた内容は、ウインドウの上の方にタイトルとして現れたり、ブックマーク (しおり) で登録するときの名前になったり、Netscapeなどでそのファイルを保存するときのタイトルになります。重要な部分ですので、分かりやすいタイトルを付けましょう。また、ブラウザによっては日本語に対応していなくて「文字化け」する場合がありますので、現時点では英数字を使う方が良いでしょう。

<BODY>と</BODY>

<BODY>と</BODY>の間には、本文を書きます。このタグの間以外には、本文はとりあえず書けないと思っておいた方が良いでしょう。

上に書いたタグは、絶対に必要なものですから、それらをあらかじめ書いたファイルを sample.html のように作っておいて、新しいファイルを作るときには、それをコピーして名前を変えて中身を書き換えるようにすると良いかもしれません。

このファイルをNetscapeで見ると、次のようになっています。上のソースのどの部分がどのように表示されているかを、図1.1で確認してみましょう。

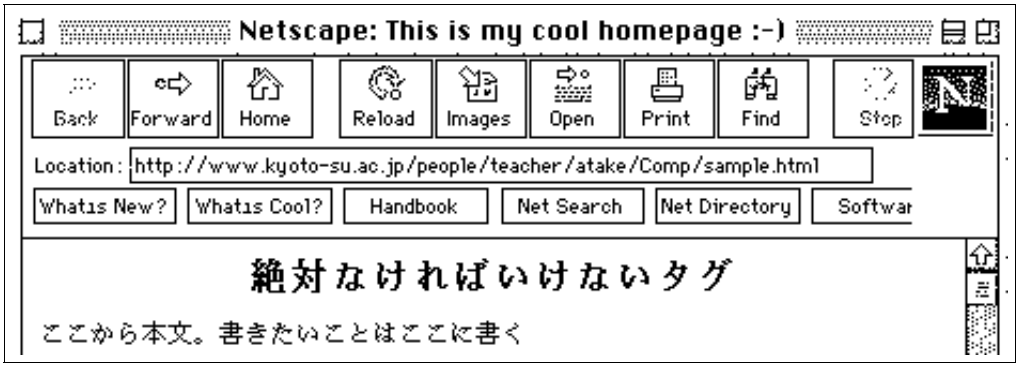


図 1.1 絶対なければいけないタグたちは、表示されない

<ADDRESS>と</ADDRESS>

その他に、そのページを作ったのが誰かということを入れておくタグも書いておいた方が良いでしょう。<ADDRESS>と</ADDRESS>の間に、次のように自分の名前や電子メイルのアドレスを入れておきます。これは、図 1.2のように自動的に斜体の文字になります。また、大抵は文書の最後に書きます。

入　力

```
<ADDRESS>竹内茂夫 (atake@cc.kyoto-su.ac.jp)</ADDRESS>
```

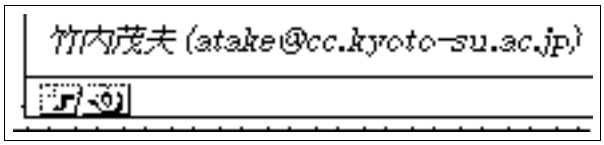


図 1.2 アドレスの表示

1.4.4 特殊文字：そのままでは出ない文字

HTML のソースを見ると、タグが出てきますが、それらは<と>に囲まれています。しかし、<と>で囲まれたものは、命令として解釈されるので、Netscape などのブラウザで見ても表示されません。

ですので、これらを表示させるためには、特別な書き方が必要です。このような特別な扱いをするいわゆる**特殊文字**には、<と>の他に、&と"があります。これらの特殊文字は、次のように書くことによって文字としてブラウザに表示されます[14]。

表示したい文字	:	書き方
<（小なり）	:	<
>（大なり）	:	>
&（アンパサンド）	:	&
"（二重引用符）	:	"

[14] しかしながら、ブラウザによっては文字にならずに書いたとおりに表示されます。例えば、&を表示するために&と書いても、&と表示されずにそのまま&と表示されるという悲しい事態になります。

1.4.5 コメント (書くけれども表示させない)

ホームページを書いていて、一旦書いたけれどもやっぱりやめたい … でも、せっかく書いたんだから残しておきたいと思うこともあるでしょう。そういう時には、<!--と-->で囲むことによって、ソースには書いているけれどもブラウザでは表示させないようにすること (コメントアウト) ができます[15]。

入　力

```
<!-- ここはコメントで表示されないから、何を書こうかな。;-) -->
```

1.4.6 改行する

HTML も LaTeX の場合と同様に、改行ははっきりと示してやらなければいけません。いくらソースファイルの方で何回も改行しても、Netscape などで見ると全く改行されていません[16]。改行は、
で示します。

入　力

```
ちょっと改行したいな。
よっこらしょっと。あれ？ これだと改行できない。:-( <BR>
ちょっと改行したいな。<BR>
よっこらしょっと。これで、改行できた。:-)
```

⇓

ちょっと改行したいな。よっこらしょっと。あれ？これだと改行できない。:-(
ちょっと改行したいな。
よっこらしょっと。これで、改行できた。:-)

図 1.3 改行する

1.4.7 段落を変える

先ほどの
では改行だけされました。では、改行してなおかつ 1 行空けることはできるのでしょうか。できます。その場合には、改行して空けたいところに<P>と書きます。

入　力

```
これで一行分空く。<P>
さあどうだ！
```

⇓

[15] ただし、LaTeX の%とは違って、万能ではありません。せっかく<!--でコメントアウトを始めても、その後に>があればそこでコメントが終ってしまい、その後は表示されてしまいます。その後もコメントアウトしたければ、再度<!--と-->で囲んでやらなければいけません。これは何とかして欲しいところです。:-(

[16] LaTeX ならば 1 行空ければ改行されるので、まだ何とかなります。:-p

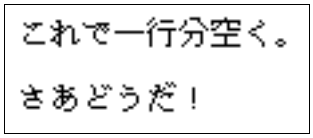


図 1.4 段落を変える

1.4.8 区切り線を入れる

段落と段落を分ける場合に、<P>を使えば 1 行分空きますが、さらにはっきりと分けるために、水平の区切り線を引くこともできます。そのためには、<HR>と書きます。

入　力

区切り線が引ける。
<HR>
なるほどなるほど。

⇓

区切り線が引ける。

なるほどなるほど。

図 1.5 区切り線を入れる

1.4.9 見出しを付ける[17]

文章を書いたら、大きく太い文字で何か見出しをつけたいですよね。また、段落に分けたときも、ちょっと大きな文字で段落のタイトルをつけたくなることもあると思います。逆に、ちょっと小さくしたいこともあるでしょう。そのためには、見出しの大きさを変えたい文字を<Hn>と</Hn>で囲みます。<Hn>の n には 1 から 6 までの数字が入ります。1 がもっとも大きく、6 がもっとも小さい見出しになります。

このタグは見出しのためのタグですので、文字の大きさと種類が一緒に変わってしまいます。また、このタグを使うと、前後の行に必ず空白が空きますので、1 行の中のある文字だけ大きさを変えるということはできません[18]。

[17] ただし、cc2000 の Netscape の日本語フォントでは正確に対応していない場合があります。

[18] ある特定の文字だけ大きさを変える方法もありますが、Netscape など一部のブラウザしか対応していません。Netscape での方法としては、大きさを変更したい文字を<FONT SIZE=n>と</FONT>で囲みます。n には 1 から 7 までの数字が入ります。7 が最も大きくて、1 が最も小さい文字です。注意しなければならないのは、<Hn>と</Hn>とは逆に**大きい数字の時に大きいフォントになる**ということです。

入　力

```
<H1>big な見出しから small な見出しまで</H1>
<H2>big な見出しから small な見出しまで</H2>
<H3>big な見出しから small な見出しまで</H3>
<H4>big な見出しから small な見出しまで</H4>
<H5>big な見出しから small な見出しまで</H5>
<H6>big な見出しから small な見出しまで</H6>
```

⇓

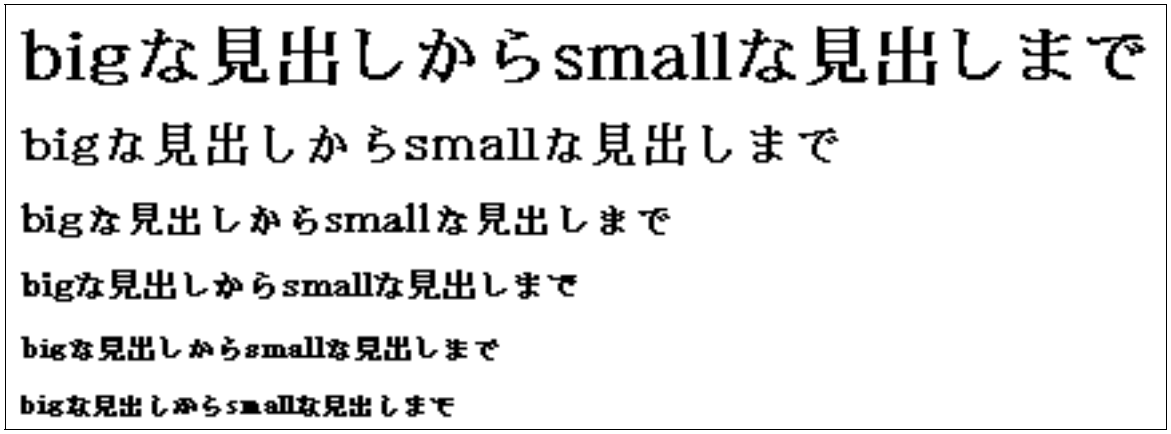


図 1.6　見出し

1.4.10　文字を修飾する[19]

<B>と</B>

文章を書いていると、文字を大きくしたりする変わりに、これは太くしたい、ということもあるのではないかと思います。HTML では、太くするには<B>と</B>で囲みます。

<I>と</I>

欧文の文章を読んでいると、強調部分や書名などが斜め (イタリック) になっていることがありますよね。斜めにするには<I>と</I>で囲みます。ただし、日本語で斜めにすると、結構見にくいものですので気をつけましょう。

<TT>と</TT>

さらに、タイプライターのような文字 (いわゆる「等幅フォント」) を使いたいこともありますよね。特に、プログラムのソースを載せたい場合には、これが必要でしょう。その場合には、<TT>と</TT>で囲みます。ただし、もしも 1 行を越えるような長い場合には、1.4.13で述べる方法を使ったほうが良いでしょう。

[19] ただし、cc2000 の Netscape の日本語フォントでは対応していない場合があります。

入　力

```
<B>太字 (bold) になる</B>
<I>斜め (italic) になる</I>
<TT>等幅 (typewriter) になる</TT>
```

⇓

太字(bold)になる

斜め(italic)になる

等幅(typewriter)になる

図 1.7 文字の修飾

1.4.11　リスト (箇条書き)

HTML でリストにする方法には、3 種類あります。すなわち、**1) 記号付きリスト、2) 番号付きリスト、3) 見出し付きリスト** です[20]。

記号付きリスト

記号付きのリストを作るには、<UL>と</UL>で囲み、それぞれの項目の前に<LI>を付けます。

入　力

```
記号付きリスト
<UL>
  <LI> First Call
  <LI> Glad
  <LI> 2nd Chapter of Acts
</UL>
```

⇓

- Fitst Call
- Glad
- 2nd Chapter of Acts

図 1.8 記号付きリスト

[20] それぞれ、LaTeX の **1) itemize 環境、2) enumerate 環境、3) description 環境**に対応すると考えればよいでしょう。

番号付きリスト

番号付きのリストを作るには、<OL>と</OL>で囲み、それぞれの項目の前に<LI>を付けます。

入　力

```
番号付きリスト
<OL>
  <LI> David Grisman
  <LI> Tony Rice
  <LI> Doc Watson
</OL>
```

⇓

図 1.9 番号付きリスト

見出し付きリスト

見出し付きのリストを作るには、<DL>と</DL>で囲み、見出しの前に<DT>を、説明の前に<DD>を付けます。それぞれの項目の前に<LI>を付けます。

入　力

```
見出し付きリスト
<DL>
  <DT> First Call
  <DD> CCM 界のコーラスグループの第一人者。ロックからジャズコーラスまでこなす幅広い音楽性が魅力。代表作は、Stevie Wonder の"Don't You Worry 'Bout A Thing"などを収録した<I>Human Song</I> (1992)。
  <DT> Glad
  <DD> 複雑なアカペラもこなす TAKE6 と並ぶ CCM 界の有数のコーラスグループ。古典的な hymn を現代的な厚いアカペラにアレンジするその手腕は見事。代表作は、<I>THE ACAPPELLA PROJECT</I> (1988)。
</DL>
```

⇓

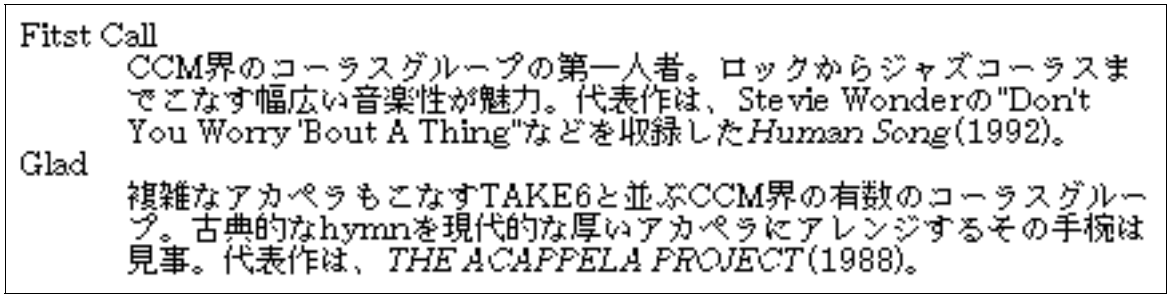


図 1.10 見出し付きリスト

リストの入れ子

これらのリストを組み合わせて、入れ子にすることもできます。

入　力

```
<OL>
  <LI> CCM
       <DL>
         <DT> First Call
         <DD> CCM 界のコーラスグループの第一人者。
         <DT> Glad
         <DD> 複雑なアカペラもこなす CCM 界の有数のコーラスグループ。
       </DL>
  <LI> Bluegrass
       <UL>
         <LI> David Grisman
         <LI> Tony Rice
         <LI> Doc Watson
       </UL>
</OL>
```

⇓

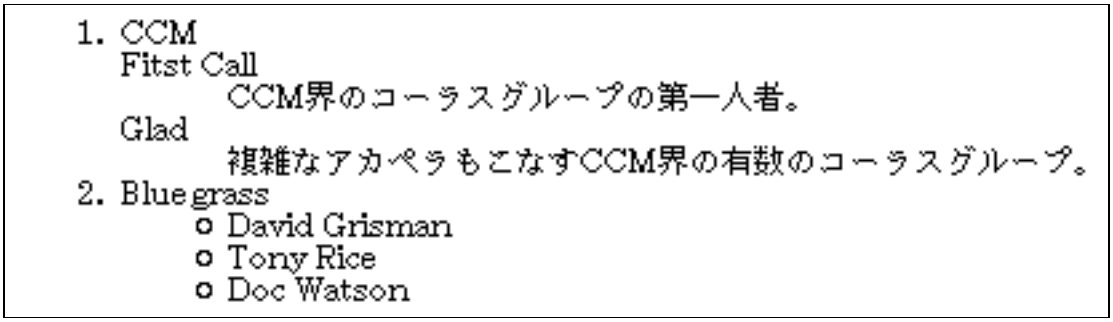


図 1.11 リストの入れ子

1.4.12 字下げ（インデント）する

本などから引用する場合、行を少しだけ頭とお尻を引っ込ませて、真ん中に集まるような形になっていることがあります。このように、行の頭を引っ込ませることを「インデント」とよびます。インデントさせたい部分を<BLOCKQUOTE>と</BLOCKQUOTE>で囲みます。

入　力

```
これが普通。
<BLOCKQUOTE>
こうすると、行の頭とお尻が引っ込む。
</BLOCKQUOTE>
```

⇓

これが普通。

こうすると、頭とお尻が引っ込む。

図 1.12 字下げ (インデント) する

1.4.13 書いたものをそのまま表示

どのブラウザでもきちんと表示される表や、コンピュータのプログラムなどを表示したい場合には、そのまま書いただけではいくら改行したり空白を空けてもうまくいかないことが多いでしょう。また、フォントもタイプライターのような等幅フォントでないと都合が悪いでしょう。そういう場合には、書いたものをそのまま表示してくれる<PRE>と</PRE>で囲みます。このタグで囲まれると、等幅フォントが使われて、空白や改行がそのまま表示されます。

ただし、気を付けなければいけないのは、いくらそのまま表示されるとはいっても、その中に例えば<P>のようにタグを入れると、**それがそのまま表示されずに、タグと解釈される**ということです[21]。<P>の場合

[21] この点、LaTeX の **verbatim 環境**は、ほとんどそのままを表示してくれます。

でしたら、改行して 1 行空けてしまいます。もし、<PRE>と</PRE>の中で、例えば<P>を表示したければ、1.4.4に記した方法で書く必要があります。つまり、<P>ならば、

入　力

```
&lt;P&gt;
```

と書かなければなりません[22]。

1.4.14　画像を入れよう (インライン画像)

ホームページを作っていると、文字だけではちょっと寂しくなってくると思います。「自分の写真を入れたい！」と思うこともあるでしょう。自作の絵を入れたいと思うこともあるでしょう。最近のホームページでは、論文などだけを載せているのでない限り、ほとんど絵とか写真が入って華やかになっているものです[23]。あなたも、入れてみましょう。:-)

ホームページを表示する時に同時に読み込まれる画像のことを、**「インライン画像」**と言います。ここでは、このインライン画像の入れ方について見てみましょう。

画像を用意する

まずは、ホームページに入れる画像を用意しなければなりません。それについては、ここで詳しく説明する余裕は残念ながらありません。例えば、写真や絵をスキャナと呼ばれる機械で取り込んだり、デジタルカメラで撮った写真を取り込んだり、コンピュータを使って書いた絵を使ったり[24]、という様々な方法が考えられます。ここでは、そうした画像は既に用意されているという前提で話を進めます。

大切なことは、入れたい画像は **GIF** (じふ) というフォーマットで保存されていて、ファイル名の最後に.gif という拡張子を持っていなければならない、ということです[25]。そして、画像のサイズはあまり大きくしないようにしましょう。大きな画像はどうしてもファイルサイズも大きくなりがちで、表示されるのに時間がかかるようになってしまいます。その結果、その画像が入ったページへのアクセスが敬遠されることにもなりかねませんので、気をつけましょう。

タグの書き方は?

このような GIF フォーマットの画像を、自分のホームページがあるディレクトリに置きます。仮に、ファイル名を mystudio.gif としておきます。そして、ホームページの中で画像を入れたいところを決めておきます。なお、画像も文字と同じように扱われますので、改行などを上手に使いましょう。

タグの書き方は、<IMG SRC="○" ALT="□">です。○の部分には、先ほどの画像のファイル名の mystudio.gif を入れます。□の部分には、その画像を表すタイトルのようなものを入れておきます。これは、もし Netscape などでも何かの事情で画像が表示できなかったり、画像を表示しないブラウザを使った時に、それがどういう画像なのかを文字で表示してくれるようにするためです。全体では、次のようになるでしょう。

入　力

```
<IMG SRC="mystudio.gif" ALT="私のスタジオ">
```

[22] ちょっと面倒ですね。:-(しかし、cc2000 の Mule なら、1.6に記す **html-helper-mode** を使うことによって、簡単に入力できます。例えば、& を入力するのは、C-c & でできます。

[23] 論文しか書かれていないページですらも、どこかに画像が入っていたりします。:-p

[24] 例えば、cc2000 の X 環境ならば、xv や xpaint を使うことができます。使い方は、「xv」や「xpaint」の章をご覧下さい。

[25] 少し前までは、インライン画像で使えるのは GIF ファイルだけでしたが、256 色までしか使えないという制約がありました。最近では **JPEG** (じぇいぺぐ) と呼ばれる 256 色以上使えるより高品位の画像も、使えるようになってきました (拡張子はに .jpeg または .jpg などが付いています)。しかし、色数が多ければファイルサイズは大きくなるので、注意しましょう。ホームページに入れる画像の色数としては、256 色かそれ以下でも十分です。白黒も、以外と効果的ですよ。

図 1.13のように画像が表示されれば、問題はありません。

図 1.13 インライン画像 (本当はもっときれいに表示されます) :-p

もし、画像が表示されない場合には、図 1.14のように ALT="□"に書いた文字が表示されます。

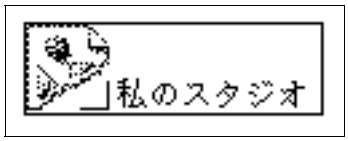


図 1.14 画像が表示されない時

画像がホームページと違うディレクトリにある時には、ディレクトリ名を画像ファイル名の前に付けなければなりません。詳しくは、後の 1.4.15 で述べるリンクの張りかたと同じように、ディレクトリを指定します。仮に、ホームページと同じディレクトリにある photos というディレクトリの中にあるとすれば、

入　力

```
<IMG SRC="photos/mystudio.gif" ALT="私のスタジオ">
```

のように書きます。

1.4.15　リンクを張ろう

この**リンク**こそが、HTML の本領です！ 何と言っても、リンクできるからこそ「ハイパーなテキスト」なんですから。これを使うことによって、世界のどこのホームページにも一瞬で[26]つながります。要するに、自分用の「どこでもドア」みたいなものですね。:-)

リンクを張るためには、リンク先のファイル名やディレクトリ名やサイト名などの URL が分かっていなければいけません。タグは、<A HREF="*"> と</A>というのを使います (A は**「アンカー (錨)」**の略です)。*には、リンク先の URL が入ります。<A HREF="*">と</A>の間には、そのページの名前などを入れます。これがないと、リンクするポイントができないので、忘れないようにしましょう！

リンクは、ブラウザで見ているときにはその部分をクリックするだけで良いので、ピンと来ないかも知れません。けれども、いざ書く時にはいくつか種類があったりします。ここでは、**1) 別のページにリンク**

[26] というのは大げさですが。;-) 最近の WWW ユーザの飛躍的な増加に回線増設がなかなか追いつかないこともあって、以前ならすぐにつながるページでも、時間がかかるようになっています。

を張る、2) 同じページの中にリンクを張る、3) 別のサイトにリンクを張る方法について説明しましょう。

別のページにリンク

この場合、2 種類考えられるでしょう。1) 同じディレクトリにあるファイルにリンクを張る場合と、2) 別のディレクトリにあるファイルにリンクを張る場合です。このあたりのことをするためには、ディレクトリのことがある程度分かっていないと、しんどいかも知れません。分からなければ、この『コンピュータガイド』の「ファイルの階層構造」の部分などを読みましょう。

同じディレクトリにあるファイルにリンクを張るのは、簡単です。次のようにファイル名を指定するだけで OK です。

入　力

```
<A HREF="atake.html">あたけ君のページ</A>
```

このように書くと、ブラウザで表示される時には、**通常色が変えられて (デフォルトは青です)、下線が引かれることもあり**[27]**、マウスの矢印 (カーソル) を合わせるとウインドウの下の部分に URL が出てきて、リンクしていることを示すようになります。**

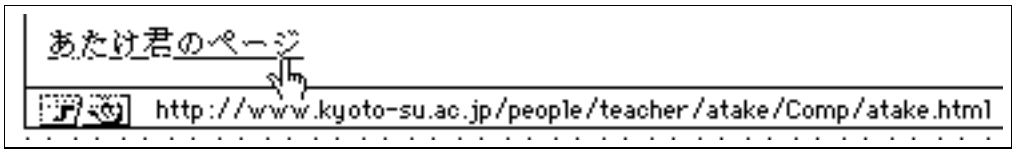


図 1.15　リンクのタグが付けられた文字列

もし、別のディレクトリにあるファイルにリンクを張りたい場合には、ディレクトリ名を付け加えれば OK です。例えば、ホームページのファイルがあるディレクトリに music というディレクトリがあって、その music ディレクトリの中の favorite.html というファイルにリンクしたいのであれば、次のように書きます。

入　力

```
<A HREF="music/favorite.html">お気に入りの音楽</A>
```

また、リンク先にはホームページ以外のものを指定することもできます。例えば、ブラウザで画面のある部分をクリックしたら、画像が出てくるようにすることもできます。例えば、画像ファイルで atake.gif にリンクを張るのであれば、

```
<A HREF="atake.gif">あたけ君の写真</A>
```

のように指定しておけば、「あたけ君の写真」という文字列にリンクのポイントができますので、ブラウザでそこをクリックした時に atake.gif の画像を表示させることができます。音声ファイルや動画ファイルなども同じように指定することができます[28]。

[27] Mac 版の Netscape では、基本的には下線が付きません。
[28] 音声ファイルや動画ファイルの扱いについては、巻末の HTML の参考文献をご覧下さい。

同じページの中にリンク

同じページの中の別のところにリンクを張るには、ちょっと工夫が必要です。まず、リンク先にあらかじめ印を付けておかなければなりません。例えば、そのファイルの中の「CCM」という文字列にリンクを張りたいのであれば、それを<A NAME="*">と</A>で囲みます。そして、*には何か名前をつけておきます。なお、このタグは<A NAME="*">で、<A HREF="*">とは微妙に違いますので、注意して下さい。

そして、リンク元では、<A NAME="*">の*であらかじめ指定した名前を、<A HREF="*">の*に、頭に#を付けて書きます。そのようにして、リンク先を指定します。

入　力

```
<A NAME="ccm">ここが CCM のコーナー</A>です。<P>
<A HREF="#ccm">CCM のコーナー</A>にリンクします。
```

⇓

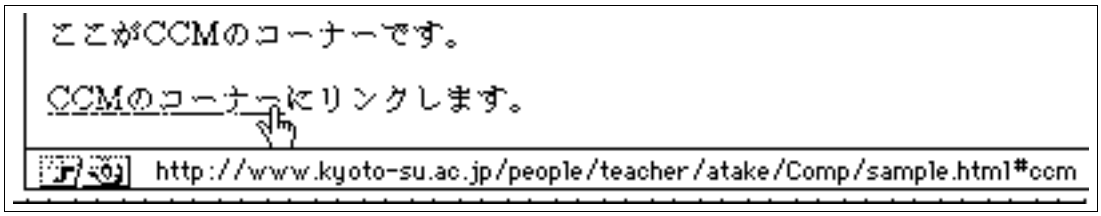


図 1.16 同じページの中でリンク

これで、色が変わっていたり下線が引かれている「CCM のコーナー」の部分をクリックすれば、上の「CCM のコーナー」にジャンプします。

繰り返しますが、<A NAME="*">と</A>であらかじめリンク先を作っておかないと、同じページ内のリンクは機能しないので気をつけましょう[29]。

別のサイトにリンク

別のサイトにリンクする場合には、<A HREF="*">の*に URL を入れます。例えば、ホワイトハウスの URL は http://www.whitehouse.gov/なので、これを*に入れて、<A HREF="*">と</A>の間に何か文字などを入れます。

入　力

```
<A HREF="http://www.whitehouse.gov/">ホワイトハウス</A>
```

また、ここでもリンク先はホームページに限りません。よく見られるのは、様々なファイルが蓄積されている「FTP サイト」にリンクが張ってある場合でしょう。例えば、京都産業大学の FTP サイトにリンクを張るのであれば、次のようにタグを書きます。

入　力

```
<A HREF="ftp://ccftp.kyoto-su.ac.jp">京都産業大学の FTP サイト</A>
```

[29] 例外として、そのページの一番先頭に戻るようにしてくれる<A HREF="#top"></A>というタグは、<A NAME=""></A>のタグでリンク先を指定しなくても機能します。

あとは、これらを組み合わせれば、世界中のどのページともリンクすることができます！ よく行くページのリンクを集めた「リンクのページ」を作っている人も多いです[30]。しかし、リンクのページはいわば借り物を集めたページなので、それ以外にも自分のオリジナルのページを作りましょうね。:-)

♣　♠　♣

これで、基本編は終りです。ともかく、色々とトライしてみましょう。:-)

[30] リンク集のページを見ると、その人がどういうことに興味があるのかが分かってなかなか面白いものです。

1.5 ちょっとカッコ良く：Netscape 編

これまでの「基本編」では、どのブラウザでもほとんど問題なく表示できる基本的なタグを紹介しました。しかしながら、今では全世界で使われているブラウザのうち、約 80％が Netscape だとも言われており、事実上のスタンダードとされています。そのために、ホームページにも「Netscape 対応」と書いているページも非常に多くあります。最近 Netscape2.0 の正式なバージョンがリリースされてさらに機能が拡張し、Netscape 以外のブラウザとの機能の差がより大きくなりました。

cc2000 には早くから Mosaic がインストールされていましたが、Mosaic では Netscape 対応のページをうまく見ることができない部分がありました。Netscape 独自のタグは、対応していないブラウザでは基本的に無視してくれるのですが、タグによっては最悪の場合それに対応していない Mosaic などのブラウザが強制的に終了してしまうこともあります[31] 。今では cc2000 にも Netscape2.0 がインストールされて、Netscape 対応のページも問題なく見ることができるようになりましたので、採り入れても良いでしょう[32]。

Netscape では、様々な機能が拡張されています。Netscape が独自に拡張した HTML は、**「拡張 HTML」**と言われることがあります。「拡張 HTML」には様々な機能が含まれますが、その中で Netscape 以外のブラウザで見ても、大きく影響を与えないような代表的な機能をいくつか紹介しましょう (と言いつつ、表組は他のブラウザではうまく見えないことがあります)[33]。なお、基本的には、cc2000 にもインストールされているバージョン 2.0 に対応した書き方をしています。

ただし、以下の **Netscape 対応のタグを使った時には、必ず Mosaic などの Netscape 以外のブラウザでも出来上がりをチェックするようにして下さい！** というのは、せっかく Netscape ではきれいに表示できても、Mosaic で見たらガタガタでカッコ悪くてしょうがなかった、ということがあるからです (特に、Netscape 対応の表を作った場合にそうなりやすいです)。もし、どうしても Netscape でないと適切に見ることができないページならば、そのページの中に「Netscape2.0 でご覧下さい」ということを書いておくと親切でしょう[34]。:-)

1.5.1 真ん中寄せ、右寄せにする

Netscape 以前には、ソースを書くと基本的にすべて左寄せになってしまいました[35]。しかし、見出しなどは、行の中央にしたいとか、名前などは右に寄せたいとかいうこともあると思います。その時には、次のようにします。

<CENTER>と</CENTER>

Netscape では、文字を真ん中に寄せるには、

入　力

```
<CENTER>真ん中寄せにする</CENTER>
```

のように書きます[36]。

---

[31] 業界用語で「死ぬ」「落ちる」などと言います。:-p

[32] 31 と 11 教室および図書館の Mac には、いち早く Netscape がインストールされていたので、Netscape 対応のページを見ることができました。なお、cc2000 の Netscape2.0 ならば、さらにあの JAVA の applet も見ることができます。:-)

[33] その他の「拡張 HTML」などの機能については、Netscape を立ち上げたときに現れる Handbook のボタンを押して必要なところを見るか、巻末の HTML 関係の参考文献を見てください。

[34] しかし、どのブラウザでもそれなりに見えるというのが一番良いのですけれども、なかなか難しいですね。

[35] もっとも、空白の入ったテキストを<PRE>と</PRE> で囲むとか、全角の空白をたくさん入れて、無理矢理真ん中寄せらしくしたり右寄せらしくするという涙ぐましい努力もありましたが、幅の狭いディスプレイでは右側が切れてしまう危険性もありました。

[36] 真ん中寄せには、次の右寄せと同じような<DIV ALIGN="center">と</DIV>で囲む書き方もあります。しかしながら、Netscape でもバージョン 1.1 では、こちらのタグは対応していないので、左寄せになってしまいます。

<DIV ALIGN="right">と</DIV>

文字列を右寄せにしたい場合には、

入　力

```
<DIV ALIGN="right">右寄せにする</DIV>
```

のように書きます。

⇓

真ん中寄せにする

右寄せにする

図 1.17 真ん中寄せと右寄せ

1.5.2　文字を点滅させる

ホームページの中で文字がチカチカと点滅しているのを見たことがある人もいるでしょう。これは、点滅させたい文字を<BLINK>と</BLINK>で囲みます。ただし、UNIX の Netscape では文字が点滅しますが、Mac の Netscape では背景が点滅します。

入　力

```
<BLINK>こうすると点滅する</BLINK>
```

⇓

こうすると点滅する　⇔　こうすると点滅する

図 1.18 文字の点滅

1.5.3　背景の色を変える

Netscape や Mosaic などのブラウザでは、背景の色は基本的にグレイです。しかしながら、Netscape では、例えば白に変えることによって、全体を明るい感じにして、さらに文字とのコントラストをはっきりさせることができます。

背景の色を変更するのは、ソースの最初の方に現れる<BODY>タグの中に少し書き足すことによって可能になります。書き方は、<BODY BGCOLOR="#??????">です。??????の部分は左から順に 2 桁ずつ、赤、緑、青の度合いを 16 進法 (0 から F まで) を使って指定します。例えば、赤の場合ならば、000000 では赤の度

合がゼロで(この場合は黒になります)、FF0000 で真っ赤っ赤になります。中間の 800000 だと、ちょっと暗い赤になります。もう一つの例として、背景を白にしたい場合には、次のように書きます[37]。

入　力

```
<BODY BGCOLOR="#FFFFFF">
```

1.5.4　文字の色を変える

Netscape2.0 では、文字の色を自由に変えることもできます。色を変えたい文字を<FONT COLOR="#??????">と</FONT>で囲みます。これも 16 進法で指定しなければなりませんので、代表的なものだけを記しておきます。

気を付けておかなければいけないのは、背景の色とのコントラストです。背景と文字を似たような色にしてしまうと、非常に見にくいページになってしまいますので、気を付けましょう[38]。

また、1 字ごとに変えたりすると、検索するときにうまくいかなくなりますから、特殊な効果を狙わない限りやめておいた方が無難でしょう。

入　力

```
<FONT COLOR="#000000">黒</FONT>
<FONT COLOR="#FF0000">赤</FONT>
<FONT COLOR="#00FF00">緑</FONT>
<FONT COLOR="#0000FF">青</FONT>
```

1.5.5　簡単な表を作る

Netscape では、表を作ることができます。それも、かなり複雑な表ができます。しかし、かなり面倒でもあります。ここでは、簡単な表を作ってみますが、それでもいくつかのタグが必要になります。

<TABLE>と</TABLE>

まず、表全体は、<TABLE BORDER>と</TABLE>で囲みます。<TABLE>タグの中の BORDER は、枠線を引くための命令です。枠線が必要なければ、<TABLE>だけで OK です。

<TR>と</TR>

<TR>と</TR>の間には、横 1 列に入る要素を書きます。

<TD>と</TD>

<TD>と</TD>の間には、個々のマス(「セル」と言います)に入る要素を書きます。

<TH>と</TH>

<TH>と</TH>の間には、個々のセルに入る要素を書きますが、見出し(ヘッダ)になりますので、太字(bold)になって、真ん中寄せになります。

[37] 16 進法で書くのは、ちょっと面倒なのも事実です。裏ワザとして、色の名前を書いて指定することもできる場合もあります。例えば、背景を白にしたいならば<BODY BGCOLOR="white">と書くこともできます。ただし、主な色しか対応していなかったり、色名では認識しないブラウザもあるので、16 進法で書く方が安全でしょう。

[38] また、背景を黒にして文字を白に指定してしまうと、ブラウザによっては文字が全く見えなくなってしまいますので、文字を白に指定するのはやめた方が良いでしょう。

<CAPTION>と</CAPTION>

<CAPTION>と</CAPTION>の間には、表の表題を書きます。何も指定しないと（つまり、「デフォルト」では）表の上に、真ん中寄せで表示されます。

ここまでで、基本的な表が作れるようになります。非常に簡単な時間割を作ってみましょう。

入　力

```
<TABLE BORDER>
<TR>
<TH></TH> <TH>月</TH> <TH>火</TH>
</TR>
<TR>
<TH>2</TH> <TD>ヘブライ語 I</TD> <TD>言語学セミナー A</TD>
</TR>
</TABLE>
<CAPTION>簡単な時間割</CAPTION>
```

⇓

図 1.19 Netscape の表

もう少し複雑な時間割を作ってみましょう。

入　力

```
<TABLE BORDER>
<TR>
<TH></TH> <TH>月</TH> <TH>火</TH> <TH>水</TH> <TH>木</TH> <TH>金</TH> </TR>
<TR>
<TH>２</TH> <TD>ヘブライ語 I</TD> <TD><BLINK>言語学セミナーA</BLINK></TD> <TD>卒論指導</TD> <TD>卒論指導</TD> <TD></TD>
</TR>
<TR>
<TH>３</TH> <TD></TD> <TD>言語学演習 B</TD> <TD></TD> <TD>卒論指導</TD> <TD></TD>
</TR>
<TR>
<TH>４</TH> <TD>言語学原論</TD> <TD></TD> <TD></TD> <TD></TD> <TD>ドイツ語 I</TD>
</TR>
<TR>
<TH>５</TH> <TD></TD> <TD>ヘブライ語 II</TD> <TD></TD> <TD></TD> <TD></TD>
</TR>
</TABLE>
<CAPTION>ある先生のある年度の時間割 ;-)</CAPTION>
```

⇓

ある先生のある年度の時間割 ;-)

	月	火	水	木	金
2	ヘブライ語I	言語学セミナーA	卒論指導	卒論指導	
3		言語学演習B		卒論指導	
4	言語学原論				ドイツ語I
5		ヘブライ語II			

図 1.20 ちょっと複雑な表

Netscape で表を書くときには、いかに見やすくソースを書くかによって、修正がしやすくもしにくくもなります。最初はなかなかうまくいかないものです。焦らずに、根気よくトライしましょう。:-)

1.6 HTML を書くお助けツール：html-helper-mode

さて、これまで HTML の書き方を説明してきましたが、注のところでたびたび「**html-helper-mode** だと便利やでー」というようなことを書いてきました。そうなんです。このような便利なものが、cc2000 の Mule には備えられているのです。例えば、リンクを張りたい時に、<A HREF="http://www.whitehouse.gov/">ホワイトハウス</A>というのを間違いなく書くのは、なかなか神経を使いますよね。ところが、html-helper-mode

を使えば、C-c C-a l[39]と押すだけで、自動的に<A HREF=""></A>を挿入してくれます。それだけではなくて、カーソルも２番目の"の上に置かれるので、すぐにURLを書き込むことができます。

html-helper-modeは、基本的なHTMLには対応しています[40]。基本的とは言っても、実はここで紹介したよりもはるかに多くのタグがあります。そして、どのタグがどのキーに割り当てられているかを覚えるのはちょっと大変かも知れません。「こんなん覚えるのはかなん」というあなたは、京都産業大学のホームページからたどれる「各種ドキュメント」の中に「html-helper-modeのキー割り当て一覧」が載せられていますので、それを見るなり印刷して持ち歩くなりすると良いでしょう[41]。

次の表1.1では、そのキー一覧割り当て表の中から、ここで紹介したタグのキー割り当てを記します(元の表の中の項目とは順番を変えて、説明を書き足しています)。

1.7 おわりに

これで、ホームページを作るためのHTMLの基本的な書き方をほぼ一通りの説明してきました。もちろん、これは網羅的ではないので、あの話やあの話を書かないのは何事か！ と思われる方もいらっしゃるかも知れませんが、HTMLの世界は日進月歩です。ですので、これ以上のことについては、

http://www.kyoto-su.ac.jp/information/

の「HTML関係」にもいくつか役に立つ情報が載っていますので、ぜひ見て下さい。それでも物足りない人は、巻末のHTMLの参考文献を見て、あなたのホームページをどんどん改良して下さい。

あとは、魅力的なページになって、たくさんの人があなたのホームページを訪れるようになるかどうかは、あなたのセンス次第です。ただ、繰り返しになるかもしれませんが、次のいくつかのことだけは心に留めておいて欲しいのです。

全世界の人に見られているんだよ

ホームページを作るということは、全世界のインターネットのユーザに対してあなたのページを公開するということです。ですので、世界の誰に見られても良い内容にしましょう。全世界に公開されていることから、日本語のページだけではなくて英語のページも作っておくと喜ばれるでしょう[42]。

見やすくしようね

誰が見ても、見やすいページにすると良いでしょう。そのためには、<P>や
などでうまく改行したり、<UL></UL>などのリストをうまく活用しましょう。また、</H2>などの/がついたタグを書かないという**タグの閉じ忘れ**でその後の部分の表示がおかしくなることが多いので、チェックしましょう。

どのブラウザでもある程度見れるようにしようね

巷ではNetscape全盛で、Netscape対応のページが多いのは事実です。仮にそのようなページを作ったとしても、Netscape以外のブラウザで見てもおかしくないように、複数のブラウザでチェックすると良いでしょう[43]。cc2000のX環境であれば、Netscapeの他に、Mosaicなども使ってチェックすることができます。

---

[39] アルファベットのLの小文字です。

[40] 残念ながら、96年3月時点ではNetscapeの拡張HTMLのタグには対応していません。文字を点滅させたい人などは、自分でガリガリと書いて下さい。

[41] URLは http://www.kyoto-su.ac.jp/people/graduate/matsuura/html-helper-mode/keybindings.pln です。

[42] ちなみに、wwwmkdirを実行してできるindex-j.htmlのjは、japaneseのjで、「日本語ページ」という意味です。英語ページを作る場合には、-jをはずしてindex.htmlなどとすると分かりやすいでしょう。

[43] Netscape自体でも、1.1と2.0では、対応するタグが異なっているので、違うように表示されることがあります。

基本キー	機能	押すキー	挿入されるタグ
C-c C-b	head element(begin の b)	C-c C-b t	<TITLE></TITLE>
	特殊文字	C-c >	>
		C-c <	<
		C-c &	&
	基本	M-TAB	タグの補完
		M-RET	<P>
		C-c RET	
		C-c -	<HR>
		M-C-t	<!-- HHMTS START -->
			セーブ時に timestamp が入ります
			<!-- HHMTS END -->
C-c C-t	ヘッダ (title の t)	C-c C-t 1	<H1></H1>
		C-c C-t 2	<H2></H2>
		C-c C-t 3	<H3></H3>
		C-c C-t 4	<H4></H4>
		C-c C-t 5	<H5></H5>
		C-c C-t 6	<H6></H6>
C-c C-p	文字の修飾 (物理的指定) :	C-c C-p b	<B></B>
	physical style (physical の p)	C-c C-p i	<I></I>
		C-c C-p f	<TT></TT>
C-c C-s	文字の修飾 (論理的指定) :	C-c C-s a	<ADDRESS></ADDRESS>
	logical style (style の s)	C-c C-s b	<BLOCKQUOTE></BLOCKQUOTE>
		C-c C-s p	<PRE></PRE>
C-c C-l	リスト (list の l)	C-c C-l d	<DL><DT><DD></DL>
		C-c C-l u	<UL><LI></UL>
		C-c C-l o	<OL><LI></OL>
		C-c C-l i	<LI>か<DT><DD>の適当と思われる方
		C-c C-l t	<DT><DD>
		C-c C-l l	<LI>
C-c C-i	イメージ (image の i)	C-c C-i t	<IMG ALT="" SRC="">
	C-i は Tab キーに相当		
C-c C-a	アンカー (anchor の a)	C-c C-a l	<A HREF=""></A>
		C-c C-a n	<A NAME=""></A>

表 1.1 本文で紹介したタグと html-helper-mode のキーバインディング

また、**ディスプレイの大きさによって、全く違うふうに見えるものです**。特に、21 教室のワークステーションを使って大きなディスプレイで丁度うまく見えるように作られたページは、例えば 11 教室の Mac の小さめのディスプレイではそれほどうまく見えなかったりします。ですので、大きさの違うディスプレイを使ってチェックすると良いでしょう。

人の素敵なページを参考にしようね

見やすいページといっても、最初はどういうふうに書いたら良いか分からないかも知れません。ですので、「あっ、これいいなぁ」と思ったページに出会ったら、そのソースを見てどのように書いてあるかを調べて、それを参考にすると良いでしょう[44]。

しかし、いくら参考にするといっても、そのままコピーしたものを使うのではなくて、あなた流のアレンジを加えて取り入れるようにしましょう。そうでないと、ただの「真似しぃ」になってしまいます。:-(

著作権だけは気を付けてね

参考にしたときにはアレンジを、と書きました。また、最初にも書きましたが、**自分のオリジナルな情報で勝負して下さい！** このことは、本当にお願いしておきます。くれぐれも、雑誌などから写真やアニメ、市販されている CD などからの音、ビデオからの画像などをあなたのホームページに入れることはやめて下さい！ 繰り返しますが、もしそういうページが京都産業大学の中で見つかった場合には、警告が与えられて、それでも改善が見られなければ、削除される可能性があります。しかし、自分のオリジナルな情報は、どんどん発信して下さいね。

♡　　♢　　♡

これで、HTML 入門はおしまいです。京都産業大学の中で魅力的なホームページがますます増えることを期待しています。:-)

[44] 個人的には、Netscape の表やフレームや JAVA の applet がそうでした。

第 2 章

LaTeX

2.1 LaTeX (らてふ)って何？

2.1.1 LaTeX とは？

今までワープロを使っていると、数学の式で、複雑なものはできませんでした。ところが、LaTeX は、それをきれいな形で出力できます。また、論文や、レポートなどにも手軽に使えます。ただ、ルールがあって、ワープロのようにすぐに使えるわけではありません。また普通のワープロに比べて自由度が大きく、さまざまなことが可能で、絵や図も簡単に文書中に埋め込むことができます。

レポートや卒論をワープロで書こうと思っている諸君、LaTeX を使って書いてみませんか。

2.1.2 LaTeX の特徴

LaTeX の特徴をいくつかあげておきます。

1. 複雑な数式をきれいな形で出力します。したがって、理科系の人にとっては必需品です。(論文など章節のはっきりした構造のある文章を作るのに向いている。)
2. お絵書きツールなどで作った図を簡単にはめ込むことができます。
3. 目次・索引・文献リストを簡単に作ることができます。

2.1.3 LaTeX の作業の進めかた

大まかに言うと次のようにして進めていきます。

1. エディタ (`Mule` など) で文書ファイル[1]を作成する。
2. 作成したファイルを LaTeX で処理 (コンパイル) する。
3. `xdvi` コマンドで、コンパイルされたものを見る。
4. 処理したファイルをプリントアウトする。

[1] NeXT にある文机や Mac の EG-WORD、solo-writer や一太郎などで書いたものは使えません。

2.2 それでは使ってみましょう

ここではDEC-3300からcc2000にインストールされているLaTeXを使う方法を紹介します。簡単な例を示しながら進めていきます。

まず、メニューのところからcc2000のktermを選択し、mule first.tex &<return>と入力しMuleを立ち上げ、LaTeXのソースファイルを作ります。(ここではDEC-3300、即ち X 環境でMuleを利用する為にコマンドの最後に & を付けています。非 X 環境でMuleを利用する場合はこの & は必要ありません。

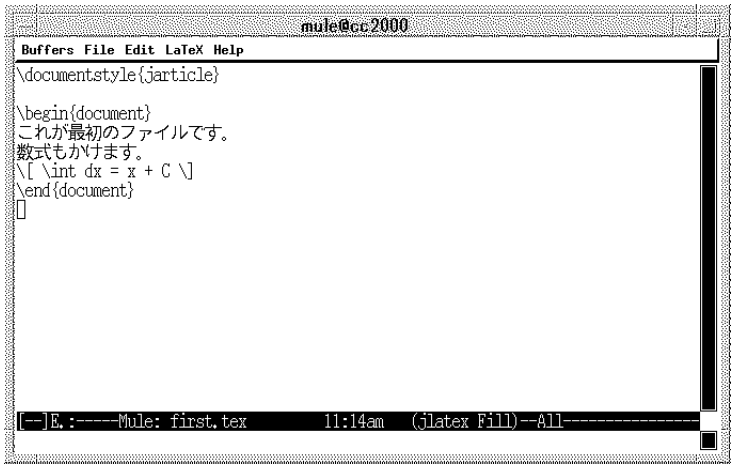


図 2.1 first.tex の画面

ファイルが出来たらMuleを C-x C-c で終了します。first.tex というファイルがあることを確認してcc2000のウィンドウの画面で、jlatex first.tex<return>と入力してやります。そうすると次のように表示されます。

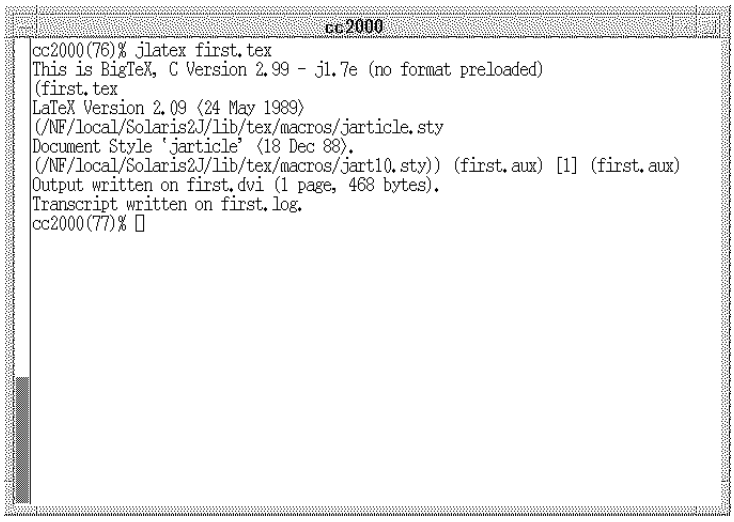


図 2.2 うまくいった時の画面

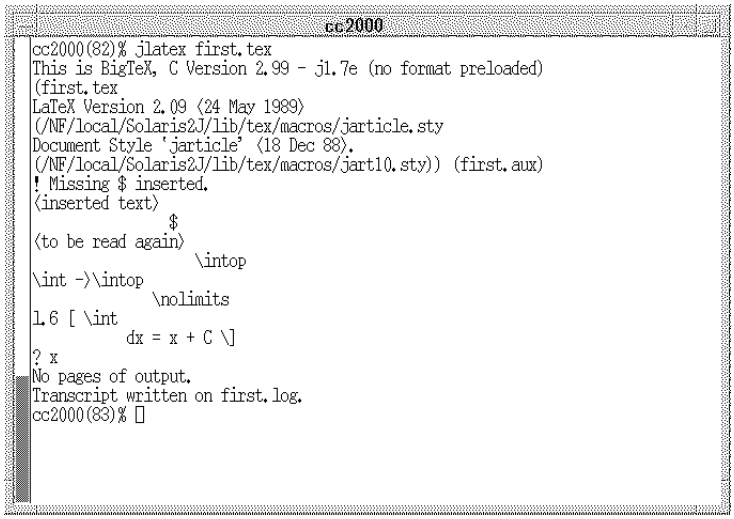


図 2.3 エラーがあるときの画面

もしエラーがあった場合には、「?」マークが表示されますので、その後に、x を入力すると、コンパイルが中断されます。またどうしても処理が中断できない場合は C-c でプロセスを中断してください。その後、タイプミスの部分を探して直してからもう一度コンパイルの作業をして下さい。詳しいことは、第 2.18節 (80ページ) を参照して下さい。

さて、実際にコンパイルされたものがどのようなものに仕上がっているのか確認してみましょう。それには、プロンプトで`xdvifirst.dvi &`と打ち込んでみて下さい。そうすると、次のような画面が現れてきます。

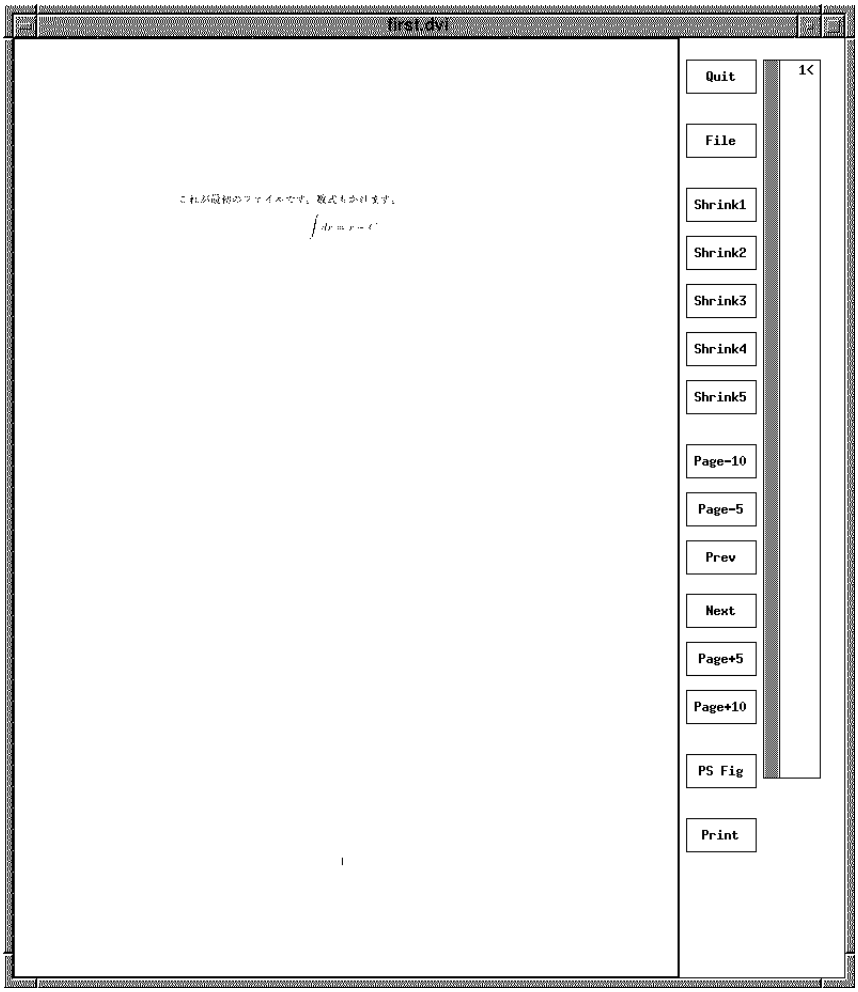


図 2.4 xdvi の画面

これが、コンパイルされて出来たものを表示した画面です。次にプリントアウトするのですが、このままではプリンターから出力できません。そのためプリントアウトできるようにして変換しておかなければなりません。

```
jdvi2kps first.dvi > first.ps<return>
```

と打ち込んでやります。これの意味はプリンターが出力できるポストスクリプトという形式に変換する作業です。

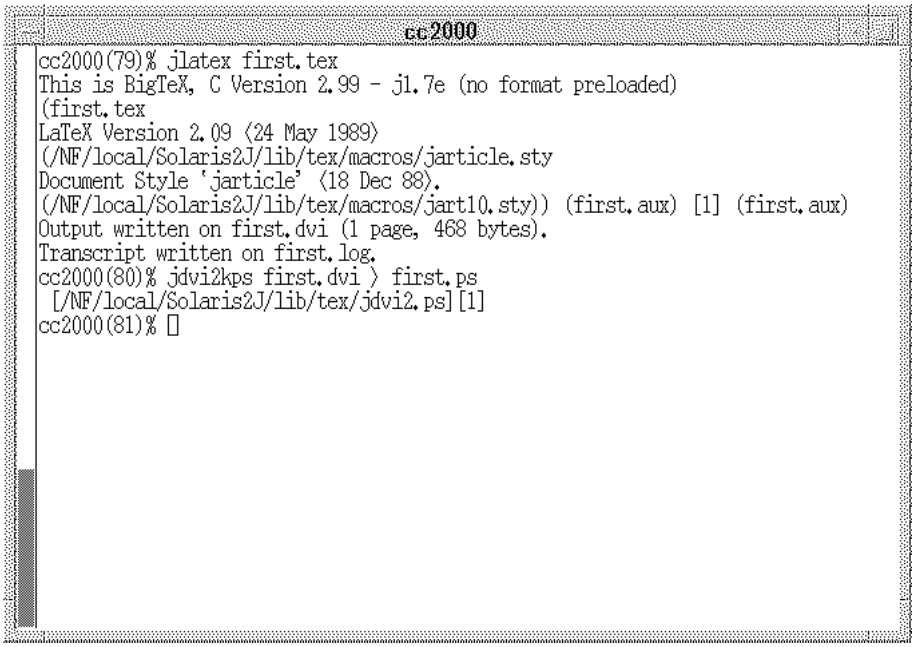


図 2.5 ポストスクリプトへの変換の画面

ようやくこれでプリントアウトできるようになりました。それでは 21 情報処理教室のプリンターから出力してみましょう。

`lpr -Pcspr01 first.ps<return>`

と打ち込んで下さい。これも作業しているディレクトリの中でどちらのウインドウでも構いません。

21 情報処理教室のプリンター以外から出力する場合は **cspr01** を下記のプリンタから当てはまるものに変更して下さい。

これで一通りの LaTeX を使う作業の手順を紹介しました。

また、dvi ファイルで出来具合いを確認して、直接プリントアウトが出来ます。もう一度 `xdvi first.dvi &` と打ってみて下さい。

dvi ファイルが立ち上がりましたね。その画面の右下のところに、print ボタン

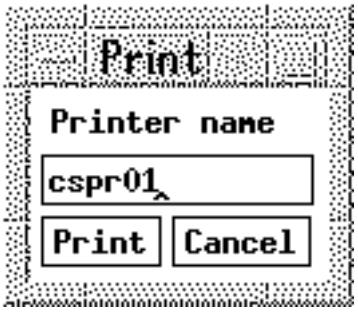


図 2.6 print ボタン

というのがあります。ここで、「print」と書いてあるところをドラッグして、(マウスのボタンを押しな

がら)「**All pages**」と書かれているところでマウスをはなすと、すべてのページがプリントアウトされます。また、ある部分だけ印刷をしたいときは、第2.19節を参照して下さい。ここで注意しておかなければならないのは、「プリント」のボタンを押した時、プリンターの名前が表示されますが、自分がいる情報処理教室のプリンターの名前になっているか確認して下さい。くれぐれもご注意を！

プリンタ名	設置場所
ccpr01	計算機科学研究所 2 階ミニコン室（白くて小さな方）
ccpr02	計算機科学研究所 2 階ミニコン室（茶色の大きな方）
cspr01	2 号館 4 階 21 情報処理教室
cspr02	2 号館 4 階 21 情報処理教室
clpr01	3 号館 2 階 31 情報処理教室
clpr02	3 号館 2 階 31 情報処理教室
cepr01	5 号館 1 階 51 情報処理教室
c1kpr01	第 1 研究室棟 2 階共同利用室
c2kpr01	第 2 研究室棟 1 階共同利用室
c3kpr01	第 3 研究室棟 1 階共同利用室
cgpr01	11 情報処理教室
cgpr02	11 情報処理教室
cgpr03	11 情報処理教室
cgpr04	11 情報処理教室
c9pr01	9 号館

2.3 LaTeX におけるルール

例示の方法

これ以降では LaTeX の記述を例としてあげる場合に、以下のような記号を用いて表現します。

入力

この記号に続く記述は LaTeX のソースを表しています。

出力

この記号に続く内容が、上記の記述によって出力されます。

2.3.1 最低限のルール

文書ファイルの形式には次のような最低限のルールがあります。

- 文書ファイル名の最後に「.tex」をつけます (例：first.tex)。
- 文書ファイルの最初と最後に半角で次の様に書きましょう。

```
\documentstyle{jarticle}
\begin{document}
        :
        :
        :
\end{document}
```

`\begin{document}`から`\end{document}`までが文書の中身です。

- 使っている環境によって\(バックスラッシュ) が使えないときは半角の円記号￥を使います。この 2 つは同じものだと思ってください。(文字コードが同じです。)

2.3.2 ドキュメントスタイルについて

文書ファイルのはじめに書いた

```
\documentstyle{jarticle}
```

というような行は文書のスタイルを指定したものです。
そしてこの中括弧 { } の中は実際のスタイルの指定で次のようなものがあります。

jarticle .. 論文・雑誌の記事・短いレポートに使う最も一般的なスタイルです。

jreport ... 長いレポートなどに使うスタイルです。

jbook ..... 本などに使うスタイルです。見開きで見れるように奇数ページと偶数ページのレイアウトが違います。

これ以外にも、幾つかのスタイルファイルがありますので、第 2.22.2節を参照して下さい。

オプションについて

括弧 [ ] の中を「オプション」と呼びます。「オプション」は必要がなければ書く必要はありません。

```
\documentstyle[..]{jarticle}
```

以下、ここにはいるオプションをあげておきます。

文字の大きさ .. 文書全体の文字の大きさを指定します[2]。`10pt` と `11pt` と `12pt` のいずれかの指定ができます。`10pt` の時はここを省略します。

紙の大きさ .... 出力する紙の大きさを指定します。`a4j`(A4)、`a5j`(A5)、`b4j`(B4)、`b5j`(B5) といった指定ができます。普通 `a4j` を指定します。

紙の向き ..... 紙の向きを指定します。縦にして使うときは `portrait`、横にして使うときは `landscape` と指定します。しかし縦書きはできません。デフォルトでは `portrait` になっています。

その他 ..... 二段組にするときは `jtwocolum` と指定します。

上のようなオプションを複数指定してやりたいときは

```
\documentstyle[a4j,12pt]{jarticle}
```

のように半角のコンマ (,) で区切り並べてやります。

2.3.3 見出しの種類

見出しの種類には次のようなものがあります。

種類	コマンド	jarticle	jreport,jbook
部	`\part`	Part I	Part I
章	`\chapter`	使えません	Chapter 1
節	`\section`	1	1.1
小節	`\subsection`	1.1	1.1.1
小小節	`\subsubsection`	1.1.1	1.1.1.1
段落	`\paragraph`	[1.1.1.1]	[1.1.1.1.1]
小段落	`\subparagraph`	[1.1.1.1.1]	[1.1.1.1.1.1]

スタイルファイルによって、出力される形が違いますので、それらを見ていきます。まず、`jarticle` の場合は、

入力

```
\documentstyle[a4j]{jarticle}
\begin{document}
\part{物理学}
\section{現代物理学}
\subsection{量子力学}
\subsubsection{量子力学の発展}
\paragraph{量子力学とは}
\subparagraph{量子力学の歴史}
\end{document}
```

[2]注意：指定した大きさのフォントがないときはエラーがでます。

出力

Part I

物理学

1 現代物理学

1.1 量子力学

1.1.1 量子力学の発展

量子力学とは

と出力されます。jreport では、

出力

Part I

物理学

Chapter 1

物理学

1.1 現代物理学

1.1.1 量子力学

量子力学の発展

量子力学とは

と出力されます。ただし、**Part I** と**物理学**は、一枚の紙の中央に、そして、改ページをして **Chapter** 以下のセクションが出力されます[3]。

最後に、jbook では、一ページ目に、**Part I** と**物理学**が出力され、2 ページ目は、何も出力されず、3 ページ目に、**Chapter** 以下のセクションが出力されます。

[3] ここでは、そのようになっていませんが、、、

2.4 いろいろなコマンドと環境

コマンドと環境の説明

このガイドで使われる**コマンド**はすべて、「 \ 」で始まるもので、タイプライタ体で書かれています。ここでいうタイプライタ体とは、`typewriter style` のような書体です。一方、**環境**とは、
`\begin{...}`と`\end{...}`
の対で使うコマンドの一種みたいなものです。**環境**はすべて**太字体**で書かれています。よく現れてきますので、これから説明するコマンドと環境の使い方を是非覚えて下さい。

2.4.1 特殊文字

さて、`\begin{document}`の下に本文を書いていくわけですが、LaTeX にはいくつかの「ルール」があります。このガイドでは、この「ルール」の説明と解説を一通りしていきます。
本文で、そのまま使える文字や記号は、アルファベットの小文字と大文字、数字の０～９と、！ ” ’ ( ) = - ‘ @ + ; * : , . / [ ] ? の１９個の記号だけです。

これ以外の記号で、`# $ % & ~ ^ _ \ { }` の 10 個の記号は、本文中で単独に使いますと、「ヤバイ」ことになります[4]。

というのも、これらの記号は、それぞれ意味を持っているからです。つまり、

- `#` は、０～９までの数字と組み合わせて、parameter として使います。詳しくは、2.20で。
- `$` は、数字やアルファベットを`$`ではさむと、そのはさんだ部分が数式モードになります。詳しいことは、第 2.12節のところで。
- `%` は、`%`をつけた後の部分からその行の終りまでを無視して出力します。つまり、自分がソースを書いている時の目印とか、その行のコメントとかに使います。

  入力
  ```
  What is done cannot be undone.%後悔先に立たず。
  ```
  出力
  What is done cannot be undone.

  このように、`%`以降の文章が出力されません。
- `&` は、表を作る時、列を区切るのに使います。詳しいことは、第 2.10節で。
- `^` は、上つき文字にします。ただし、数式の環境で使います。詳しいことは、第 2.12節で。

  入力
  ```
  $ a^{11}+b^{11}=0 $
  ```
  出力
  $a^{11} + b^{11} = 0$
- `_` は、下つき文字にします。ただし、これも数式モードで使います。詳しいことは、第 2.12節で。

  入力
  ```
  $ a_{11}+b_{11}=0 $
  ```

[4] 思った通りに出力してくれないばかりか、エラーになるかも知れない、ということです。

出力

$a_{11} + b_{11} = 0$

- ~ は、一個分の空白を作ります。二個つければ 2 個分の空白を作ります。

入力

```
What is do~ne cannot be undone.
```

出力

What is do ne cannot be undone.

- \ は、バックスラッシュといって、コマンドの前につけます。前節で説明しました。
- { と } は、コマンドの後ろにつけます。前節で説明しました。

したがいまして、これらを単独で使うことは避けた方が良いでしょう。どうしても使いたいのであれば、以下のようにして下さい。但し、大きさが少し大きめの文字として出力されます。

特殊文字	出力する為の記述	その結果表示される文字
#	\#	#
$	\$	$
%	\%	%
&	\&	&
~	\~{}	~
^	\^{}	^
_	_	_
\	$\backslash$	\
{	\{	{
}	\}	}

~、^、\ については以下のようにして出力することも出来ます。

特殊文字	出力する為の記述	その結果表示される文字
~	\symbol{"7E}	~
^	\symbol{"5E}	^
\	{\tt\symbol{"5C}}	\

これらの記号について上記の方法が気に入らなければ、第 2.4.2節で紹介する方法で出力して下さい。

また、使ってもエラーを起こしたりする事はないのですが < > | の三つの記号はそれぞれどういうわけか ¡ ¿ — などという記号として出力されてしまいます[5]。< > | の記号を出力したい場合はやはり第 2.4.2節で紹介する方法で出力して下さい。これらの記号については、第 2.4.2節か、第 2.13節で紹介する方法で出力して下さい。

[5] 但し TypeWriter フォントの時は大丈夫です。これが TEX の流儀だそうです。なぜこうしたのか想像もつきませんが。

2.4.2　特殊文字でもそのまま出力する

アルファベットの小文字、大文字や数字、！“ ’ @ ‘ ( ) - = [ ] ; + : * , . ? / の半角文字はそのまま出力されました。これら以外の半角文字は各種のコマンドや意味を持った特殊文字として TEX に解釈されてしまい、そのままでは出力されませんでした。これらの記号を出力させるには **verbatim**[6] という環境を使います。**verbatim** 環境で書かれた文字はコマンドや特殊な意味を持つとは解釈されません。**verbatim** 環境には二種類の使い方があります。以下にそれぞれ説明します。

記号一文字、もしくは一行におさまる文字列をそのまま出力する

`\verb` に続く記号でそのまま出力したい文字列をはさんでやります。どんな記号でも構いませんが、同じ記号ではさまないとエラーになりますので注意しましょう。以下に「 `%`（パーセント記号）」を出力させる例を示します。

入力 `\verb!%!`
出力 `%`

入力 `\verb+sample サンプル ;:@]/., こんなぐあい+`
出力 `sample` サンプル `;:@]/.,` こんなぐあい

複数行にわたる文章をそのまま出力する

複数行にわたる文章をそのまま出力したければ、**verbatim** 環境を使います。

入力
```
\begin{verbatim}

たとえば !@#$%^&*()<>;: なんかも
[]{}'"\|-もみーんなこんなもんさ

\end{verbatim}
```
出力

たとえば !@#$%^&*()<>;: なんかも
[]{}'"\|-もみーんなこんなもんさ

このようになります。

2.4.3　文字の空白

文字による空白には、全角空白と半角空白があります。全角空白は、そのまま、スペースキーで空けます。半角空白は、何個空けても空白は 1 個分しか空けませんので注意しましょう。文字の空白は、`\␣`を使用します。␣ は、space key を一回押す、という意味です。そうすると、半角文字一個分だけ空白ができます。それ以上 space key をいれて空白を空けても 1 個分だけしか空白は空きません。半角空白を何個も出力するのは`\␣`を繰り返しておきます。

入力 `Time\␣shock`
出力 Time shock

[6]「 ヴァーベイティム 」と読みます。日本語で「文字通り」という意味です。

入力	`Time␣␣␣␣␣␣shock`(space key を 6 回押しました。)
出力	Time shock
入力	`Time\␣\␣\␣\␣\␣shock`
出力	Time　　shock

2.4.4　改行と改ページ

LaTeX で改行をするには、一行改行を入れるか、「`\\`」というコマンドをつける必要があります。

一行改行を作ると、LaTeX のルールで改行されて少し頭が引っ込みます。つまり、段落ができるということです。一方、「`\\`」は、強制改行のコマンドでただ改行するだけで頭は引っ込みません。

入力

```
改行を一行作ると改行されます。

そして、頭が引っ込みます。「\verb!\\!」はただ改行するだけです。\\
頭は引っ込みません。\\
このように改行されました。\\
```

出力

改行を一行作ると改行されます。
そして、頭が引っ込みます。「\\」はただ改行するだけです。
頭は引っ込みません。
このように改行されました。

改ページをするには、`\newpage` という強制的に改ページするコマンドがあります。

2.4.5　水平方向と垂直方向の空白

まず、水平方向の空白は、`\hspace` があります。これは、引数[7]と一緒に使います。

入力

```
水平方向に 0.5cm 空白\hspace{0.5cm}を入れます。
```

出力

水平方向に 0.5cm 空白　　を入れます。

このように、「空白」と「を入れます」の間に 0.5cm の空白ができました。

`\hspace` は、右の方をプラスとすると、マイナスの方向つまり、左の方向にも空白を入れることができます。空白を入れるというよりも、空白を「取る」という感じです。

入力

```
水平方向に-0.5cm 空白\hspace{-0.5cm}を入れます。
```

[7] 中括弧の中の数字です。

出力

水平方向に-0.5cm 空白を入れます。

このように、出力されます。よく見ますと、文字が重なってしまいました。これは、「を入れます。」という言葉が前に 0.5cm 移動したと考えて下さい。文章中に\hspace を入れたらそこから水平方向にずれましたが、改行した後で、\hspace を使っても水平方向にはずれません。そこで、\hspace*{...}というコマンドを使います。たとえば、

入力

```
あいうえお\\
\hspace{2cm}あいうえお\\
\hspace*{2cm}あいうえお\\
あいうえおあいうえおあいうえお\\
```

出力

あいうえお
あいうえお
　　あいうえお
あいうえおあいうえおあいうえお

このようにすれば、行の始めから空白を入れることができます。
次に垂直方向の空白です。コマンドは、\vspace です。使い方は、\hspace と同じ使い方です。

入力

```
垂直方向に 3.5cm 空白\vspace{3.5cm}を入れます。\\
こうなります。
```

出力

垂直方向に 3.5cm 空白を入れます。

こうなります。

このように、3.5cm 垂直方向に空白ができました。
\vspace も同様に、マイナスの方向に空白を入れることができます。

入力

```
垂直方向に-0.5cm 空白\vspace{-0.5cm}を入れます。\\
こんな感じになります。
```

出力

垂直方向に 0.5cm 空白を入れます。

御覧のように文字が重なっていまいました。これも「こんな感じになります」という文が、上に 0.5cm 移動したことになります。

長さの単位を以下に挙げておきます。

単位の名前	長さ	幅
cm	センチメートル	＋　＋
em	大文字の M の横幅	＋　＋
ex	小文字の m の横幅	＋＋
in	インチ (約 25.4mm)	＋　＋
pc	パイカ (12pt, 約 4.21mm)	＋　＋
pt	ポイント (0.35mm)	＋＋
mm	ミリメートル	＋＋

出力に出ているのは、その単位で出力される間隔です。[8]

2.4.6　引用

引用を行うものとして、**quote** と **quotation** の二つの環境があります。**quote** 環境は、次のように、短い文章を引用するのに向いています。

入力

```
\begin{quote}
我々は、京都産業大学の学生である。
\end{quote}
```

出力

> 我々は、京都産業大学の学生である。

一方、**quotation**[9] 環境は、比較的長い文章や、複数のパラグラフがあるような文章を引用するのに向いてます。

入力

```
\begin{quotation}
７月初旬のおそろしく暑い時分のこと、とある夕方近く、一人の若い男が、C-横町の借家人からまた借りていた自分の部屋から街路に出て、なんとなく心のきまらないさまで、のろのろと K-橋の方へ歩いて行った。
\end{quotation}
```

出力

[8] この原稿は縮小されて印刷されていますので実際はもう少し広いです。
[9] 出力例を見ていただいたらわかるように左右に余白ができています。

7 月初旬のおそろしく暑い時分のこと、とある夕方近く、一人の若い男が、C-横町の借家人からまた借りていた自分の部屋から街路に出て、なんとなく心のきまらないさまで、のろのろと K-橋の方へ歩いて行った。

2.4.7 箇条書き

箇条書きをするのには、以下の 3 つの方法があります。

- **itemize** は、各々の箇条の先頭に・をつけた (記号つき箇条書き) 形です。
- **enumerate** は、各々の箇条の先頭に番号をつけた形 (番号つき箇条書き) です。
- **description** は、各々の箇条の先頭にラベルをつけた形 (見出しつき箇条書き) です。

箇条書きをするには、これらを環境として使い、それぞれの行の先頭に`\item` をつけます。

記号つき箇条書き

記号つき箇条書きとは、箇条の先頭に記号をつけた形で出力します。

入力

```
\begin{itemize}
\item {\bf itemize}は、各々の箇条の先頭に・をつけた (記号つき箇条書き) 形です。
\item {\bf enumerate}は、各々の箇条の先頭に番号をつけた形 (番号つき箇条書き) です。
\item {\bf description}は、各々の箇条の先頭にラベルをつけた形 (見出しつき箇条書き) です。
\end{itemize}
```

出力

- **itemize** は、各々の箇条の先頭に・をつけた (記号つき箇条書き) 形です。
- **enumerate** は、各々の箇条の先頭に番号をつけた形 (番号つき箇条書き) です。
- **description** は、各々の箇条の先頭にラベルをつけた形 (見出しつき箇条書き) です。

のようになります。

番号つき箇条書き

これは、箇条の先頭に番号をつけた形になります。

入力

```
\begin{enumerate}
\item {\bf itemize}は、各々の箇条の先頭に・をつけた (記号つき箇条書き) 形です。
\item {\bf enumerate}は、各々の箇条の先頭に番号をつけた形 (番号つき箇条書き) です。
\item {\bf description}は、各々の箇条の先頭にラベルをつけた形 (見出しつき箇条書き) です。
\end{enumerate}
```

出力

1. **itemize** は、各々の箇条の先頭に・をつけた (記号つき箇条書き) 形です。
2. **enumerate** は、各々の箇条の先頭に番号をつけた形 (番号つき箇条書き) です。
3. **description** は、各々の箇条の先頭にラベルをつけた形 (見出しつき箇条書き) です。

となります。
また、`\item` の後に、`[1a.]` などの番号を入れておきますと、

入力

```
\begin{enumerate}
\item[1a.] {\bf itemize}は、各々の箇条の先頭に・をつけた (記号つき箇条書き) 形です。
\item[1b.] {\bf enumerate}は、各々の箇条の先頭に番号をつけた形 (番号つき箇条書き) です。
\item[1c.] {\bf description}は、各々の箇条の先頭にラベルをつけた形 (見出しつき箇条書き) です。

\end{enumerate}
```

出力

1a. **itemize** は、各々の箇条の先頭に・をつけた (記号つき箇条書き) 形です。
1b. **enumerate** は、各々の箇条の先頭に番号をつけた形 (番号つき箇条書き) です。
1c. **description** は、各々の箇条の先頭にラベルをつけた形 (見出しつき箇条書き) です。

このようになります。

見出しつき箇条書き

この環境は、前の二つと少し違っていて、以下のようにします。
まず、`\item` の後に、`[見出し]` をつけます。この見出しが前の二つでいう「記号」や「数字」に当たります。例で見てみましょう。

入力

```
\begin{description}
\item[自然数] 自然数とは、
\item[整数] 整数とは、
\item[有理数] 有理数とは、
\item[無理数] 無理数とは、
\item[複素数] 複素数とは、
\end{description}
```

出力

自然数 自然数とは、

整数 整数とは、

有理数 有理数とは、

無理数 無理数とは、

複素数 複素数とは、

これらを入れ子状にしても使えます。例えば、**itemize** 環境では、

入力

```
\begin{itemize}
\item 箇条書き
  \begin{itemize}
  \item 箇条書き
    \begin{itemize}
    \item 箇条書き
      \begin{itemize}
      \item 箇条書き
      \end{itemize}
    \end{itemize}
  \end{itemize}
\end{itemize}
```

出力

- 箇条書き
  - 箇条書き
    - 箇条書き
      - 箇条書き

となります。**enumerate** 環境では、

入力

```
1. 箇条書き
  (a) 箇条書き
    i. 箇条書き
      A. 箇条書き
```

となります。このような形で入れ子状の箇条書きができます[10]。

しかし、この入れ子状も 3 回までで限度になります。それ以上やるとエラーになります。注意しましょう。

[10] 箇条書きの記号や番号は変えることができます。例えば、数字の番号ではなく、アルファベットの番号にすることもできますが、ここでは説明しません。

2.4.8　右寄せ、中央寄せ、左寄せ

文章を右に寄せたり、中央に寄せたり、左に寄せたりするには、**flushright**、**center**、**flushleft** を環境として使います。

入力

```
\begin{flushright}
文章を右によせます。
\end{flushright}
\begin{center}
文章を中央に寄せます。
\end{center}
\begin{flushleft}
文章を左に寄せます。
\end{flushleft}
```

出力

文章を右によせます。

文章を中央に寄せます。

文章を左に寄せます。

という具合になります。

右寄せ、中央寄せ、左寄せは、他に、`\raggedleft` ,`\raggedright` , `\centering` でも行えます。但しこれを書いた以降の全ての文に影響を与えますので、どちらかというと文書全体の形を決める宣言として考えて下さい。どのような形に整形されるかは、既に出来上がっている複数行を含む文書で試して下さい。

2.4.9　文字の大きさ

基本となる文字のサイズは 10pt です。その他に 11pt,12pt が指定できましたね。これ以外で、文字の大きさを文中で指定することができます。以下にあげたように 10 個の文字の大きさがあります。`\tiny` は、ルビサイズです。`\footnotesize` は、脚注の文字の大きさです。`\normalsize` は、普通の文字の大きさです。`\large` からは、文字の大きさが少しずつ大きくなっていきます。(注意)[11]ここでひとこと。文中ではあまり文字のサイズを変えることはおすすめできません[12]。

入力	出力	入力	出力
`\tiny`	tiny	`\large`	large
`\scriptsize`	scriptsize	`\Large`	Large
`\footnotesize`	footnotesize	`\LARGE`	LARGE
`\small`	small	`\huge`	huge
`\normalsize`	normalsize	`\Huge`	Huge

[11] ここで、印刷されているのは縮小して印刷していますので、若干、文字が小さくなっています。
[12] けっこう、読みにくいものです。注意しましょう。

2.4.10 書体

入力	欧文出力	日本語出力
\rm	roman	普通の欧文書体
\bf	**boldface**	**太字体**
\it	*italic*	斜体
\sl	*slanted*	傾斜体
\sf	sans serif	サンセリフ体
\sc	SMALL CAPS	大文字体
\tt	Typewriter	タイプライタ体
\em	*emphasize*	強調体

見て判るように、日本語は太字体にしか変わりません。また、強調体は、斜体と同じ文字となります。

これらの命令はひとたび使いますと、以降の文章をすべて、その書体にしてしまいますので、それを防ぐには、その書体にしたい範囲を{ ... }で囲っておきますと、その括弧の中だけが、コマンドの文体になります。

入力

```
abcdefg 京都産業大学\\
{\rm abcdefg 京都産業大学}\\
{\tt abcdefg 京都産業大学}\\
{\bf abcdefg 京都産業大学}\\
{abc{\bf de}f 京都{\bf 産業}大学}
```

出力

abcdefg 京都産業大学
abcdefg 京都産業大学
abcdefg 京都産業大学
abcdefg 京都産業大学
abc**de**f 京都**産業**大学

文字の大きさと種類を同時に指定する時は、書体を先に指定し、その次に大きさを指定します。

	入力	出力
書体が先で大きさが後	{\huge {\bf UNIX}}	**UNIX**
大きさが先で書体が後	{\bf {\huge UNIX}}	UNIX

下の例では、太文字体の指定が無効になってしまいました。

さらに、{\huge \bf UNIX}のように中の括弧を取り除くことが出来ます。

入力	出力
{\huge \bf UNIX}	**UNIX**

ここで、注意しておかなくてはならないことは、文字の大きさと書体を同時に指定した時、大きさが変わるのは漢字と roman 体、bold 体のみである、ということです。つまり、文字を大きくする命令は漢字と roman 体、blod 体にしか働かないということです。従って、これ以外の書体で大きくしたいときには、2.24 節を参照して下さい。

2.5 表題

2.5.1 タイトルの作り方

タイトルをつけるには、以下にあげるコマンドを用います。

- \title{タイトル名}—ここにタイトル名を書きます。
- \author{著者名}—著者が複数の時は、\and コマンドを使って、連名にします。
- \date{日付}—\date が省略された時は、その原稿ファイルが LaTeX にかけられ、結果ファイルが作られた日になります。
- \maketitle—タイトルを出力します。

これらは、\documentstyle{...}と\begin{document}の後ならどこでも構いません。
そして、\maketitle は、それらの後におきます。後の例を参考にして下さい。

2.5.2 概要の作り方

タイトルの次は、概要の作り方です。これは論文などではよく見かけられるものです。やり方は、

```
\begin{abstract}
    概要の内容
\end{abstract}
```

とします。

入力

```
\title{ここにタイトルがつきます。}
\author{ここが書いた人の名前\and 共同者の名前}
\date{出来上がった日付}
\maketitle
\begin{abstract}
  今までワープロを使っていると、数学の式で、複雑なものはできませんでした。ところが、\LaTeX は、それをきれいな形で出力できます。また、論文や、レポートなどにも手軽に使えます。ただ、ルールがあって、ワープロのようにすぐに使えるわけではありません....。
\end{abstract}
```

出力

ここにタイトルがつきます。

ここが書いた人の名前　　共同者の名前

出来上がった日付

Abstract

今までワープロを使っていると、数学の式で、複雑なものはできませんでした。ところが、LaTeX は、それをきれいな形で出力できます。また、論文や、レポートなどにも手軽に使えます。ただ、ルールがあって、ワープロのようにすぐに使えるわけではありません。....

ここで、注意してほしいのは、ドキュメントスタイルを指定すると、それに合わせて、Part や Chapter が変るのと同じように、表題もスタイルファイルによって変わります。つまり、上の例は、jarticle で書かれたものです。jreport とjbook では、一枚の紙に、タイトル、著者、日付を出力します。そして、ページをかえて、Abstract を出力し、またページをかえて、本文が始まります。ところが、\jbook では、Abstract が使えません。

ドキュメントスタイル	title	author	date	Abstract
jarticle	○	○	○	○
jreport	○	○	○	○
jbook	○	○	○	×

2.6 傍注

傍注として本文の横に「余白だ!」と出力するには、\marginpar {余白だ!}というコマンドを使います。 余白だ！

傍注が本文の右に来るか、左に来るかは\documentstyle{...}で指定した本文のスタイルに依存します。どこに出るかはいろいろ試して見るのがいいでしょう。

\marginpar[左余白]{右余白}と書けば、右に余白があった場合にはそこに「右余白」、左に余白があった場合にはそこに「左余白」と出力されます。

2.7 脚注

脚注[13] を出力するには、\footnote というコマンドを使います。\footnote{ここに出る注釈を脚注と呼びます。}という具合です。

本文中の\footnote が現れたところのフットノートナンバーが振られ、括弧の部分がページの下に書かれます。

2.8 相互参照

論文や、レポートといった文章を書いている時、「第?節を参照して下さい」とか、「(6) 式に (7) を代入して...」といったことがよくあります。これをそのまま本文に書いていると、文章を加筆したり、削除したりすると番号が違ってくる時があります。

そうなると、いちいち番号をかえていかなくてはなりません。そこで、この番号振りを自動でやってくれるのが、\label{...}というコマンド[14] です。例えば、

```
\section{...}\label{bun}
```

としておきますと、{...} というセクションに「bun」というラベルがついたことになります。

このラベルには何でも構いませんが、\,{,},... といった特殊な記号は駄目です。

また、ダブって使われるラベルはエラーのもとになりなすので注意しましょう。ラベルを出力するには、相棒となる\ref{...}を使います。また、ページ番号を出力したい時には、\pageref{bun}とすると、出力されます。

入力例

```
\subsection{相互参照}\label{bun}
... 第\ref{bun}節参照...
```

[13] ここに出る注釈を脚注と呼びます。

[14] ラベルは本文中ならオールマイティにつけることができます。

```
...\pageref{bun}...
```

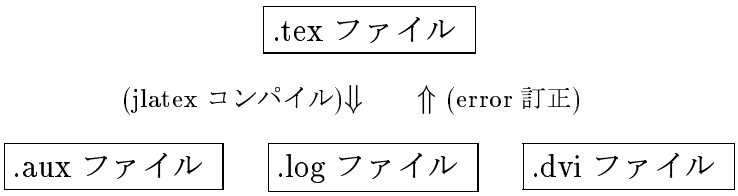
出力例

... 第 2.18.2節参照...

...82ページ参照...

(復習)

`\label` コマンドを使った時は、必ず、コンパイルを 2 回以上行なって下さい。というのは、一連のコンパイルの手順は、以下のような形になります。



.tex ファイルを作って、一回でもコンパイルをすると、この残り 3 つのファイル[15]が出来上がります。

- .tex ファイル 原稿のファイルです。
- .aux ファイル は、補助ファイルです。ここに、相互参照のラベルのついたものが書かれます。
- .log ファイル は、エラーメッセージや実行状態が書かれます。ここに、エラーの原因が書いてありますので、もしコンパイルのときエラーを起こして、直したけれども、それでもまだエラーを起こすような時、原因がよくわからなければここを見ると解決することがあります。
- .dvi ファイル 出力結果のファイルです。

相互参照に利用されるファイルは上記の .aux ファイルです。これは 1 回だけでは、ただラベルが列挙されただけで、`\label` と`\ref` の対応関係がついていません。もう一度コンパイルの作業をすることによって、きちんと`\label` と`\ref` の対応関係がつきます。

つまり相互参照を含むドキュメントを修正した場合に、参照している章節の番号やページ位置が保証されるのは内容を修正せずに二回連続でコンパイルした時だけだということです。

[15] これ以外にも LaTeX コンパイルした時に作られるファイルはあります。

2.9 箱

LaTeX では、いろいろな箱を作ることができます。枠つきの箱、点線の箱、枠なしの箱があります。

2.9.1 一行に収まる文字列を囲む

枠つきの箱 fbox と framebox

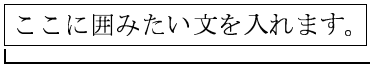
枠つきの箱を作るコマンドには、

- \fbox
- \framebox

があります。

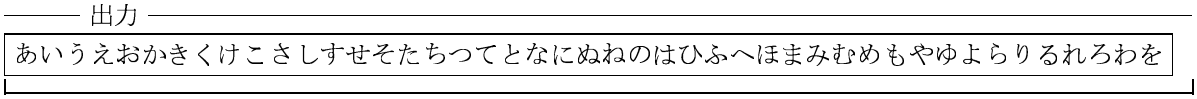
\fbox は、一行分の箱を作ります。使い方は、

入力
```
\fbox{ここに囲みたい文を入れます。}
```
出力



となります。この\fbox は、一行分の箱しかできません。つまり、２行以上にわたる文もすべて一行にしてしまいします。例えば、

入力
```
\fbox{あいうえおかきくけこさしすせそたちつてと\\ なにぬねのはひふへほまみむめもやゆよらりるれろわを}
```
出力

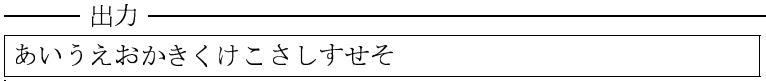


のように、一行の箱になります[16]。\fbox は、キチキチの箱でしたが、\framebox というコマンドを使えばこの枠を自由に変えることができます。
\framebox[長さ][位置]{囲みたい文}とします。文の位置は、指定しないと中央になります。位置の指定には他に、

- [r] 右に合わせます。
- [l] 左に合わせます。

があります。いろいろな長さの箱ができますが、これも一行分の箱しかできません。例えば、

入力
```
\framebox[10cm][l]{あいうえおかきくけこさしすせそ}
```
出力



となります。

[16] 一行に収まらない文は、はみ出してでも一行の箱にします。

もっとでかい箱を作るには、\framebox を、以下のようにして使います
\framebox(横の大きさ, 縦の大きさ)[位置]{箱の中に入る文}
のようにして使います。縦、横の大きさの単位は pt(ポイント) を使います。位置は、中央が基本で、

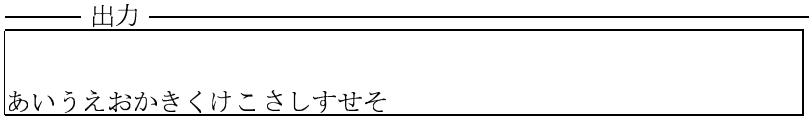
- [t] 上の中央に合わせます。
- [b] 下の中央に合わせます。
- [r] 右の真中に合わせます。
- [l] 左の真中に合わせます。
- [tr] 右上の角に合わせます。
- [tl] 左上の角に合わせます。
- [br] 右下の角に合わせます。
- [bl] 左下の角に合わせます。

があります。例えば、

入力

```
\framebox(300,30)[bl]{あいうえおかきくけこさしすせそ}
```

出力



となります。これも、一行分しか入りません。試してみて下さい。

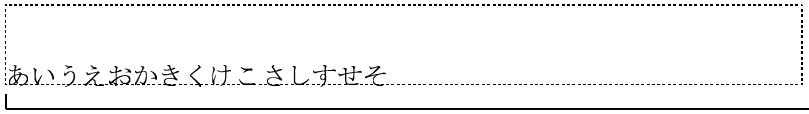
点線の箱 dashbox

点線の箱を作るには、\dashbox というコマンドを使います。使い方は、
\dashbox{点線の長さ}(横の大きさ, 縦の大きさ)[位置] {箱の中に入る文}
のようにします。例えば、

入力

```
\dashbox{1}(300,30)[bl]{あいうえおかきくけこさしすせそ}
```

出力



となります。

枠なしの箱 makebox

今までは枠がありましたが、\makebox というコマンドを使えば枠のない箱を作ることもできます。使い方は\framebox と同じです。
\makebox(横の大きさ, 縦の大きさ)[位置]{箱の中に入る文}
例えば、

入力

```
\makebox(300,30)[tl]{あいうえおかきくけこさしすせそ}
```

出力

あいうえおかきくけこさしすせそ

となります。これも、一行分しか入りません。

2.9.2 複数行にわたる文の箱を作る

今までは一行分の箱しかできませんでしたが、複数行の箱を作るには、\parbox というコマンドと、**minipage** 環境があります。

parbox を使った箱

\parbox を使うには、以下のようにします。
\parbox[基準線]{横幅の長さ}{文章}
とします。とにかく例を見て下さい。●は、わかりやすくするために\parbox と\parbox を区切っています[17]。

入力

```
\parbox[t]{4.5cm}{あいうえおかきくけこさしすせそ}●
\parbox[t]{4.5cm}{あいうえおかきくけこさしすせそたちつてとなにぬねのはひふへほ}●
\parbox[t]{4.5cm}{あいうえおかきくけこさしすせそたちつてとなにぬねのはひふへほまみむめもやゆよらりるれろわゐゑを}
```

出力

あいうえおかきくけこさしす
せそ●あいうえおかきくけこさしす
せそたちつてとなにぬねのは
ひふへほ●あいうえおかきくけこさしす
せそたちつてとなにぬねのは
ひふへほまみむめもやゆよら
りるれろわゐゑを

また、

入力

```
\parbox[b]{4.5cm}{あいうえおかきくけこさしすせそ}●
\parbox[b]{4.5cm}{あいうえおかきくけこさしすせそたちつてとなにぬねのはひふへほ}●
\parbox[b]{4.5cm}{あいうえおかきくけこさしすせそたちつてとなにぬねのはひふへほまみむめもやゆよらりるれろわゐゑを}
```

出力

あいうえおかきくけこさしす
せそ●ひふへほ
あいうえおかきくけこさしす
せそたちつてとなにぬねのは●りるれろわゐゑを
あいうえおかきくけこさしす
せそたちつてとなにぬねのは
ひふへほまみむめもやゆよら

[17] この●は別のものでも、なくても構いません。

となります。例で見ていただいたように、基準線とは、\parbox を上で揃えるか[t]、下で揃えるか[b]、ということです。枠の大きさを決めてしまえば、後はその大きさに沿って文章を改行していきます[18]。これは枠がありませんでしたが、枠をつけるには、\fbox と一緒に使います。つまり、

入力

```
\fbox{\parbox[t]{4.5cm}{あいうえおかきくけこさしすせそ}}●
\fbox{\parbox[t]{4.5cm}{あいうえおかきくけこさしすせそたちつてとなにぬねのはひふへほ}}●
\fbox{\parbox[t]{4.5cm}{あいうえおかきくけこさしすせそたちつてとなにぬねのはひふへほまみむめもやゆよらりるれろわゐゑを}}●
```

出力

あいうえおかきくけこさしすせそ● あいうえおかきくけこさしすせそたちつてとなにぬねのはひふへほ● あいうえおかきくけこさしすせそたちつてとなにぬねのはひふへほまみむめもやゆよらりるれろわゐゑを●

と枠ができ上がります。

2.9.3 minipage 環境

これは、\parbox を発展させたものです。以下のようにして使います。

```
\begin{minipage}[基準線]{横幅の長さ}

        ..................

\end{minipage}
```

入力

```
\begin{minipage}[t]{8cm}
{\bf minipage}環境は、小さなページのようになります。そして、
脚注もできます\footnote{footnote もつけることができます。}。\\
\end{minipage}
```

出力

minipage 環境は、小さなページのようになります。
そして、脚注もできます[a]。

[a] footnote もつけることができます。

ここで[t] は、基準線を決めるものでした。そして、$8cm$ の小さなページができあがりました。さらに、このページを枠で囲ってみましょう。

入力

[18] 改行するというより、文章をつめていく、と言った方がいいかも知れません。

```
\fbox{
      \begin{minipage}[t]{8cm}
      {\bf minipage}環境は、小さなページのようになります。そして、
      脚注もできます\footnote{footnote もつけることができます。}。
      \end{minipage}
     }
```

出力

minipage 環境は、小さなページのようになります。そして、脚注もできます[a]。

[a] footnote もつけることができます。

次の例は **minipage** 環境をたくさん使ったものです。

入力

```
\fbox{
  \begin{minipage}{15cm}
    minipage 環境です。\\
     \fbox{
       \begin{minipage}{10cm}
         minipage 環境です。\\
          \fbox{
            \begin{minipage}{5cm}
              minipage 環境です。\\
               \fbox{
                 \begin{minipage}{4cm}
                   minipage 環境です。\\
                 \end{minipage}}
            \end{minipage}}
      \end{minipage}}
  \end{minipage}}
```

出力

minipage 環境です。

minipage 環境です。

minipage 環境です。

minipage 環境です。

このような使い方も出来ます。

2.10 表

表を作るのが、**tabular**[19]環境です。表を作る環境は、**tabular** 環境の他に、**tabbing** 環境がありますが、ここでは、省略させていただきます。

```
\begin{tabular}{列指定}
表全体
\end{tabular}
```

このような形で作ります。列指定とは、表の中にある要素一個一個の配置を指定します。列指定には以下のものがあります。

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

そして、列の数だけこれを並べます。表の要素一個一個を区切るには、「 & 」を入れます。そして、列の最後には、「 \\ 」をつけます。くれぐれも列指定の数と表の要素の列の数を同じにして下さい。

入力

```
\begin{tabular}{lcrlcr}
品名 & 値段 & 数量 & 品名 & 値段 & 数量 \\
苺 & 200 円 & 2 ケース & 葡萄 & 400 円 & 3 ケース\\
西瓜 & 1000 円 & 2 玉 & 檸檬 & 100 円 & 10 個\\
\end{tabular}
```

出力

品名	値段	数量	品名	値段	数量
苺	200 円	2 ケース	葡萄	400 円	3 ケース
西瓜	1000 円	2 玉	檸檬	100 円	10 個

これに罫線を入れてみましょう。縦の罫線は、列指定のところで、入れたいところに「 | 」を入れます。横の罫線は、**\hline** です。「 \\ 」の後に置きます。「 || 」や**\hline\hline** とすれば、2 重線になります。

入力

```
\begin{tabular}{|l|c|r||l|c|r|} \hline
品名 & 値段 & 数量  & 品名 & 値段 & 数量 \\ \hline
苺 & 200 円 & 2 ケース & 葡萄 & 400 円 & 3 ケース\\ \hline
西瓜 & 1000 円 & 2 玉 & 檸檬 & 100 円 & 10 個  \\ \hline
\end{tabular}
```

出力

品名	値段	数量	品名	値段	数量
苺	200 円	2 ケース	葡萄	400 円	3 ケース
西瓜	1000 円	2 玉	檸檬	100 円	10 個

[19]「タビュラー」と読んで下さい。

となります。
もう少し複雑な表を作ってみましょう。

入力

```
\begin{tabular}{|c|c|c|}\hline
時刻 & \multicolumn{2}{c|}{平日}\\ \cline{2-3}
     & 神社発    & 本学発    \\ \hline
10   & 5,10,20  & 0,10,15  \\
     & 25,30,45 & 20,35,45 \\ \hline
\end{tabular}
```

出力

時刻	平日	
	神社発	本学発
10	5,10,20	0,10,15
	25,30,45	20,35,45

ここで使われている\multicolumn というコマンドは、表のある部分だけ、例えばこの表であれば、「平日」という項が、「神社発」と「本学発」の欄にまたがっています。つまり、複数の項目にわたって出力したり、その欄だけ他と違うような形で出力したい時に使います。使い方は、
\multicolumn{またがる項の数}{列指定}{項目の内容}
とします。ここでは、\multicolumn{2}{c|}{平日}としました。また、\cline{2-3}というコマンドは、2 項目と 3 項目にだけ線を引きます。だから、ここの数字をかえれば、部分的に線を引くことができます。
これ以上の詳しい説明はやめにしておきますが、付録の参考文献にあげておきました LaTeX に関する本を参考して、もっと複雑な表を作ってみて下さい。

2.11 絵

文章を書いていると、絵や図が必要になることがあります。そんな時役に立つが **figure** 環境です。ここでは、絵や図の作り方は説明しませんが、xpaint とか tgif などで作ったとします。ただし、これらの絵や図をセーブする時は必ず PostScript 形式で行なって下さい。

さて、注意しなければならないことがあります。それは、`\documnetstyle` のところで、「`[epsbox]`」というオプションをつけなくてはいけません。これがないとエラーになります。

では、実際にやってみましょう。mathematica でトーラス曲線をはめ込みたいとします。Mathematica で、グラフを書いて、xv を使って、グラフを切りとります。

入力

```
\begin{figure}[hbtp]
  \begin{center}
    \leavevmode
    \psbox[scale=0.5]{tolus-curve.ps}
  \end{center}
  \label{tolus}
  \caption{トーラス曲線}
\end{figure}
```

出力

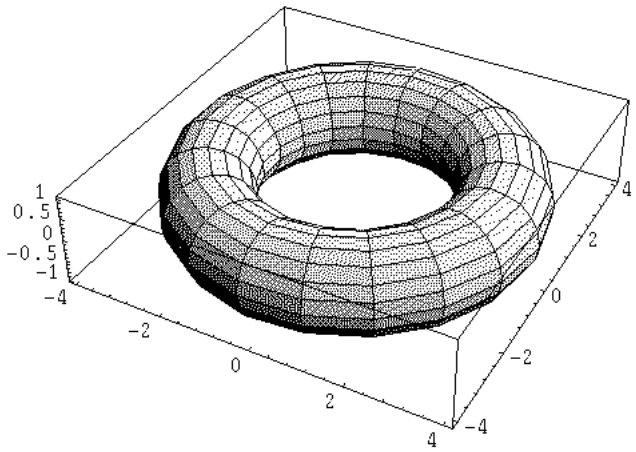


図 2.7 トーラス曲線

それでは、説明していきます。

- まず、**figure** 環境を書きます。これは単に絵や図をはめ込むための領域を確保するためのものです。
- オプションとして、`[hbtp]` を指定しておきます
- **center** 環境は、単に真中に図を置きたかったから入れました[20]。

[20] 別にあってもなくても構いません。

- これが重要なコマンド`\psbox` です。このコマンドが絵や図を文章の中にはめてくれます。
- `[scale=0.5]` は、図の倍率の指定です。ここでは、元の大きさの 0.5 倍に指定しています。
- `{tolus-curves.ps}`は、ファイル名です。「.ps」は省略しても構いません。
- `\label` コマンドは、第 2.18.2節を参照して下さい。
- `\caption` コマンドは、絵や図の下に、表題をつけてくれます。別にいらなければ書かなくても結構です。

これらをとにかく呪文のように打って下さい。これで、図を文章の中にはめ込むことが出来ます。

コメント

オプションとして`[hbtp]` と指定しましたが、これらは次のような意味です。

h here の頭文字で、「ここにはめ込む」という感じです。

b 「そのページの下にはめ込む」という感じです。入らなければ次のページの下にはめ込みます。

t 「そのページの上にはめ込む」という感じです。こちらも入らなければ次のページの上にはめ込みます。

p 独立したページにはめ込みます。つまり、その絵を出力し改ページをします。

それぞれの意味を書きましたが、LaTeX は、オプションの順番通りに作業してみて、一番いい状態のものを出力してくれます。しかし、おそらくあなたの希望通りには絵をはめ込んではくれないでしょう。つまり、「ここに絵を入れたい」といっても LaTeX はやってくれるかどうかわかりません。そこで、`\newpage` コマンドを使って何とかしてみるしかありません。

このように PostScript 形式のファイルであれば、LaTeX にはめ込むことができます。もう一つ、tgif で書いた`coordinates.obj` という図[21]もはめ込むこともできます。

入力

```
\begin{figure}[htbp]
  \begin{center}
    \input{coordinates.tps}\label{fig-1}
  \end{center}
  \caption{球面座標の直交基底}
 \end{figure}
```

出力

[21] tgif のところを参照して下さい。

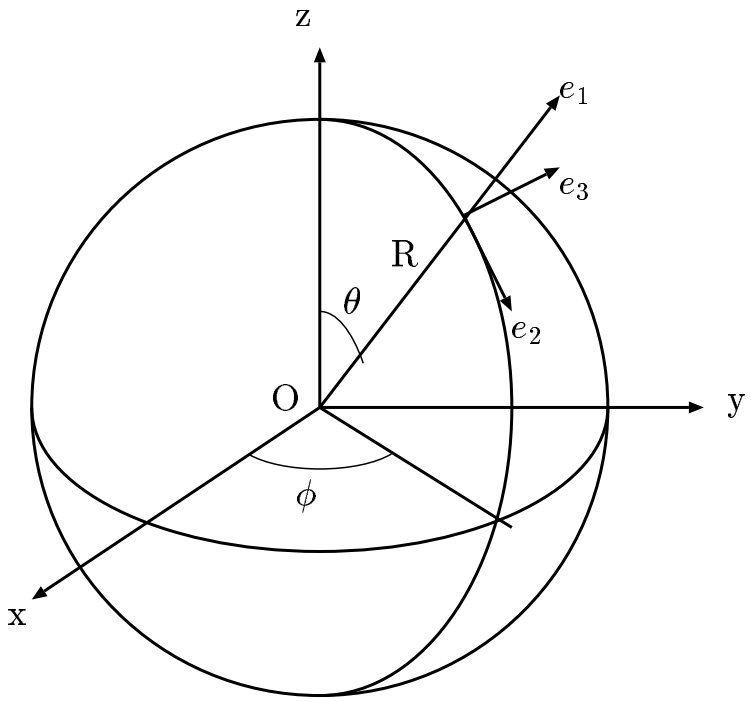


図 2.8 球面座標の直交基底

2.12　数式

2.12.1　数式の環境

displaymath 環境と math 環境

数式を使う方法として、数式を単に表示する方法と、文中に数式を入れる二通りの方法があります。**displaymath** 環境は、本文とは別の行に、（普通次の行ですね）数式を出力します。**math** 環境は、文中に数式を出力します。しかし、「displaymath 」と打ち込むのが大変だという人は、`\[...\]` という環境がありますので、こちらを使ってもよいでしょう。後者も`$...$`か、あるいは、`\(...\)` で挟んだ部分が数式環境として扱われます。

- **displaymath** か、`\[...\]` のどちらか。これがあると、改行して、数式を本文の中央に出力します。
- **math** か、 `$...$` か、`\(...\)` のいずれか。本文中に続けて出力します。

本文中の一部分に数式が現れる時

本文中に現れる数式を出力するには、二通りの方法があります。

入力

```
\begin{math}
ここで、x=r\cos \theta ,y=r\sin \theta とおくと、...
\end{math}
```

出力

ここで、$x = r\cos\theta, y = r\sin\theta$とおくと、...

あるいは、

入力

```
ここで、$x=r\cos \theta ,y=r\sin \theta $とおくと、...
```

出力

ここで、$x = r\cos\theta, y = r\sin\theta$とおくと、...

どちらの方法でも同じように出力されました。

数式だけ出力する時

これも二通りのやり方があります。

入力

```
\begin{displaymath}
a(b+c)=ab+ac
\end{displaymath}
これは分配法則とよばれるもので、...
```

出力

ここで、以下のことが成り立つとします。

$$a(b+c)=ab+ac$$

これは分配法則とよばれるもので、...

あるいは、

入力

```
ここで、以下のことが成り立つとします。
\[ a(b+c)=ab+ac \]
これは分配法則とよばれるもので、...
```

出力

ここで、以下のことが成り立つとします。

$$a(b+c)=ab+ac$$

これは分配法則とよばれるもので、...

このように出力されます。

equation 環境と eqnarray 環境

displaymath 環境と **math** 環境は、一行の数式で、数式番号がつきませんでした。この数式番号をつける環境が、**equation** 環境と **eqnarray** 環境です。

入力

```
\begin{equation}
a+b=c
\end{equation}
```

出力

$$a+b=c \tag{2.1}$$

しかし、**equation** 環境も一行分の数式しか扱えません。

複数の数式を扱うには、**eqnarray** という環境を使います。**eqnarray** 環境を使うと、すべての行に番号が振られてしまいますので、番号を付けないようにするには、**eqnarray*** としてやります。一部だけに付けるには、番号を付けない行の終わりに、`\nonumber` をつけます。

入力

```
\begin{eqnarray}
a+b=c\\
c+d=e\\
e=a+b+d\nonumber
\end{eqnarray}
```

出力

$$a+b=c \tag{2.2}$$
$$c+d=e \tag{2.3}$$
$$e=a+b+d$$

入力

```
\begin{eqnarray*}
a+b=c\\
c+d=e\\
e=a+b+d
\end{eqnarray*}
```

出力

$$a+b=c$$
$$c+d=e$$
$$e=a+b+d$$

これ以降に挙げる数式の様々な例については、全て数式環境の中で書くこととします。

2.12.2 添字

数式の上付き添字は、「 ^ 」を用いて、上にしたい添字の部分を{...}で囲みます。下付き添字も同様に、「 _ 」を用いて下にしたい部分を{...}で囲みます。

入力

```
f(x)=a_{0}x^{n}+a_{1}x^{n-1}+ \cdots +a_{n-1}x+a_{n}
```

出力

$$f(x)=a_0x^n+a_1x^{n-1}+\cdots+a_{n-1}x+a_n$$

入力

```
{X^Y}^Z    {{{6^6}^6}^6}^6
```

出力

$$X^{Y^Z} \qquad 6^{6^{6^6}}$$

2.12.3 平方根

平方根をするには、
\sqrt{...}

というコマンドを用います。また、n 乗根は、
`\sqrt[n]{...}`
としてやります。

入力
```
\sqrt[5]{(x_{1}^2+y_{1})^3}
```
出力

$$\sqrt[5]{(x_1^2+y_1)^3}$$

入力
```
\sqrt[3]{\sqrt{2}+1}
```
出力

$$\sqrt[3]{\sqrt{2}+1}$$

2.12.4 分数

分数を表すには、
`\frac{分子}{分母}`
というコマンドを用います。

入力
```
y=\frac{ax+b}{cx+d}
```
出力

$$y=\frac{ax+b}{cx+d}$$

分数表示は、文中でも使えますが、$y=\frac{ax+b}{cx+d}$のように、小さく出力され、細かい字があると見えにくくなります。そういう場合は、$y=(ax+b)/(cx+d)$ と普通の「/」を使えば、見栄えが良くなります。どうしても $y=\frac{ax+b}{cx+d}$の形で見やすくするには、
`$\displaystyle y=\frac{ax+b}{cx+d}$`
と書きます。そうすると、文中でも、$y=\displaystyle\frac{ax+b}{cx+d}$のようになります。`\displaystyle` は、**displaymath** 環境のように、数式を本文中に出力するコマンドです。

連分数も書けます。

入力
```
\frac{c_1}{b_1 + \frac{c_2}{b_2 + \frac{c_3}{b_3}}}
```
出力

$$\frac{c_1}{b_1+\frac{c_2}{b_2+\frac{c_3}{b_3}}}$$

以下のようにも書けます。

入力

```
a_0 + \frac{b_1}{a_1}{{}\atop +}
      \frac{b_2}{a_2}{{}\atop +}
      \frac{b_3}{a_3}{{}\atop +}
      \frac{b_4}{a_4}{{}\atop +}
```

出力

$$a_0 + \frac{b_1}{a_1} \frac{b_2}{a_2} \frac{b_3}{a_3} \frac{b_4}{a_4} +$$

ここで、\atop というコマンドは、{上\atop 下}のようにして使います。これは、横棒のない分数を出力するコマンドです。$\overset{上}{下}$のようになります。もし、上または、下に何もついていない場合は、上記の例のように{}としてやります。

2.12.5 括弧

{} を出力するには \{ \} などとします。それ以外の ( ) [ ] についてはそのまま書けば出力されます。即ち数式の中で $[\{ (abc) \}]$ と書けば $[\{(abc)\}]$ と出力されます。但し括弧は対にして使わないとエラーになります。

括弧が全て同じ大きさでは見づらい場合があるかも知れません。大きくするには、括弧の前に、\big をつけます。

左括弧には、\bigl、右括弧には、\bigr とします。\big より更に大きくするには、\Big,\bbig,\Biggl をつけます。

入力

```
( \bigl( \Bigl( \biggl( \(Biggl(
```

出力

$$(\bigl(\Bigl(\biggl(\Biggl($$

\big などは括弧の大きさを絶対的に指定するものですが、次のように \left,\right とすると、括弧の大きさがその内容に合わせて相対的に選ばれます。

入力

```
\left \{ \left( \frac{ax+b}{cx+d} \right)+
         \left( \frac{ex+f}{gx+h} \right) \right \}
```

出力

$$\left\{\left(\frac{ax+b}{cx+d}\right)+\left(\frac{ex+f}{gx+h}\right)\right\}$$

基本的には括弧は左右の数が合っていないとエラーになります。もし片方だけの括弧を使いたいなら、以下のようにしてピリオド (.)[22]を使います。

入力

```
\left \{ \left( \frac{ax+b}{cx+d} \right)+
```

[22] ピリオドは何も出力しないという意味です。

```
\left( \frac{ex+f}{gx+h} \right) \right.
```

出力

$$\left\{ \left(\frac{ax+b}{cx+d} \right) + \left(\frac{ex+f}{gx+h} \right) \right.$$

2.13 LaTeX で扱える記号

LaTeX で扱える各種の記号類を挙げます。

2.13.1 雑記号

入力	出力	入力	出力	入力	出力	入力	出力
\S	§	\P	¶	|	—		
\lq	‘	\rq	’	\lbrack	[	\rbrack	]
\slash	/	\copyright	©	\aa	å	\AA	Å
\dag	†	\ddag	‡	\TeX	TeX	\LaTeX	LaTeX
\dots	...	-	-	--	–	---	—

独立した記号ではありませんが、望みの文字に下線を引くことも出来ます。

入力 `\underbar{a}`

出力 a

入力 `ここらで一発\underbar{下線}が欲しいなあ`

出力 ここらで一発下線が欲しいなあ

2.13.2 空白を空ける文字

ここに挙げるコマンドはそれぞれ一定の大きさの空白を、幅ならば右方向に、高さならば下方向に作ります。但し大きさが負の場合、幅ならば左方向に、高さならば上方向に空白が出来ることになります。長さの単位については 第 2.4.5節を参照して下さい。

入力	意味	入力	意味	入力	意味
\space	空白一文字	\empty	何もなし	\null	幅ゼロの箱
\thinspace	1/6em の幅	\negthinspace	-1/6em の幅	\enspace	1/2em の幅
\enskip	1/2em の幅	\quad	1em の幅	\qquad	2em の幅
\,	3mu の幅	$\>$	4mu の幅	$\;$	5mu の幅
\!	-3mu の幅				
\smallskip	3pt の高さ	\medskip	6pt の高さ	\bigskip	12pt の高さ

`$\>$`と`$\;$`については`$` で囲まれていることから判るように数式中で使うべきものですが、入れる場所がなかったのでここに採録しました。本来は数式中で微妙な隙間を作る為に、もともとあった`\,` に追加されたような気がします。なお、これらが使っている mu という長さの単位についてはその定義が判りませんでした。1mu で 1 ミリあるかないかの短い長さです。実際に試してみて下さい。

2.13.3 アクセントなど

入力	出力	入力	出力	入力	出力	入力	出力
\‘{a}	à	\’{a}	á	\v{a}	ǎ	\u{a}	ă
\={a}	ā	\^{a}	â	\.{a}	ȧ	\H{a}	a̋
\~{a}	ã	\"{a}	ä	\d{a}	ạ	\c{c}	ç
\b{aa}	a̲a	\t{aa}	a͡a				

例）

入力 `Poincar\'e`

出力 Poincaré

`\"`を除いては`{}`は不要です。つまり、`\‘␣a␣`で à になります。また、ç は、c 専用の記号です。

2.13.4 ヨーロッパ系言語特有の記号

入力	出力	入力	出力	入力	出力	入力	出力
`<`	¡	`>`	¿	`\l`	ł	`\L`	Ł
`\ss`	ß	`\oe`	œ	`\ae`	æ	`\o`	ø
`\AE`	Æ	`\OE`	Œ	`\O`	Ø		

2.14 数式環境で使える記号

LaTeX の数式で扱える各種の記号類を挙げます。

2.14.1 雑記号

入力	出力	入力	出力	入力	出力	入力	出力
\hbar	$\hbar$	\mho	$\mho$	\surd	$\surd$	\angle	$\angle$
\neq,\ne	$\neq$	\colon	$\colon$	\bowtie	$\bowtie$	\amalg	$\amalg$
\lhook	$\lhook$	\rhook	$\rhook$	\ldotp	$\ldotp$	\cdotp	$\cdotp$
\ldots	$\ldots$	\cdots	$\cdots$	\vdots	$\vdots$	\ddots	$\ddots$
\Vert,\parallel, \\|	$\Vert$	\models	$\models$	\uplus	$\uplus$	\cdot	$\cdot$
\backslash,\setminus	$\backslash$	\vert,\mid	$\mid$	\doteq	$\doteq$	\equiv	$\equiv$
\approx	$\approx$	\sim	$\sim$	\simeq	$\simeq$	\asymp	$\asymp$
\rangle	$\rangle$	\langle	$\langle$	\rbrace	$\rbrace$	\lbrace	$\lbrace$
\rceil	$\rceil$	\lceil	$\lceil$	\rfloor	$\rfloor$	\lfloor	$\lfloor$
\aleph	$\aleph$	\imath	$\imath$	\jmath	$\jmath$	\ell	ℓ
\wp	$\wp$	\Re	$\Re$	\Im	$\Im$	\partial	∂
\infty	∞	\prime	$\prime$	\emptyset	$\emptyset$	\nabla	∇
\top	$\top$	\bot	$\bot$	\perp	$\perp$	\neg	$\neg$
\forall	$\forall$	\exists	$\exists$	\flat	$\flat$	\natural	$\natural$
\sharp	$\sharp$	\dagger	$\dagger$	\ddagger	$\ddagger$	\smallint	$\smallint$
\clubsuit	$\clubsuit$	\diamondsuit	$\diamondsuit$	\heartsuit	$\heartsuit$	\spadesuit	$\spadesuit$
\triangleleft	$\triangleleft$	\diamond	$\diamond$	\Diamond	$\Diamond$	\bullet	$\bullet$
\triangleright	$\triangleright$	\wedge	$\wedge$	\vee	$\vee$	\wr	$\wr$
\bigtriangleup	$\bigtriangleup$	\div	$\div$	\times	$\times$	\mp	$\mp$
\bigtriangledown	$\bigtriangledown$	\pm	$\pm$	\smile	$\smile$	\frown	$\frown$
\cap	$\cap$	\cup	$\cup$	\sqcap	$\sqcap$	\sqcup	$\sqcup$
\sqsubseteq	$\sqsubseteq$	\sqsupseteq	$\sqsupseteq$	\sqsubset	$\sqsubset$	\sqsupset	$\sqsupset$
\odot	$\odot$	\oslash	$\oslash$	\otimes	$\otimes$	\ominus	$\ominus$
\oplus	$\oplus$	\circ	$\circ$	\bigcirc	$\bigcirc$	\triangle	$\triangle$
\propto	$\propto$	\ast	$\ast$	\star	$\star$	\Box	$\Box$
\dashv	$\dashv$	vdash	$\vdash$	\leq	$\leq$	\geq	$\geq$
\succ	$\succ$	\prec	$\prec$	\succeq	$\succeq$	\preceq	$\preceq$
\supset	$\supset$	\subset	$\subset$	\supseteq	$\supseteq$	subseteq	$\subseteq$
\in	$\in$	\ni	$\ni$	\gg	$\gg$	\ll	$\ll$

先に上げた記号の中で、同じ記号を出力しますが、空白領域が微妙に違うものがあります。例えば、\cdotp と\cdot は、

入力 `$GG\cdotp GG\cdot GGG$`

出力 $GG\cdotp GG\cdot GGG$

また、\bot と\perp も微妙に違います。

入力 `$FF\perp FF\bot FF$`

出力 $FF\perp FF\bot FF$

独立した記号ではありませんが、\not を使って否定型を作ることも出来ます。

入力 `a\not\in X`

出力 $a\not\in X$

入力 `A\not\supset B`

出力 $A\not\supset B$

☆少し大きめの記号です。

入力	出力	入力	出力	入力	出力	入力	出力
\coprod	$\coprod$	\intop	$\int$	\prod	$\prod$	\sum	$\sum$
\bigvee	$\bigvee$	\bigwedge	$\bigwedge$	\bigcap	$\bigcap$	\bigcup	$\bigcup$
\bigotimes	$\bigotimes$	\bigoplus	$\bigoplus$	\bigodot	$\bigodot$	\ointop	$\oint$
\bigsqcup	$\bigsqcup$						

☆矢印です。

入力	出力	入力	出力
\Longrightarrow	$\Longrightarrow$	\Longleftarrow	$\Longleftarrow$
\Rightarrow	$\Rightarrow$	\Leftarrow	$\Leftarrow$
\longrightarrow	$\longrightarrow$	\longleftarrow	$\longleftarrow$
\rightarrow , \to	$\rightarrow$	\leftarrow , \gets	$\leftarrow$
\longmapsto	$\longmapsto$	\mapsto	$\mapsto$
\Longleftrightarrow	$\Longleftrightarrow$	\longleftrightarrow	$\longleftrightarrow$
\iff	$\iff$	\leadsto	$\leadsto$
\Leftrightarrow	$\Leftrightarrow$	\leftrightarrow	$\leftrightarrow$
\hookleftarrow	$\hookleftarrow$	\hookrightarrow	$\hookrightarrow$
\rightharpoonup	$\rightharpoonup$	\leftharpoonup	$\leftharpoonup$
\rightharpoondown	$\rightharpoondown$	\leftharpoondown	$\leftharpoondown$
\rightleftharpoons	$\rightleftharpoons$		
\Uparrow	$\Uparrow$	\Downarrow	$\Downarrow$
\uparrow	$\uparrow$	\downarrow	$\downarrow$
\Updownarrow	$\Updownarrow$	\updownarrow	$\updownarrow$
\nearrow	$\nearrow$	\searrow	$\searrow$
\nwarrow	$\nwarrow$	\swarrow	$\swarrow$

続いて単独の記号ではなく、文字の上に付ける記号です。付ける元の文字をコマンドの後ろに{ }で囲んで与えます。

入力	出力	入力	出力	入力	出力	入力	出力
\acute{a}	$\acute{a}$	\grave{a}	$\grave{a}$	\ddot{a}	$\ddot{a}$	\tilde{a}	$\tilde{a}$
\bar{a}	$\bar{a}$	\breve{a}	$\breve{a}$	\check{a}	$\check{a}$	\hat{a}	$\hat{a}$
\vec{a}	$\vec{a}$	\dot{a}	$\dot{a}$				

再び単独の記号ではなく、文字の上に付ける記号です。今度は付ける元の文字の長さに合わせて記号の長さも変化します。同じく付ける元の文字をコマンドの後ろに{ }で囲んで与えます。

入力	出力	入力	出力
\underline{abc}	$\underline{abc}$	\overline{abc}	$\overline{abc}$
\widetilde{abc}	$\widetilde{abc}$	\wodehat{abc}	$\widehat{abc}$
\overrightarrow{abc}	$\overrightarrow{abc}$	\overleftarrow{abc}	$\overleftarrow{abc}$
\overbrace{abc}	$\overbrace{abc}$	\undervrace{abc}	$\underbrace{abc}$
\sqrt{abc}	$\sqrt{abc}$	\sqrt[3]{abc}	$\sqrt[3]{abc}$

何と言えば良いのでしょう？とにかく見て下さい。

入力	出力
\{abc}choose{def}	$\binom{abc}{def}$
\{abc}brack{def}	$\left[\begin{smallmatrix}abc\\def\end{smallmatrix}\right]$
\{abc}brace{def}	$\left\{\begin{smallmatrix}abc\\def\end{smallmatrix}\right\}$

2.14.2 ギリシャ文字

ギリシャ文字の一覧表です。[23]

ギリシャ文字				
文字の名前	大文字		小文字	
	出力	入力	出力	入力
alpha	A		α	\alpha
beta	B		β	\beta
gamma	Γ	\Gamma	γ	\gamma
delta	Δ	\Delta	δ	\delta
epsilon	E		ϵ , ε	\epsilon, \varepsilon
zeta	Z		ζ	\zeta
eta	H		η	\eta
theta	Θ	\Theta	θ , ϑ	\theta, \vartheta
iota	I		ι	\iota
kappa	K		κ	\kappa
lambda	Λ	\Lambda	λ	\lambda
mu	M		μ	\mu
nu	N		ν	\nu
xi	Ξ	\Xi	ξ	\xi
omicron	O		o	o(英語の o と同じ)
pi	Π	\pi	π,ϖ	\pi, \varpi
rho	P		ρ,ϱ	\rho, varrho
sigma	Σ	Sigma	σ,ς	\sigma, \varsigma
tau	T		τ	\tau
upsilon	Υ	Upsilon	υ	\upsilon
phi	Φ	\Phi	ϕ,φ	\phi, \varphi
chi	X	\Chi	χ	\chi
psi	Ψ	\Psi	ψ	\psi
omega	Ω	\Omega	ω	\omega

[23] 大文字で抜けているのは英語の大文字と同じだからです。

2.14.3 関数

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

最後に挙げた mod に関してはもう一つ以下のような形が用意されています。

入力 `\pmod{n}`

出力 $\pmod{n}$

2.15 数学のテクニック

2.15.1 行列を作る。

行列を作るには、数式モードで、**array** 環境を使わなければなりません。**array** 環境は、**tabular** 環境と同じ使い方をします。

入力

```
\[ A =
\left( \begin{array}{ccc}
 6 & -3 & -7 \\
-1 &  2 &  1 \\
 5 & -3 & -6
\end{array} \right) \]
```

出力

$$A = \left(\begin{array}{ccc} 6 & -3 & -7 \\ -1 & 2 & 1 \\ 5 & -3 & -6 \end{array} \right)$$

2.15.2 = の位置を揃える方法。

「=」の位置を揃えるには、**eqnarray** 環境で、揃えたい等号を、「&」で挟んでおけばよいでしょう。例えば、

入力

```
\begin{eqnarray*}
  x &=& (y+z)^2\\
    &=& y^2+2yz+z^2
\end{eqnarray*}
```

出力

$$\begin{aligned} x &= (y+z)^2 \\ &= y^2+2yz+z^2 \end{aligned}$$

となります。

2.15.3 数式モードでの書体

数式モードでしか使えない大文字の筆記体というのがあります。

入力

```
\cal ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

出力

$$\mathcal{ABCDEFGHIJKLMNOPQRSTUVWXYZ}$$

2.15.4 新しくコマンドを作る

新しくコマンドを作るには、\newcommand というコマンドを使います。このコマンドは、TEX がもともと持っているコマンド以外であれば、どんなものでも作ることができます。例えば、物理でベクトルを表すのに太字体の記号を用います。太字体にするには\boldmath Dとしてやればいいのですが、たくさんある時などはいちいち打つのは大変ですから、以下のようにして新しく太字体の記号をつくっておきます。

```
\newcommand{\bmD}{\mbox{\boldmath $D$}}
```

とすることによって新しくコマンドを作ることができます。では、説明していきます。\newcommand は新しいコマンドをつくる事を宣言するコマンドです。\newcommand{\bmD} とすれば、新しいコマンド\bmD をつくれ、ということを意味します。その新しいコマンドの内容として、{\mbox{\boldmath D}}としておきますと、「数式モードで文字を斜体の太字体にする」というコマンドができあがります。

入力

```
\newcommand{\bmD}{\mbox{\boldmath $D$}}
\[ \bmD=D_1+D_2+D_3 \]
```

出力

$$\boldsymbol{D} = D_1 + D_2 + D_3$$

2.15.5 コマンドの変更

ところで、数式モードでのアルファベットは、斜体で出力されます。一方、rot や div といった演算子記号は、普通ローマン体で書かれています。そこで、演算子記号を数式モードで、rot や div といったものをローマン体で出力するには、\def というコマンドを使います。\def は TEX が持っているコマンドを再定義するコマンドで、\def に続くものを定義します。例えば、

```
\def\mathop {\rm rot}\nolimits
```

とすると、数式モードでアルファベットをローマン体にすることができます。ここで、\mathop というコマンドは、数式モードで使うすべてのコマンドや環境を指します。\nolimits というコマンドは、lim や$\sum$などは、上や下に記号がつきますが、そうではなくて、その記号の上や下に何も記号がつかない、あるいは、つけないという意味です。したがって、\mathop {\rm rot}\nolimits は、数式モードで、$\rm rot$とすると、ローマン体の rot が出力されます。あとは、\it や\rm を組み合わせて、いろいろな文字や記号を作ってみましょう。

入力

```
\def\rot{\mathop {\rm rot}\nolimits}
\newcommand{\bmE}{\mbox{\boldmath  $E$}}
```

```
\newcommand{\bmB}{\mbox{\boldmath  $B$}}
\newcommand{\bmr}{\mbox{\boldmath  $r$}}
電磁誘導のファラデーの法則
\[ \rot{\bmE}({\bmr},t)+
\frac{\partial {\bmB}({\bmr},t)}{\partial t}=0 \]
```

出力

電磁誘導のファラデーの法則

$$\mathrm{rot}\,\boldsymbol{E}(\boldsymbol{r},t)+\frac{\partial \boldsymbol{B}(\boldsymbol{r},t)}{\partial t}=0$$

2.16 複雑な数式を少々

入力

```
\def\dfrac#1#2{{\displaystyle \frac{#1}{#2}}}
\begin{eqnarray*}
  f(x) =
  \left \{ \begin{array}{ll}
   x\sin \dfrac{1}{x} & (x\ne 0)\\
   0                  & (x=0)
  \end{array} \right.
\end{eqnarray*}
```

出力

$$f(x)=\left\{\begin{array}{ll} x\sin\dfrac{1}{x} & (x\neq 0)\\ 0 & (x=0)\end{array}\right.$$

入力

```
\begin{eqnarray*}
  I^{2} &=& \int\!\!\!\int e^{-r^{2}} r\,drd\theta \\
        &=& \int _{0}^{\infty }re^{-r^{2}}dr\int_{0}^{\pi /2}d\theta \\
        &=& \frac{\pi }{2}\left[ -\frac{1}{2}e^{-r^{2}} \right]_{0}^{\infty }\\
        &=& \frac{\sqrt{\pi }}{4}
\end{eqnarray*}
```

出力

$$\begin{aligned} I^2 &= \iint e^{-r^2} r\,drd\theta \\ &= \int_0^{\infty} re^{-r^2}dr \int_0^{\pi/2} d\theta \\ &= \frac{\pi}{2}\left[-\frac{1}{2}e^{-r^2}\right]_0^{\infty} \\ &= \frac{\sqrt{\pi}}{4} \end{aligned}$$

入力

Schr\"odinger 方程式

```
\[ \left[ -\frac{\hbar^2}{2m}\nabla^2+V(r) \right]  u =Eu \]
```

出力

Schrödinger 方程式

$$\left[-\frac{\hbar^2}{2m}\nabla^2+V(r)\right]u=Eu$$

入力

運動方程式

```
\newcommand{\bmx}{\mbox{\boldmath  $x$}}
\newcommand{\bmF}{\mbox{\boldmath  $F$}}
\[ m\ddot{\bmx} = \bmF = - \nabla V(\bmx)
\]
```

出力

運動方程式

$$m\ddot{\boldsymbol{x}}=\boldsymbol{F}=-\nabla V(\boldsymbol{x})$$

入力

角運動量の同時固有値問題

```
\begin{eqnarray*}
\left\{ \begin{array}{l}
\hat{J}^2 |J,M \rangle = j(j+1)\hbar|J,M \rangle \\
\hat{J}_z |J,M \rangle = m\hbar |J,M \rangle
\end{array} \right.
\end{eqnarray*}
\[ \left[ -\frac{\hbar^2}{2m}\nabla^2+V(r) \right]  u =Eu \]
```

出力

角運動量の同時固有値問題

$$\begin{cases} \hat{J}^2|J,M\rangle = j(j+1)\hbar|J,M\rangle \\ \hat{J}_z|J,M\rangle = m\hbar|J,M\rangle \end{cases}$$

2.17　さらに例題

数学でナブラ ∇(\$\nabla\$) の２乗 (∇^2) をラプラシアンと呼びます。その記号を、$\triangle$(\$\triangle\$) で表します。

入力

```
\[ \nabla ^{2}=\triangle =\frac{\partial ^{2}}{\partial
x^{2}}+\frac{\partial ^{2}}{\partial y^{2}}+\frac{\partial
^{2}}{\partial z^{2}}\]
```

出力

$$\nabla^2 \equiv \triangle = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$$

さらに、演算子記号として、ダランベルシャンというのがあります。これは、４元ベクトルのラプラシアンです。$\Box$(\$\Box\$) という記号で表します。数式で書くと、

入力

```
\[ \Box = \frac{\partial ^{2}}{\partial x^{2}}+\frac{\partial ^{2}}
{\partial y^{2}}+\frac{\partial ^{2}}{\partial
z^{2}}-\frac{\partial ^{2}}{c^{2}\partial t^{2}}\]
```

出力

$$\Box = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} - \frac{\partial^2}{c^2 \partial t^2}$$

となります。

相対論では、t, x, y, zの文字を使わずに、$x^0 = ct, x^1 = x, x^2 = y, x^3 = z$を使って座標を表しますので、先ほどの式は、

入力

```
\[ \Box =\frac{\partial ^{2}}{(\partial {x^1})^{2}}+\frac{\partial ^{2}}
{(\partial {x^2})^{2}}+\frac{\partial ^{2}}{(\partial
{x^3})^{2}}-\frac{\partial ^{2}}{(\partial {x^0})^{2}}\]
```

出力

$$\Box = \frac{\partial^2}{(\partial x^1)^2} + \frac{\partial^2}{(\partial x^2)^2} + \frac{\partial^2}{(\partial x^3)^2} - \frac{\partial^2}{(\partial x^0)^2}$$

となります。

線形代数で、「直和」、「直積」を表すのに、$\oplus$(\oplus) と$\otimes$(\otimes) を用います。

入力

```
\[ 直和:\; W=W_1 \oplus W_2 \oplus \cdots \oplus W_n\]
\[ 直積:\; W=W_1
\otimes W_2 \otimes \cdots \otimes W_n\]
```

出力

直和： $W = W_1 \oplus W_2 \oplus \cdots \oplus W_n$

直積： $W = W_1 \otimes W_2 \otimes \cdots \otimes W_n$

2.18　エラーの対処

2.18.1　エラーの表示

サンプルファイルとして、abc.tex を作ります。

入力

```
\documentstyle{jarticle}
\begin{document}
\[ \left\{ \begin{array}{c}
x'=ax+by\\
y'=cx+dy
\end{array} \right. \]
\end{document}
```

このファイルを jlatex にかけて、コンパイルしますと、

出力

```
cc2000(83)% jlatex abc.tex
This is BigTeX, C Version 2.99 - j1.7e (no format preloaded)
(abc.tex
LaTeX Version 2.09 <24 May 1989>
(/NF/local/Solaris2J/lib/tex/macros/jarticle.sty
Document Style `jarticle' <18 Dec 88>.
(/NF/local/Solaris2J/lib/tex/macros/jart10.sty)) (abc.aux) [1] (abc.aux)
Output written on abc.dvi (1 page, 456 bytes).
Transcript written on abc.log.
```

と表示された時は、正しくコンパイルされたことになります。

正しい例を出しましたが、どこか間違っている例を出しましょう。

ここからはまず読んで、それから実際にやってみて下さい。

入力

```
\documentstyle{jarticle}
\begin{document}
\[ \left\{ \begin{array}{c}
x'=ax+by\\
y'=cx+dy
\end{array} \right \]
\end{document}
```

さきほどの abc.tex のところで、6 行目の`\right.` の「.」をはずして見ます。そして、コンパイルしますと、

出力

```
cc2000(84)% jlatex abc.tex
This is BigTeX, C Version 2.99 - j1.7e (no format preloaded)
(abc.tex
LaTeX Version 2.09 <24 May 1989>
(/NF/local/Solaris2J/lib/tex/macros/jarticle.sty
Document Style `jarticle' <18 Dec 88>.
(/NF/local/Solaris2J/lib/tex/macros/jart10.sty)) (abc.aux)
No file abc.aux.
! Missing delimiter (. inserted).
<to be read again>
                   \edef
\@latexerr #1#2->\edef
                       \@tempc {#2}\expandafter \errhelp \expandafter {\@tem...

\]->\relax \ifmmode \ifinner \@badmath
                                      \else $$\fi \else \@badmath \fi \igno...
l.6 \end{array} \right \]

?
```

と、プロンプトが入力待ちの状態で止まります。これが間違った時に表示される**エラー表示**です。それでは、重要な部分だけを解説していきましょう。

出力

```
cc2000(83)% jlatex abc.tex
This is BigTeX, C Version 2.99 - j1.7e (no format preloaded)
(abc.tex
LaTeX Version 2.09 <24 May 1989>
(/NF/local/Solaris2J/lib/tex/macros/jarticle.sty
Document Style `jarticle' <18 Dec 88>.
(/NF/local/Solaris2J/lib/tex/macros/jart10.sty)) (abc.aux)
No file abc.aux.
```

1 行目から 8 行目までは、エラーとは関係ありませんので飛ばします。

出力

```
! Missing delimiter (. inserted).
```

ここからが、エラーを表示しているところです。これは、区切り文字が見つからないという事です。つまり、対になるべきところがなっていないという事です。次に、

出力

```
\@latexerr #1#2->\edef
```

これは、エラーの種類を表しています。つまり、latexerr とあるように、コマンドの使い方が間違っている、という事です。そして、その場所が、

出力

```
1.6 \end{array} \right \]
```

6 行目のところにあります、という意味です。
最後の行に「?」が表示されて、コンパイルの作業が中断していますので、「x」と打ってみて下さい。そうすると、

出力

```
? x
No pages of output.
Transcript written on abc.log.
```

と表示されてるはずです。この後は、Mule を立ちあげて、間違っていろところを直していきましょう。

2.18.2 ちょっと違うやり方

先ほどは「?」の後に、「x」を打ちましたが、それ以外にも次のようなものがあります。

s, または r これは、エラーがあっても構わず最後までコンパイルをします。従って、いっぱいエラーがあると思われるファイルは使用しない方がいいでしょう。

h これは、英語で何が間違っているかを教えてくれます。

e このキーを打つと、自動的に mule が立ち上がり、かつエラーを起こしている行の先頭にカーソルが出るようになっています。

みなさん、それぞれ試してみて下さい。

2.18.3 エラーの種類

ここではいくつかのエラーの表示を見てみます。

- `! Missing $ inserted.`
  1. 数式環境でエラーをしています。
- `! Undefined control sequence.`
  1. 定義されていないコマンドを使ったか、
  2. コマンドの後に空白がない、
  3. 余計なところで「\」がついてしまったためにコマンドとしてみなしてしまっている。
- `! \begin{eqnarray} ended by \end{document}.`
  1. `\begin`、`\end` の対応関係がおかしい。つまり、どちらかが多かったり、ぬけている。
- `! Missing } inserted.`
- `! Missing { inserted.`
  1. 括弧の対応が正しくありません。
- `Runaway argument?`
  1. `\section` において、閉じ括弧を忘れている。
  2. `\verb` で、はさんでいる記号が違う。など、コマンドの使い方が間違っている。
- `! Extra alignment tab has been changed to \cr.`
  1. 表を宣言した時に、指定した列の数よりソースにある列の数の方が多い。
  2. 列の最後に改行マーク「 `\\` 」が抜けている。
- `! Environment ?????? undefined.`
  1. 環境の命令の綴りが違っている。

などなど、エラーはたくさんありますので列挙するのはやめます。とにかく、エラーを起こしている行番号を頼りに直していって下さい。自分の勘違いやタイプミスがほとんどです。あきらめずに一つずつ直していきましょう。

もし、エラーの対処についてわからなければ、LaTeX の達人に作っている自分のファイルとそのファイル名の拡張子が `.log` のファイルの両方を見てもらいましょう。

2.19 部分印刷する方法

さて、完成した文章に少し手を加えて書き直した時など、最初から印刷するのは、時間と紙がもったいないと思います。そこで、部分印刷するには`jdvi2kps`コマンドを次のようにして使います。

`cc2000(3)% jdvi2kps -f` 開始ページ `-t` 終了ページ `foo.dvi > foo.ps`

としますと、開始ページから終了ページまでの PostScript ファイル foo.ps が出来上がります。

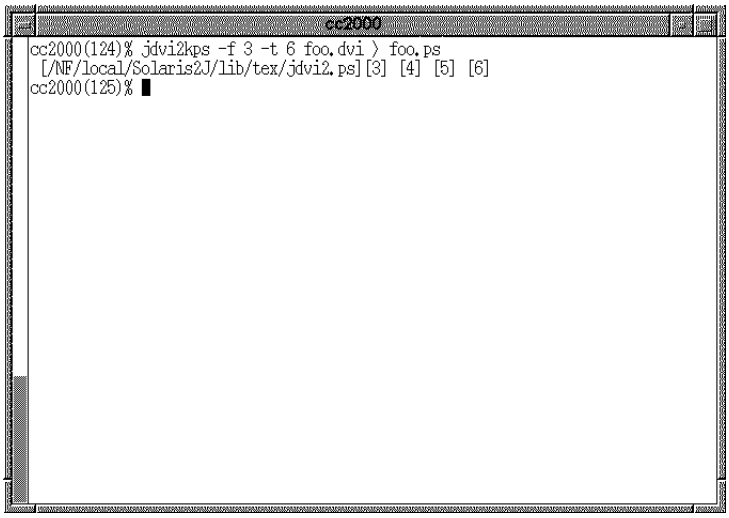


図 2.9 dvi ファイルから部分だけ取り出す方法

ここで、一旦 PostScript ファイル を確認してみましょう。プロンプトで、
`ghostview foo.ps` と打ってみて下さい。そうすると、

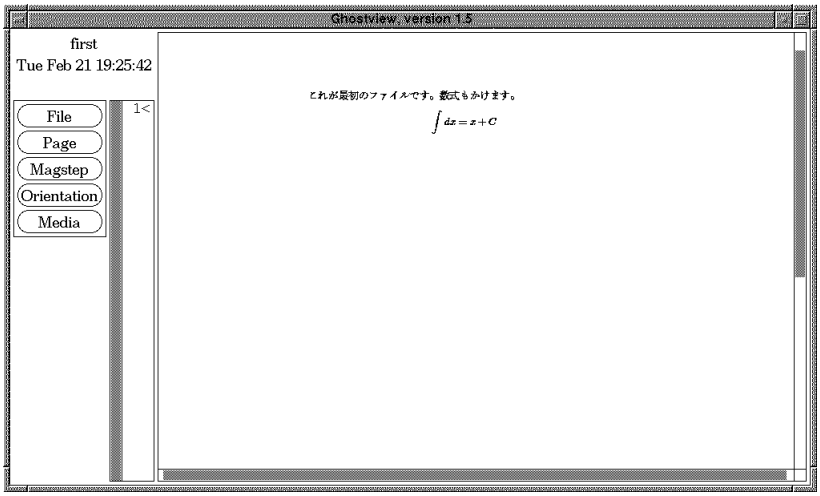


図 2.10 PostScipt 形式のファイルの確認

このような画面が現れてきます。これで、PostScript 形式のファイルを見ることが出来ます。これで良ければプロンプトで、

```
lpr -Pcspr01 foo.ps
```

とすれば、プリントアウトされます。

また、xdvi の画面でプリントボタンをクリックすると、次のような画面が現れて、

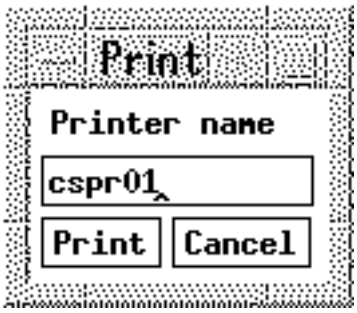


図 2.11 プリントボタン

と表示されたところをドラッグして、「**Current page**」のところでマウスボタンを離すと、そのページだけを印刷することができます。

2.20 自分の命令 (マクロ)

LaTeX では自分だけの命令 (マクロ) を作ることができます。これは既存の命令を組み合わせて定義し作ります。

2.20.1 簡単な命令を作ってみよう

いま用紙の中央に次のように書こうとします。

京都産業大学

これを実現するためには

入力
```
\begin{center}
京都産業大学
\end{center}
```

とします。
これと同じ働きをする\KSU という命令[24]を作ります。そのためには文書ファイルにこの命令を使うより以前のところでつぎのように\KSU を定義しておきます。

```
\newcommand{\KSU}{\begin{center}京都産業大学\end{center}}
```

ファイルのこれ以降の場所で\KSU と書けば

```
\begin{center}京都産業大学\end{center}
```

と書いたのと同じ働きをします。\newcommand については、第 2.15.5節を参照して下さい。

2.20.2 引数を持っている命令の作り方

さっきの\KSU を「京都産業大学」をセンタリングするだけでなく、ほかの単語もセンタリングできるようにバージョンアップしてみましょう。
次のようにします。

```
\newcommand{\namae}[1]{\begin{center} #1 \end{center}}
```

このようにすればこれ以降、\namae{...}と書けば (... のところに何か語を書くんですよ) 真ん中に単語がレイアウトされます[25]。ここで[1] は引数の数、#1 は引数の入る場所です。引数とは、そのマクロに対応する文字列のことです。だから、... が引数に当たります。

2.20.3 マクロの名前の付け方

このときの名前は、「\namae 」のような英字も、「\名前 」のような漢字も使うことができます[26]。しかし大文字と小文字は区別されます。
また記号や数字を含む名前も作ることができません。

[24] 何でも構いません。好きなものをつくって下さい。
[25] これってただの center 環境ですね
[26] あまり漢字はお薦めできません

2.21 ファイルを分けて LaTeX を使う方法

このガイドのように分量のあるものを作るとき、1 つのファイルとして作業しているとエラーも見つけにくく、処理に時間がかかってしまいます。そこで「小さなまとまりごとに文書ファイルを作る」という考え方が出てきます。LaTeX では`\input{...}`を使って実現することができます。例として、

```
\input{first.tex}
```

とすると、その場所に`first.tex` のファイルの中身を書いたことになります。

これを使って先程の`\KSU` を自分専用のファイルに入れましょう。まずエディタを立ち上げ、`macro.tex` というファイル名のファイルをつくって、その中に`\KSU` の定義を書き込んでおきます。そしてこの`macro.tex` というファイルを作業しているディレクトリに入れておきます。そして使いたい文書ファイルの中で次のように`\input{...}`を使ってこのファイルを読み込みます。

```
\input{macro}
```

そうすればそこの場所に`macro.tex`[27] のファイルの内容を書いたことになり、これ以降の場所で`\KSU` のマクロが使えるようになります。

2.22 標準以外のスタイルファイル

普通スタイルファイルは、(j)article,(j)report,(j)book の 3 つでしたが、以下にあげるような方法で、いろいろなスタイルファイルが使えます。

2.22.1 日本語 LaTeX 定番スタイル集の使い方

京都産業大学がライセンス契約しているインプレスの日本語 LaTeX 定番スタイル集の使い方を紹介します。このスタイル集を使うことによって、`jarticle` や`jreport` というスタイルだけでなく英文レター、はがき、時刻表、レポート用紙、アドレス帳、論文などのさまざまなスタイルの文書を作ることができます。使い方の例として英文レターの場合は次のようにします。

入力

```
\documentstyle{iletter}
\begin{document}
\begin{letter}
:
:
\end{letter}
end{document}
```

のようにドキュメントスタイルの指定を変えてやります。またそれだけではなく、そのスタイルで使いやすいような命令, 環境 (ここでは`\begin{letter}...\end{letter}`のところです) が入っていますので、各情報処理教室の棚やロッカーにあるマニュアルを見て有効に利用して下さい。

[27] 例文では.tex を省略しました。同じファイルがなければ構いません。

2.22.2 京都産業大学独自のスタイルファイル

ccreport.sty

これは主に、UNIX ガイドを書く時に、使われたドキュメントスタイルです。ただし、jreport 専用となります。このスタイルを使おうと思ったら、以下のようにする必要があります。

入力

```
\documentstyle[ccreport]{jreport}
\begin{document}
(内容)
\end{document}
```

と、オプションのところに ccreport としておけば使えます。そうすると、次のようなスタイルにかわります。

入力

```
\documentstyle[a4j,ccreport]{jarticle}
\begin{document}
\chapter{ chapter のタイトル}
\section{ section のタイトル}
\subsection{ subsection のタイトル}
\subsubsection{ subsubsection のタイトル}
\paragraph{ paragraph のタイトル}
\subparagraph{ subparagraph のタイトル}
\end{document}
```

出力

第 1 章

chapter のタイトル

1.1 section のタイトル

1.1.1 subsection のタイトル

subsection のタイトル

paragraph のタイトル

subparagraph のタイトル

これ以外にも「 figure 1.1 崩壊曲線」が 「**図 1.1** 崩壊曲線」に変わります。つまり、英語だったのが日本語に変わりました。一度お試しあれ。

2.23 参考文献

参考文献を作るやり方は 2 種類あって一つは **thebibliography** 環境を使う方法と、もう一つは **BibTeX** という実行コマンドを用いて`my-bib.bib` というファイルを別個に作って、本文に組み込むという方法があります。後者に関しましては、ここでは取り上げないで、前者についてだけ説明していきます。**thebibliography** 環境は、卒業論文や修士論文とかで、文章中に`\cite{`ラベル`}`というコマンドを用いて参考文献を引用することができます。例えば、文章中では、

入力

```
参考文献であげられているすずき~\cite{すずき}や奥村~\cite{奥村}が一般的に適しています。特に野寺~\cite[pp.141--]{野寺}は大変役に立つでしょう。
```

のようにしていきます。考え方としては第 2.8節で説明した相互参照と同じです。あとは、**thebibliography** 環境を用いることにより、参考文献を列挙しておくことができます。例の文章では以下のようにしておきます。

入力

```
\begin{thebibliography}{99}
\bibitem{野寺}
野寺　隆志．共立出版刊 ．第二版 .1994
\bibitem{奥村}
奥村　晴彦．技術評論社刊．1994
\bibitem{すずき}
すずき ひろのぶ．オーム社刊．1991
\end{thebibliography}
```

ちょっと説明しましょう。`\begin{thebibliography}{99}`の`{99}`は 99 個まで文献を列挙することができます、という意味です。ここの数値はいろいろかえることができます。ただし、この数値以上に`\item` の個数が上回るとエラーになります。また、文章中にある`\cite{野寺}`と`\bibitem{野寺}` が対応します。出来上がりは、以下のようになります。

出力

参考文献であげられているすずき [3] や奥村 [2] が一般的に適しています。特に野寺 [1, pp.141–][28]は大変役に立つでしょう。

[28] ここのページ数は適当に書いています。

参考文献

[1] 野寺　隆志. 共立出版刊 . 第二版 .1994

[2] 奥村　晴彦. 技術評論社刊. 1994

[3] すずき ひろのぶ. オーム社刊. 1991

2.24　新しく font を定義する

2.4.10節で注意しましたようにイタリック体で`\Huge` の font を出力したい時には、以下のような文をプリアンブルに付け加えて下さい。

入力

```
\newfont{\tencmmi}{cmmi10 scaled\magstep5}
\newfam\cmmifam
\textfont\cmmifam=\tencmmi
\newcommand{\Hugeit}[1]{{\fam\cmmifam\relax#1}}
```

とすれば、新しい font が出来上がります。ただし、ASCII 版の LaTeX はこの新しい font を作る領域が狭いため二つまでしか定義できませんので、注意して下さい。

では説明していきましょう。まず、

`\newfont`

というコマンドで新しい font を作ることを宣言します。その次の

`\tencmmi`

は新しく作るフォントのコマンド名です。何でも構いませんがこのコマンド名の中に数字を入れてはいけません。最後の括弧は、どういう font をどの大きさで表示しますか、ということを表しています。ここではイタリック体でしたから

`cmmi10`

という font を使います。この `cmmi10` というフォントはイタリック体で太字体の 10pt のフォントを表しています。そして、

`scaled\magstep5`

は文字の倍率の指定で、`\magstep5` は 1.2^5 倍で表示します。つまり、10pt の 1.2^5倍の大きさのフォントを作ります。ここの数値は 1 ～ 5 までの整数が使えます。ちょうど`\magstep5` は Huge の大きさになっています。

2 行目は新しく定義した font の領域を確保するためのものです。その次の行で

`\textfont`

は、普通の文章の中で使うことを宣言します。もし数式とかで上つきや下つきで用いたい場合はここのところを

`\scriptfont`

とします。この時 10pt の文字を使いますと大き過ぎますので 7pt のものを使いましょう。さらに上つきのさらに上つきなどの場合は

`\scriptscript`

として 5pt のものを使えば見栄えの良い形になります。

最後の行で、大きさが Huge でイタリック体の文字を表示するためのコマンドを定義しています。

入力	倍率=大きさ
倍率の指定なし	10pt
`\magstephalf`	10pt×$1.2^{0.5}$ ≃11pt
`\magstep1`	10pt×1.2^{1}=large
`\magstep2`	10pt×1.2^{2}=Large
`\magstep3`	10pt×1.2^{3}=LARGE
`\magstep4`	10pt×1.2^{4}=huge
`\magstep5`	10pt×1.2^{5}=Huge

図 2.12 10pt の時の場合の対応

使い方は本文で以下のようにします。

入力

```
$\Hugeit H$uge style {\Huge H}uge style
```

出力

$\Hugeit H$uge style {\Huge H}uge style

となります。$ではさむのは「約束事」として下さい。

以下の表にあるフォントはほんの一部です。

msam10	数学特殊記号その 1	msbm10	数学特殊記号その 2
eufm10	ドイツ語の花文字	eufb10	ドイツ語の花文字の太字体
eusm10	大文字スクリプト体	eusb10	大文字スクリプト体の太字体
rsfs10	Ralph Smith's Formal Script 体	wncyr10	キリル体

図 2.13 フォントいろいろ

これらのフォントがどんなものかを確かめるには

`/NF/local/Solaris2J/lib/tex/macros/testfont.tex`

に testfont.tex というファイルがありますからこれを自分のディレクトリーにコピーしてきて、

入力

```
cc2000(44)% tex testfont.tex
```

と tex でコンパイルしますと、

入力

```
This is TeX, Version 3.1415 (C version 6.1)
(testfont.tex
Name of the font to test =
```

と表示されますので、すかさず、試したいフォントの名前を入力して下さい。そうしますと、

入力

```
Now type a test command (\help for help):)
*
```

と表示されてまた止まります。ここで、以下のように入力します。

入力

```
\table\bye
```

すると、

出力

```
[1]
Output written on testfont.dvi (1 page, 5892 bytes).
Transcript written on testfont.log.
```

と表示されて、試したいフォントの文字のテーブルが出来上がります。あとは、

入力

```
cc2000(45)% xdvi testfont.dvi &
```

とすれば、プレビューアが立ち上がりますので確認することができます。

2.25 cc 環境で使える TEX のパッケージ

現在 cc 環境で使える TEX は ASCII 仕様のものと NTT 仕様のものが入っています。これらが使えるものは以下のようになっています[29]。日本語が混じっているものも cc 環境でコンパイルすることができます。

[29] BIBTEX などで書かれたものについては説明しません。

- TEX で書かれているもの
- LATEX で書かれているもの
- $\mathcal{AMS}$TEX で書かれているもの
- $\mathcal{AMS}$LATEX で書かれているもの

具体的には cc2000 のプロンプトからコマンドを入力すると、以下のものがコンパイルすることができます。

- 英語版の TEX および $\mathcal{AMS}$TEX··· `tex tex-file`
- 日本語版の TEX および $\mathcal{AMS}$TEX
  - ASCII 仕様の日本語 TEX、$\mathcal{AMS}$TEX··· `jtex tex-file`
  - NTT 仕様の日本語 TEX、$\mathcal{AMS}$TEX··· `ntex tex-file`
- 英語版の LATEX··· `latex tex-file`
- 日本語版の LATEX
  - ASCII 仕様の LATEX、$\mathcal{AMS}$LATEX··· `jlatex tex-file`
  - NTT 仕様の LATEX、$\mathcal{AMS}$LATEX··· `nlatex tex-file`

例えば、ASCII 仕様の LATEX で書かれたファイルを仮に`aa.tex` としまして、

入力

```
\documentstyle[12pt,a4j]{jarticle}
\begin{document}
Hamiltonian $H$がパラメータ${\bf R}(t)$を介して時間に依存して断熱的に変化する系を考察する。系の状態を記述する波動関数は Schr\"{o}dinger 方程式
\begin{eqnarray}
  i\hbar \frac{d}{dt} |\psi(t)\rangle &=& H({\bf R(t)}) |\psi(t)\rangle
\end{eqnarray}
で表される。
\end{document}
```

このようなソースファイルを作ります。そこで、`aa.tex` をコンパイルしますと、

入力

```
cc2000(20)% jlatex aa.tex
```

出力

```
This is BigTeX, C Version 2.99 - j1.7e (no format preloaded)
(aa.tex
LaTeX Version 2.09 <24 May 1989>
(/NF/local/Solaris2J/lib/tex/macros/jarticle.sty
```

```
Document Style ‘jarticle’ <18 Dec 88>.
(/NF/local/Solaris2J/lib/tex/macros/jart12.sty))
(/NF/local/Solaris2J/lib/tex/macros/a4j.sty) (aa.aux) [1] (aa.aux)
Output written on aa.dvi (1 page, 824 bytes).
Transcript written on aa.log.
```

の表示されれコンパイルが完了します。ます。また、NTT 仕様の LaTeX で書かれたファイルをab.tex としまして、

入力

```
\documentclass[12pt,a4paper]{jarticle}
\begin{document}
Hamiltonian $H$がパラメータ${\bf R}(t)$を介して時間に依存して断熱的に変化する系を考察する。系の状態を記述する波動関数は Schr\"{o}dinger 方程式
\begin{eqnarray}
  i\hbar \frac{d}{dt} |\psi(t)\rangle &=& H({\bf R(t)}) |\psi(t)\rangle
\end{eqnarray}
で表される。
\end{document}
```

ab.tex をコンパイルすると、

入力

```
cc2000(42)% nlatex ab.tex
```

出力

```
This is JTeX, Version 1.6, based on TeX Version 3.1415 (C version 6.1)
(ab.tex
LaTeX2e <1994/12/01>
(/NF/local/Solaris2J/lib/texmf/tex/jlatex2e/base/jarticle.cls)
(/NF/local/Solaris2J/lib/texmf/tex/jlatex2e/base/j-article.cls
Document Class: j-article 1995/01/31 v1.2y Standard JLaTeX document class
(/NF/local/Solaris2J/lib/texmf/tex/jlatex2e/base/j-size12.clo
(/NF/local/Solaris2J/lib/texmf/tex/jlatex2e/base/jresize12.clo))) (ab.aux)
[1] (ab.aux) )
Output written on ab.dvi (1 page, 1332 bytes).
Transcript written on ab.log.
```

のように表示されてコンパイルが完了します。詳しいガイドにつきましては、さまざまな本が出版されておりますので、そちらを参考にして下さい。

2.25.1 LaTeX2ε への対応

1995 年半ばに LaTeX2ε に対応した日本語版が NTT-jTeX1.6 がリリースされて cc2000 にもインストールされています。しかし、日本語の解説書が見当たらない状況なので、LaTeX2ε に関しましては各自で勉強していただくことになると思います。

`/NF/local/Solaris2J/lib/texmf/doc/latex2e`

にドキュメント (英語) がありますので、参考にして下さい。

2.26 最後に

LaTeX から AUCTeX へ

第第 3 章章で、AUCTeX が紹介されていますが、これは、いちいちコマンドや環境を手で打つのは面倒になってきて、何とか楽をしたい、という人にはお推めのものです。

さいごに

さて、ここまで進まれてきた人は、一通りのことが出来るようになっているものと思われます。ここに書いたものだけでも、ある程度のレポートや卒論は書けるはずです。

ところで、ここを何とかしたいとか、こういう環境は使えないのか、といったことは後ろにあげておきました参考文献を片手に取り組んでいただきたいと思います。このガイドは、最低限の事しか書かれていません。足りないところは自分で補っていただけたら幸いかと存じます。

第 3 章

AUCTEX

AUCTEX は Mule 上で動作し，TEX を書く作業を支援してくれます．この機能を利用することによって，LATEX の文書を編集する作業がずいぶん便利になります．特にエラー修正の効率はかなり上がると思われます．これらの機能は初心者ほど効果があると思いますので，まずいきなり AUCTEX を使って LATEX を試してみると言うのも良いかも知れません．但し Mule に機能を追加して利用していますので，Mule の操作に慣れている必要があります．

3.1 AUCTEX で LATEX 生活が変わる

AUCTEX を使うとコマンドひとつで

```
\documentstyle[a4j]{jarticle}
\begin{document}

\end{document}
```

等を書いてくれます．カーソルは\begin の行と\end の行の間に置かれます．もちろん document 環境以外の任意の環境もコマンドひとつで書いてくれます．これで\begin{}\end{}や{}の非対応のエラーから解放されます．

AUCTEX を使うと Mule の中からコンパイルとプレビューと印刷のコマンドを実行する事が出来るようになります．これで Mule と kterm を行ったり来たりする必要が無くなります．

AUCTEX を使うとエラーメッセージが日本語で表示されます．その上コマンドひとつで次々とテキストのエラー箇所にカーソルが移動します．これで意味が全く理解できなかった TEX のエラーメッセージからのデバッグ作業の苦痛は取り除かれます．

以上は AUCTEX の機能のほんの一部ですが，それらを使うだけでも LATEX の編集作業が地獄から天国に一変する事でしょう．それではまず 3.2で AUCTEX を起動してから，3.3以降でこの節で紹介したすばらしいコマンド達の使い方を説明します．これらのコマンドはいわゆる必修項目です．といってもそのために覚える必要のあるコマンドはたったの 6 つ (実質 4 つ) です．その後，3.7ではその他の便利なコマンドをざーっと紹介します．

3.2 AUCTEX の起動

さて，AUCTEX モードの起動からはじめましょう．cc 環境では AUCTEX は cc2000 の Mule から使うことができます．拡張子が.tex のファイルを cc2000 の Mule で読み込むと，自動的に AUCTEX が起動するよ

うに設定されています．そして AUCTEX は原稿を見て適切なモード[1]に入ります．新しいファイルを呼び出した時など，その情報が無い場合は, デフォルトとしてLaTeX モードが起動します.

3.3 C-c C-e

AUCTEX のコマンドのほとんどは C-c で始まります．そしてそれに続く C-e の ‘e’ は ‘environment の e’ と覚えましょう．このコマンドは LATEX の様々な環境, つまり \begin{なんとか}\end{なんとか} を対話的にセットしてくれます.

具体的な作業に進みましょう．新しい TEX の原稿の書き始めには, まず document 環境をセットするためにこのコマンドを入力することになるでしょう．すると AUCTEX が

```
Master file: (default this file)
```

と聞いてきます．これは新規ファイルに対して最初にコマンドを入力した時だけ聞いてくる質問で，3.7で説明する分割編集の際に参照される情報です．ここではとりあえず<return>キーだけを押しておきましょう．すると，次に

```
Environment type: (default document)
```

と聞かれるので，使いたい環境名をタイプします．document 環境をセットしたいので,

```
Environment type: (default document) document
```

とタイプします.

丸カッコの中に「デフォルト（default)」としてdocument が用意されている旨が表示されています．この様に AUCTEX ではコマンド入力時には, 大抵, 良く考えられた「デフォルト」が用意されています．そのデフォルトを採用する場合には, 何もタイプせずに<return>キーを押すだけで良いのです．以降では，積極的にこの機能を使ってゆく事にします.

先に進みましょう．document 環境をセットするための質問はまだ続いています.

```
Document style: (default jarticle)
```

ここで style 名をタイプします．デフォルトはjarticle になっています．デフォルトを採用する場合は単に<return>キーを押しましょう．jreport が良ければそうタイプして<return>キーを押します．最後に Options を質問してきますので, A4 の日本語ならa4j とタイプします．さらに原稿中で PostScript の図を取り込むなら , に続けてepsbox とタイプしておきます.

```
Options: a4j, epsbox
```

[1] AUCTEX には LaTeX モード，JTeX モード，JSliTeX モード，TeX モード等があります.

これで document 環境がセットされます.
　他の環境もセットしてみましょう．例えば equation 環境なら

```
Environment type: (default itemize) equation
label: eq:abel
```

のように対話が進みます．他にも, table 環境なら

```
Environment type: (default equation) table
Float to: htbp
Caption: すごい表
Label: tab:wao!
Center: (y or n) y
Psition:
Format: |c||lll|
```

のように AUCTEX とユーザーの間で対話が進みます.

3.4　C-c {

　C-c { で{}がセットされます．これで LaTeX で多用される中括弧の閉じ忘れのデバッグ作業から解放されます.

3.5　C-c C-c

　このコマンドを使うことで, Mule の中からコンパイル, プレビュー, 印刷, スペルチェック等のコマンドを実行する事が出来ます．‘c’ は ‘command の c’ です.
　LaTeX の原稿の編集が一段落したら, 原稿を LaTeX に通します．さぁ，コマンド C-c C-c を入力しましょう．原稿を保存せずにこのコマンドを入力した時には，まずセーブするかどうかを聞いてきます.

```
Save file? /home/kyoin1/matsuura/tex/foo.tex ? (y or n) y
```

y とタイプすると

```
Command: (default jLaTeX)
```

の様に AUCTEX は聞いてきます．上の例では(default jLaTeX) となっています．つまり，jlatex コマンドがデフォルトに設定されています．デフォルトで良ければ何もタイプせずに<return>を押します．コンパイルを実行すると

```
Type ‘C-c C-l’ to display results of compilation.
```

と教えてくれます．コンパイルの様子を見たければ AUCTEX の言う通り C-c C-l と入力します．現在走らせているコンパイルなどを途中で中止させたい場合には C-c C-k と入力します．

コマンドは jLaTeX 以外にも多数用意されていますが，一つ一つのコマンドのスペルを覚える必要はありません．Command: と聞かれている時にスペースキーを押すと，全コマンド一覧の表示とコマンドの補完を同時に行なってくれます．その中から目的のコマンド名を探して，Command: の行にタイプして<return>を押します．AUCTEX はデフォルトの値を適切に選んで設定してくれるので，コマンドをタイプする事なく<return>キーを押す事が一番多いでしょう．View やPrint を選んだ時には，AUCTEX は実際に実行するコマンドを確認してきます．View の場合のやりとりを見てみましょう．

```
Command: (default View)
```

LaTeX のエラーやワーニングが無ければデフォルトが View になっている事でしょう．その場合，<return>キーを押します．

```
View command: xdvi foo
```

の状態で待ってくれます．そのまま<return>キーを押せばxdvi が起動します．例えば B4 用紙サイズで表示したい場合には，ここで

```
View command: xdvi -paper b4 foo
```

のようにコマンドに修正を加えてから<return>キーを押します．また，Print の場合は

```
Command: (default View) Print
Printer: (default cspr01)
```

のようにプリンターの名前を聞いてきます．印刷させたいプリンターが cspr01 でない場合はプリンター名をタイプします．次に実行されるコマンドを確認してきます．

```
Print command: jdvi2kps -d 400 foo | lpr -Pcspr01
```

問題なければそのまま<return>を押します．

3.6 C-c ‘

さあ天国に行きましょう．デバッグコマンドです．このコマンドは覚える必要さえありません．なぜならコンパイル中にでエラーが見つかった場合，AUCTEX は親切なことに “エラーがあったよ．C-c ‘ を押してね.” と教えてくれます．次のエラーに進むときも C-c ‘ です．’ ではなく ‘ であることに注意しましょう．

3.7 その他の機能

前節までの機能で LaTeX 生活環境は格段に向上します．ですから初めて AUCTEX を使おうという人はこの節をざっと目を通すだけにするか，あるいは読むのをやめてしまっても良いでしょう．

3.7.1 部分的なコンパイル

AUCTEX を使うと部分的にコンパイルをする事ができます．複雑な equation 環境や table 環境になると，コンパイルとプレビューを何度も繰り返しながら修正する事になります．その度に毎回ドキュメント全体をコンパイルするのはばかばかしいですね．そこで適当な領域だけを部分的にコンパイルすることにします．コンパイルしたい領域の先頭で C-<Space> を入力して，その領域の後尾の次の所にカーソルを置いて C-c C-r と入力します．するとコマンド入力待ちになるので，前述の手順 (3.5節参照) に従ってコンパイルします．複雑な equation 環境や table 環境等を部分コンパイルしたいときには，部分コンパイルしたい環境の中にカーソルを置いた状態で C-c . と入力するだけで領域としてその環境が設定されます．そこで続けて C-c C-r と入力します．他にも，編集している節を領域として設定するためのコマンド C-c * があります．これも続けて C-c C-r と入力する事で，部分コンパイルをすることが出来ます．さらに編集しているバッファ全体を部分コンパイルするためのコマンド C-c C-b もあります．部分コンパイルの結果をプレビューするときも C-c C-b または C-c C-r と入力します．おそらく View がデフォルトに設定されているので，<return>を押せば xdvi が起動します．部分コンパイルの結果を印刷したければPrint とタイプして<return>を押します．

3.7.2 ドキュメントの分割編集

LaTeX のドキュメントを複数のファイルに分けて編集していく方法がよく取られます．AUCTEX を使うとその手の LaTeX のドキュメントの管理が楽になります．ここでは例として，原稿全体は親ファイルと子ファイル (下の例では section1.tex,section2.tex,section3.tex) から構成されるとします．まず親ファイル (parent.tex) を作ります．3.3節で説明したように，document 環境を入力する時に master file の名前を聞いてきますが，ここではそのまま<return>を押します．そしてそのファイルには

```
\documentstyle[a4j]{jarticle}
\begin{document}
\input{section1}
\input{section2}
\input{section3}
\end{document}

% Local Variables:
% mode: japanese-latex
```

```
% TeX-master: t
% End:
```

のように子ファイルを読み込むコマンドを書いておきます．次に子ファイルを作ります．新規に小ファイルを開いて最初に AUCTEX のコマンドを入力する時にも, 例のごとく`master file` の名前を聞いてきます．ここでは，次のように親ファイル名の (拡張子.tex は付けずに) フルパスをタイプします.

```
Master file: (default this file) ~/tex/parent
```

こうしておけば，いちいち親ファイルに戻らずに, 子ファイルを編集している状態で C-c C-c と入力することで, ドキュメント全体のコンパイルが出来るようになります．そして編集中の思考錯誤時には，分割して編集しているメリットを生かして C-c C-b で子ファイルだけの部分的コンパイルを行なうとよいでしょう (3.7.1参照)．子ファイルを編集中に親ファイルを見たくなった時には C-c ^ と入力すれば親ファイルが開かれます.

3.7.3 アウトラインマイナーモード

3.7.2の様にドキュメントを分割して作成する方法の他にも, ひとつのファイルに全ての文章を書く方法もあります．しかしそうすると,「さて第 2 章第 3 節を修正しようか」という時に, そこを画面に表示させるために C-v を何回も押して…おっとっと，行き過ぎた！<ESC> v，なんてことになりそうです．AUCTEX でアウトラインマイナーモードを使えばそんな面倒を避けられます.

それでは<ESC> x outline-minor-mode と入力してアウトラインマイナーモードに入りましょう．アウトラインマイナーモードに入ったら, 最初に C-c C-o C-t と入力して文章部分 (ボディ) を隠しておきましょう．\chapter や\section の行 (ヘッダ行) だけが表示されます．この状態で，編集したい節の\section の所までカーソルを持っていっておいて C-c C-o C-e と入力すると, その節の隠されていた文章部分だけが表示されます．その節の編集を終えたら C-c C-o C-c と入力して再び文章を隠しておきます．ちなみに原稿全てのボディを表示させるコマンドは C-c C-o C-a です.

編集中での各節を部分コンパイルする時は C-c *を使って領域を指定しましょう (3.7.1参照)．コマンドは上に挙げた他にも色々あります．その一覧を表 3.1に示しておきます.

3.7.4 LaTeX マクロの入力

AUCTEX を使うと様々な LATEX マクロを完全に覚えていなくても使う事ができるようになります．これで，LATEX の教科書が横に無くてもある程度 LATEX のコマンドが使えるようになります．その機能の一つが LATEX マクロの補完です．補完といえば, <Tab>キーを使った tcsh の補完機能がお馴染みでしょう．AUCTEX での LATEX マクロの補完のコマンドは<ESC> <Tab> です．候補が複数あると補完は途中で止まります．その場合は, その続きを何文字か入力してさらに<ESC> <Tab> を入力します．候補がひとつになるまでこれを続けて，ひとつになった所で LATEX マクロが最後まで補完されます．マクロを対話式に入力する方法を使うことで, 覚えていない LATEX マクロを入力することもできます．C-c C-m で対話的に LATEX マクロの入力を求められます．マクロ名を途中までしか覚えてなくても, <Tab>キーや<Space>キーを押せばマクロ名を補完してくれたり, マクロ候補一覧を表示してくれたりするので安心です．\chapter{}や\section{}も LATEX マクロなのでしょうが，AUCTeX ではこれらの入力には別のコマンドが割り当てられています．C-c C-s がそれで，以下のように対話しながら，\chapter{}や\section{}の入力を行ないます.

表 3.1 アウトラインマイナーモードのコマンド一覧

key	binding
C-c C-o C-a	すべてを表示する
C-c C-o C-t	すべてのボディを隠す
C-c C-o C-o	カーソルの置かれている節以外のボディを隠す
C-c C-o C-q	カーソルの置かれている節のレベル以下をすべて隠す
C-c C-o C-e	カーソルの置かれている節を表示する
C-c C-o C-c	カーソルの置かれている節を隠す
C-c C-o C-l	カーソルの置かれている節以下のボディを隠す
C-c C-o C-k	カーソルの置かれている節以下のヘッダ行を表示する
C-c C-o C-d	カーソルの置かれている節以下をすべて隠す
C-c C-o C-s	カーソルの置かれている節以下をすべて表示する
C-c C-o TAB	カーソルの置かれている節直下のヘッダ行を表示する
C-c C-o C-b	同じレベルで一つ前の節のヘッダ行に移動する
C-c C-o C-f	同じレベルで一つ後の節のヘッダ行に移動する
C-c C-o C-u	一つ上のレベルの節のヘッダ行に移動する
C-c C-o C-p	カーソルの置かれている所より前方で，現在表示されている最も近いヘッダ行に移動する
C-c C-o C-n	カーソルの置かれている所より後方で，現在表示されている最も近いヘッダ行に移動する
C-c C-o C-h	コマンドヘルプを表示する

```
Select level: (default section) chapter
What title: すごい章
What label: cha:wao!
```

3.7.5 複数行のコメントの付け外し

AUCTEX を使うと連続した多数行にわたってコメント記号% を一度に付ける事ができます．コメント記号をつけたい領域の先頭で C-<Space>，後尾の次の行で C-c ; とすればその領域にわたって行頭にコメント記号が付けられます．コメント記号を外したい場合には，その領域の先頭で C-<Space>，後尾の次の行で C-u C-c ; とすればコメント記号が削除されます．

3.7.6 書体の指定

AUCTEX を使って書体設定の一括入力ができます．例えば bold 書体のコマンドを入力すると{\bf }とタイプされます．イタリック書体\it, emphasized 書体\em そして slanted 書体\sl についてはイタリック補正もセットされるという芸の細かさです．

一旦書体指定をセットした後で書体指定を別の書体に変更したくなった場合には, C-u C-c C-f <KEY> と入力します．<KEY>には変更後の設定したい書体に対応する, C-b C-i C-r C-e C-t C-s C-c のいずれかが入ります．

3.7.7 数式モードの支援

表 3.2 書体指定をセットするコマンド一覧

bold	C-c C-f C-b
italics	C-c C-f C-i
roman	C-c C-f C-r
emphasized	C-c C-f C-e
`typewriter`	C-c C-f C-t
slanted	C-c C-f C-s
SMALL CAPS	C-c C-f C-c
書体設定の消去	C-c C-f C-d

表 3.3 数式モードでの記号入力

ギリシャ文字

α	(alpha)	a
β	(beta)	b
δ	(delta)	d
ϵ	(epsilon)	e
ϕ	(phi)	f
γ	(gamma)	g
η	(eta)	h
κ	(kappa)	k
λ	(lambda)	l
μ	(mu)	m
∇	(nabla)	N
ν	(nu)	n
ω	(omega)	o
π	(pi)	p
θ	(theta)	q
ρ	(rho)	r
σ	(sigma)	s

τ	(tau)	t
υ	(upsilon)	u
χ	(chi)	x
ψ	(psi)	y
ζ	(zeta)	z
Δ	(Delta)	D
Γ	(Gamma)	G
Θ	(Theta)	Q
Λ	(Lambda)	L
Ψ	(Psi)	Y
Π	(Pi)	P
Σ	(Sigma)	S
Υ	(Upsilon)	U
Φ	(Phi)	V
Ω	(Omega)	O

記号

$\rightarrow$	(rightarrow)	C-f
$\leftarrow$	(leftarrow)	C-b
$\uparrow$	(uparrow)	C-p
$\downarrow$	(downarrow)	C-n
$\leq$	(leq)	<
$\geq$	(geq)	>
$\tilde{}$	(tilde)	~
∞	(infty)	I
$\forall$	(forall)	A
$\exists$	(exists)	E
$/$	(not)	!
$\in$	(in)	i
$\times$	(times)	*
$\cdot$	(cdot)	.
$\subset$	(subset)	{
$\supset$	(supset)	}
$\subseteq$	(subseteq)	[

$\supseteq$	(supseteq)	]
$\backslash$	(backslash)	\
$\setminus$	(setminus)	/
$\cup$	(cup)	+
$\cap$	(cap)	-
$\langle$	(langle)	(
$\rangle$	(rangle)	)
exp	(exp)	C-e
sin	(sin)	C-s
cos	(cos)	C-c
sup	(sup)	C-^
inf	(inf)	C-_
det	(det)	C-d
lim	(lim)	C-l
tan	(tan)	C-t
$\hat{}$	(hat)	^
$\vee$	(vee)	v

AUCTEX を使うと数式モードで利用される記号の入力が簡単になります．C-c ^ で LaTeX-math-mode になります．再び C-c ^ で LaTeX-math-mode から抜けます．数式モードでは「‘」(逆シングルクォート) と一文字の入力で，数式環境でよく使われる記号が展開されます．例えば ‘ a と入力すると\alpha がセットされます．表 3.3が数式モードのコマンド一覧です．

3.8 最後に

AUCTEX にはまだ他にも幾つかのコマンドがあります．また，カスタマイズについては説明しませんでした．それらについては AUCTEX のパッケージに附属の英語のマニュアル[2]や info ファイルを参照して下さい．英語はいやだ！という人には，ちょっと古いかも知れませんが，UNIXmagazine の'93 年 9 月号 (基本的な使用方法)，10 月号 (様々な機能)，11 月号 12 月号 (カスタマイズの方法) が図書館にあります．また，X 環境で Mule を使っている場合に限られますが，AUCTEX は Mule の (というより Emacs 19 の) 拡張機能の一つであるメニューバー機能に対応しています．(Japanese-) LaTeX-mode に入ると **LaTeX** という項目が追加され，そこに AUCTEX のコマンドのメニューが登録されています．

このマニュアルは WWW の学内 Home Page の「各種ドキュメント、情報検索サービス」にも載せてありますので，併せて御利用下さい．

[2] dviman というコマンドを用意しています．とりあえず，ターミナルで dviman -l とタイプしてみて下さい．

第 4 章

レポートシステム

レポートシステムは、あなたが出そうと思っているレポートを紙に印刷して学部事務室や先生に提出する代わりに、電子メイルで送るものです。電子メイルを扱う方法さえ理解していればレポートシステムを使う事が出来ます。

4.1 レポートシステムを使う前に

レポートをメイルで提出するという事は、レポートの内容をファイルに作成して、そのファイルをメイルとしてレポートシステム宛に送るという事を意味しています。

どのようなレポートでもいつでも闇雲にレポートシステム宛に送ればいいというものではありません。そのレポートが何という先生の何という課題のレポートなのかが判らないと意味がありません。すなわちあなたが出そうと思っているレポートにはあらかじめ担当の先生によって設定されたレポート名がある筈です。このレポート名を間違えたり登録されていないものだったりした場合、レポートシステムはそれを受け付けません。まずこれを注意して確認して下さい。

また、レポートには提出期限があります。この期限はあらかじめ先生によって課題ごとに決められたもので、この期限を過ぎた提出をレポートシステムは受け付けません。この期限も確認しておいて下さい。

提出期限内で先生による評価前であればレポートは何度でも訂正して再提出できます。逆に評価後の再提出をレポートシステムは受け付けません。

これらの条件を越えるような場合（期限後の提出、評価後の再提出）については先生に相談して下さい。

4.2 レポートを提出するには

レポートメイルの宛先は常に `report@cc.kyoto-su.ac.jp` です。レポートをメイルで送る時にはレポート名を示す名前をメイルのサブジェクトに付けてやります。

レポートはメイルとして送られる為、一般のメイルと同じ制限があります。例えば C や FORTRAN のプログラム演習で作成した実行モジュールなどのバイナリファイルはそのままでは送れません。レポートを読んで採点する先生の為に一行はある程度（アルファベット 70 文字、漢字なら 35 文字ぐらい）で改行した方がいいでしょう。(もちろん仕方なく長くなる場合を除きます。)

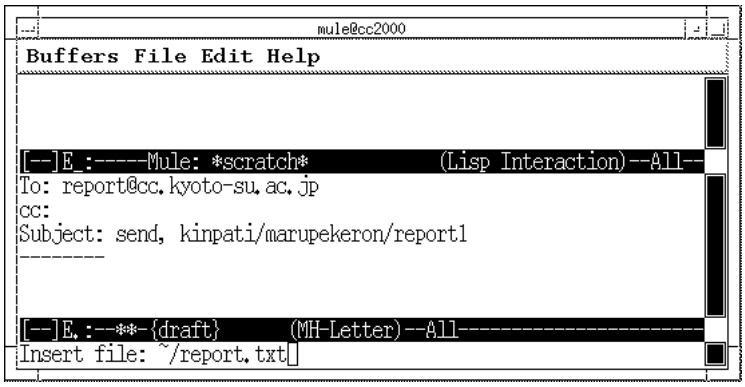
まずレポートの内容をファイルにまとめます。これ以降の例では `report.txt` という名前のファイルにレポートを書いたとしましょう。レポートファイルの先頭には誰のレポートか読んだ時にはっきり判るように学生証番号、氏名、課題の名前などを書いておく事を心がけましょう。

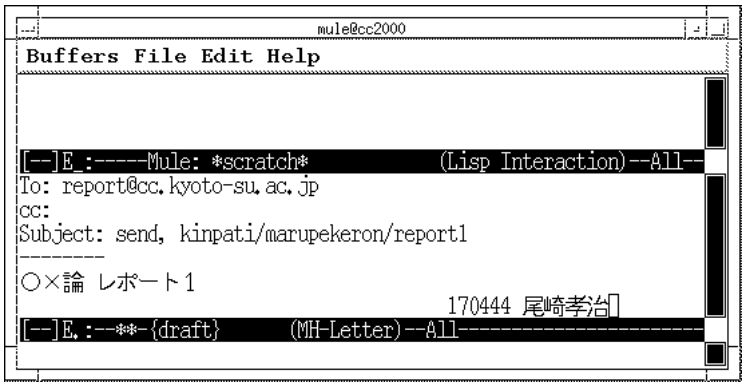
また、例では `kinpati/marupekeron/report1` というレポート名で提出する事にします。

4.2.1　そのレポートを初めて提出する場合

レポートのファイルをメールとして report@cc.kyoto-su.ac.jp 宛にサブジェクトをつけて送ります。サブジェクトに send 指定と、先生が指定するレポート名を書きます。具体的には以下のようにします。

あらかじめ作っておいた report.txt を読み込むためには C-x i[1]を使ってファイルをカーソルの位置に読み込みます。





提出すると受理されたかどうかのメールが返ってきます。

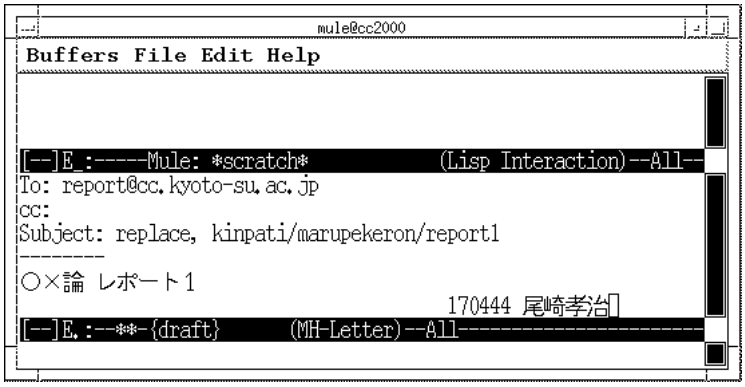
きちんと受理されたのなら「確かにあなたのレポートを受けとりました」とメイルで返ってきますし、失敗した時は後述のエラーメッセージがメイルで返ってきます。メイルが返ってこない時は何らかのミスで受理されていません。何回やっても駄目な時は先生に相談して下さい。

4.2.2　そのレポートを再び提出する場合

一度出したレポートも、期限前でしかも先生がまだ採点していなければ、再提出して以前のものと置き換える事が出来ます。

レポートのファイルをメイルとして report@cc.kyoto-su.ac.jp 宛にサブジェクトをつけて送ります。サブジェクトに今度は replace 指定と、先生が指定するレポート名を書きます。具体的には以下のようにします。

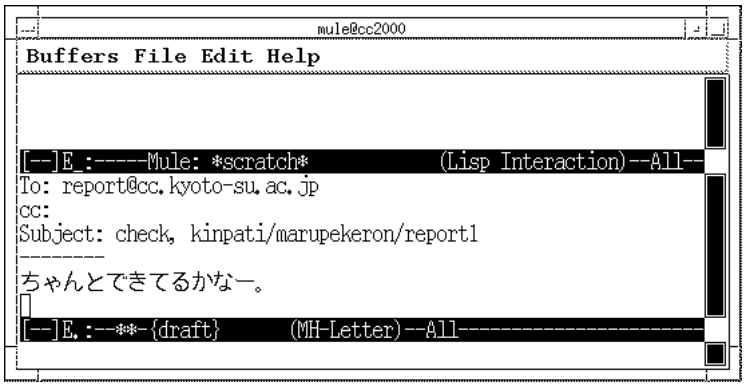
[1] コントロールキーを押しながら x、コントロールキーを離して i ですね。



初めての提出と同じように受理されたかどうかのメイルが返ってきます。

4.2.3 提出したレポートの確認

提出したレポートがどんなものだったか確認する事も出来ます。以下のようにサブジェクトに今度は `check` 指定と、確認したいレポート名を書いて `report@cc.kyoto-su.ac.jp` 宛にメイルを送って下さい。その際、メイルの中身は何でも構いません。



きちんと受理されたのなら以前に提出したレポートがメイルで返ってきます。何かおかしければ後述のエラーメッセージがメイルで返ってきます。

4.3 返ってくるメイルのメッセージ

- **このレポートは既に評価されています。**

  既にレポートに対して評価が行なわれてしまった為、受理できません。まだ期限以内なら担当の先生に相談して下さい。

- **確かにあなたのレポートを受けとりました。**

  初めての `send`、及び `replace` の場合で、きちんと受理されました。

- **あなたが提出しようとしたレポートはすでに提出されています。置き換えるなら `replace` で再提出して下さい。**

間違って上書きすることのないよう、二度目以降の send をした時の警告です。(この send コマンドは受理されていません)

- **これがあなたの提出したレポート です。**

  check で、前にそのレポートを出していた場合に現在受理しているものを送ります

- **あなたが確かめようとしているレポート は提出されていません。**

  check をしようとしたレポートがまだ提出されていません。

- **あなたの提出しようとしたレポート のコマンド が判別できませんでした。Subject に send、replace、check のいずれかを指定して下さい。**

  subject に send,replace,check 以外の物が指定してあります。確認してもう一度送り直して下さい。

- **あなたの提出しようとしたレポート は期限が過ぎていますので受け付けられません。**

  send、及び replace をしようとしたレポートの提出期限が過ぎていますので受理できません。

- **あなたの提出したレポート の宛先が見つかりませんでした。サブジェクト をお確かめの上もう一度送って下さい。**

  指定されたレポート宛先が見つかりませんでした。おそらく宛先を書き間違えていると思われます。もしくは宛先が存在しません。

- **あなたの提出したレポート にはサブジェクト が付いていなかった為どこ宛のレポート か識別できませんでした。もう一度送り直して下さい。**

  メイルにサブジェクトが付いていませんでした。サブジェクトを付けてもう一度送って下さい。

特殊な場合

- **あなたの送られたメイルからは日付が判別できませんでした。もう一度他のメイルシステムからお送り下さい。**

  メイルから日付が判別できませんでした。特殊な環境から送っていませんか? cc 環境から送ってみて下さい。

- **レポート システムがあなたのメイルを宛先に書き込むことが出来ませんでした。担当の先生に御連絡下さい。**

  メイルをシステム上処理できませんでした。まず起こりませんが、起こった時は速やかに担当の先生に御連絡下さい。(ディスクスペースが無くなっていると思われます。)

メイルが返ってこない場合

宛先を省略していませんか? report ではなく report@cc.kyoto-su.ac.jp として下さい。

第 5 章

Mathematica

5.1 Mathematicaってなあに？

Mathematica は Wolfram Research,Inc から発売されている数学とその応用のための汎用プログラムです。『数学を使って何かをやろう』と思っているけど『計算力に自信がない』というあなた。『数学を使って何かをやらなければならない』けど『面倒な計算は嫌だ』というあなた。そして、『数学を見てみたい』というあなたのためのプログラムです。

と、いっても何のことか解らない人も多いと思われるので、ともかく動かして試してみましょう。

5.1.1 ともかく起動、そしてやってみる！

ともかくやってみましょう。まず、2 号館の 4 階の 21 情報処理教室の csosf01 〜 csosf40 でのやりかたを例にして説明します。

まず csosf01 〜 csosf40 のうちにどれかに login し、ルートメニューの中のアクセサリメニューから「Mathematica」を選択します。また、ターミナルからコマンドとして `mathematica &`と打ち込んでも構いません。すると、

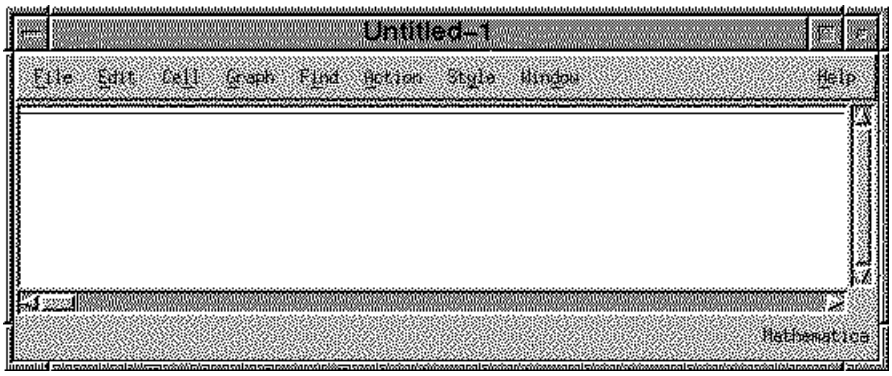


図 5.1 ノートブックのウインドウ

と、いうふうに、 Mathematica のノートブックというウインドウが新しく出てきます。そのノートブックが何かという説明は後回しにしましょう。ともかくやるのです。

先ず試しに簡単なかけ算をやってみましょう。4 <Space>4 <Shift>+ <return>と入力します。しばらくするとウインドウは以下のようになります。

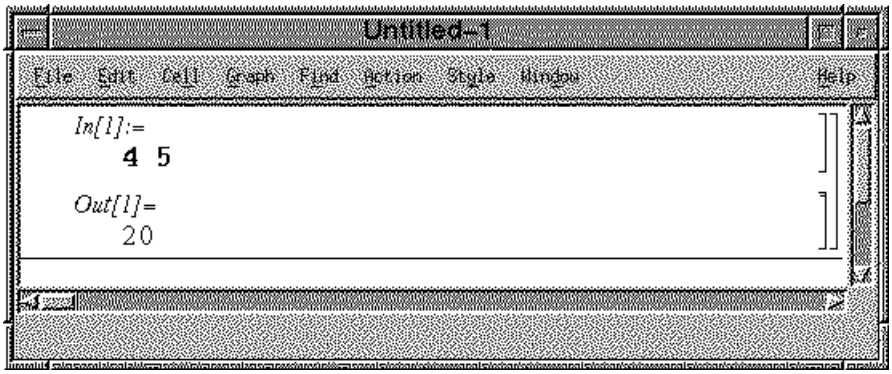


図 5.2 最初の簡単なかけ算

『こんな簡単な計算にこんなに時間がかかるのか!?』などとは驚かないで下さい。これは Mathematica を起動して最初の計算をやらせる時だけです。

さて、もう少し試してみましょう。今度は$\sqrt{10}$の計算をやってみましょう。

N[Sqrt[10], 40] <Shift>+ <return>と打ち込みます。入力する時は必ず、<Shift>+ <return>です。<return>だけでは駄目です。以下<Shift>+ <return>は省略します。つまりこれからは『 N[Sqrt[10], 40] と入力する』という表現を使います。

ともあれそうすると、

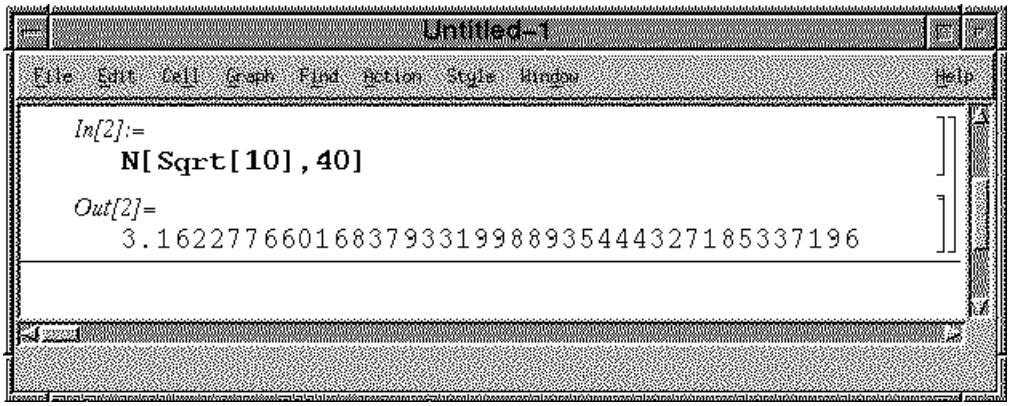


図 5.3 ちょっとした計算

となります。今度は素早く計算したはずです。

また、以下のようなグラフィックを描くことも可能です。

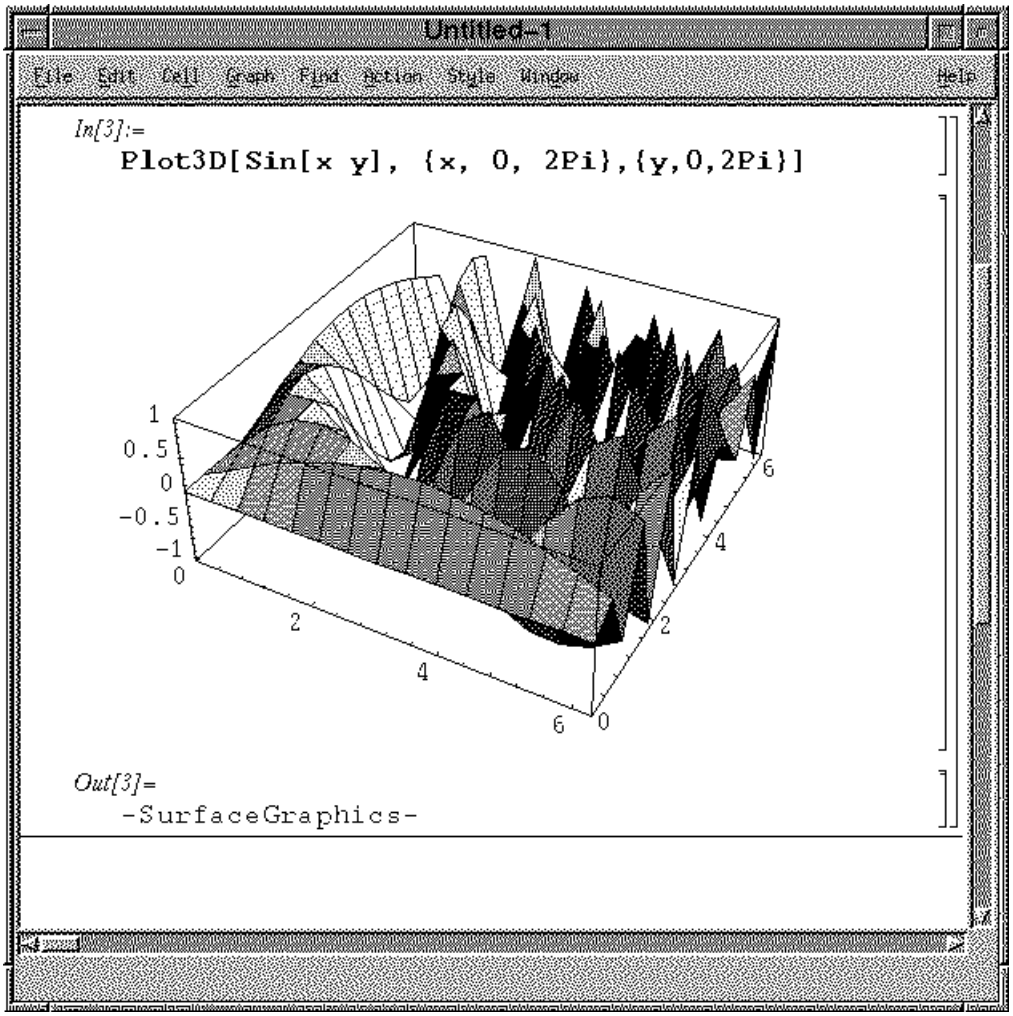


図 5.4 y = sin(xy) のグラフ

終了する時は mathematica の File メニューから Quit を選択します。

5.1.2 その他の場所の Mathematica

2 号館 21 情報処理教室の他に産大では以下の場所に行けば Mathematica を使うことができます。

1) 計算機科学研究所棟 3 階の C2 情報処理教室
直接コマンドラインで mathematica と入力するだけです。すると、csosf01 〜 csosf40 と同様にノートブックウインドウが出てきます。

2) 計算機科学研究所棟 3 階の C3 情報処理教室の NeXTStation 及びあちこちにあるマッキントッシュや Windows パソコン。
Mathematica のアイコンをダブルクリックするだけです。それでノートブックウインドウが出てきます。

5.2 ノートブックとコマンドラインについて

さて、先に、

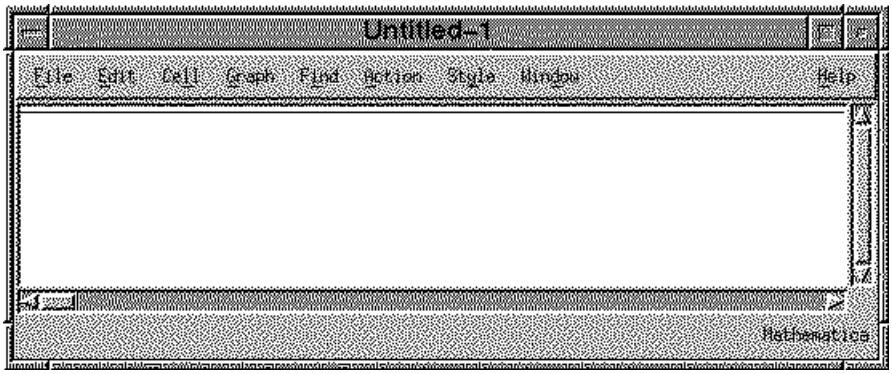


図 5.5 ノートブックのウインドウ

をノートブックウインドウと呼びました。これは実際のノートと同じような機能を持っていて Mathematica で行なったすべての入出力をまるごとファイルにしてセーブしておくことができます。もちろんそれをまたオープンすることもできます。
使い方も簡単でマウスでメニューから選択するだけです。

また、21 情報処理教室や C2 情報処理教室では、このようなノートブックではなくて直接コマンドラインで Mathematica を使うこともできます。その場合は math と入力します。ノートブックとは違って、最初の計算が極端に遅い、ということはありません。しかし 21 情報処理教室ではグラフィックが出ないという欠点もあります[1]。
コマンドラインで終了する時は Quit と入力します。

5.3 Mathematica の簡単な命令

1) : 数値計算

- 足し算: x+y
- 引き算: x-y
- かけ算: x*y または x y

```
_   In[n]:= 3 4     ⇐ 入力
    Out[n]:= 12     ⇐ 出力
```

- 割算: x/y
- 乗巾: xˆy

[1]が、グラフィックは PostScript で書かれているので、ファイルにしてユニックスの他のコマンドを使って表示することはできます。

- 厳密な計算:
  - 通常の電卓と違い、厳密な計算も行ないます。

```
In[n]:= 3/7     ⇐ 入力
Out[n]:= 3/7    ⇐ 出力
```

```
In[n]:= 3/7 + 1/3    ⇐ 入力
Out[n]:= 16/21       ⇐ 出力
```

- 近似値計算:
  - //N を使えば近似値で出ます。

```
In[n]:= 3/7 + 1/3 //N    ⇐ 入力
Out[n]:= 0.781905        ⇐ 出力
```

  - 実数値で入力しても近似値となります。

```
In[n]:= 3.0/7.0      ⇐ 入力
Out[n]:= 0.428571    ⇐ 出力
```

2) : 数学関数

数学関数は、普通の英語の名前がついていて大文字から始まり引数は [ ] の中に入れるようになっています。また、複数の引数をとる場合は引数と引数の間を「 , 」で区切ります[2]。例えば以下のような関数があります。

- 平方根（$\sqrt{(x)}$）: Sqrt[x]
- 自然対数 （$log_e(x)$）: Log[x]

厳密な計算、近似値計算 //N の使い方は 1) : 数値計算と同じです。

数学関数は、普通の英語の名前がついている訳ですから、逆に名前から探すこともできます。つまり、Sin[x] って、定義されているかな、と思った時に。

- In[1]:= ??Sin

とすると、Sin[z] gives the sine of z と出てきます。[3] つまり、 Sin[ ] も数学関数として組み込まれていて、

- In[1]:= Sin[3] //N
- Out[1]:= 0.14112

というふうになります。

3) : 代数解析

- Expand[ ]: 式を展開

[2] [x y] では [x*y] というふうにかけ算になってしまいます。
[3] これは次章でも説明します。

```
In[1]:= Expand[(x + y)^3]          ⇐ 入力
Out[2]:= x^3 + 3x^2y + 3xy^2 + y^3   ⇐ 出力
```

- Factor[ ]: 因数分解

```
In[1]:= Factor[x^3 + 3 x^2 y + 3 x y^ 2 + y^ 3   ⇐ 入力
Out[2]:= (x + y)^3                               ⇐ 出力
```

- Integrate[ ]: 積分
  - どの変数について積分するかの指定が、式の後に必要です。

```
In[1]:= Integrate[2 x, x]   ⇐ 入力
Out[2]:= x^2                ⇐ 出力
```

- D[ ]: 微分
  - どの変数について微分するかの指定が、式の後に必要です。

```
In[1]:= D[x^2 , x]   ⇐ 入力
Out[2]:= 2x          ⇐ 出力
```

4) : グラフィック

- Plot[f(x), {x, 0, 10}]: f(x) を x = 0 から 10 まで表示
- Plot3D[f(x, y), {x, 0, 10}, {y, 0, 10}]: f(x, y) を 3 次元で表示

例えば、 $y = \sin(x)$ というグラフを $x = 0$ から 2πまで表示したければ、Plot[Sin[x], x, 0, 2Pi] と入力します。$z = \sin(x + y)$ というグラフを $x = 0$ から 2π まで $y = 0$ から 2π 表示したければ、Plot3D[Sin[x + y], x, 0, 2Pi, y, 0, 2Pi] と入力します。

5.4 入出力一般

この章では（いろんな方法で） Mathematica を起動した後、どのように Mathematica を利用すれば良いか、について説明します。

直接入力する場合のコマンド 色々

1 ）前の結果の取り込み

- 最後に得られた結果： %
- k 回前に得られた結果： %%,,,,,%（ k 回）
- 出力 Out[n]:= で得られた結果： %n
  - 以下のように使います。

```
In[1]:= 3/7 + 1/3 //N    ⇐ 入力
Out[1]:= 0.781905        ⇐ 出力
In[2]:= %                ⇐ 入力
Out[2]:= 0.781905        ⇐ 出力
In[3]:= %% + 1           ⇐ 入力
Out[3]:= 1.781905        ⇐ 出力
In[4]:= %2 + 2           ⇐ 入力
Out[4]:= 2.781905        ⇐ 出力
```

2）変数

数値計算とはいえ、やたら長い数値を入力するのは面倒ですよね。少しでも楽をしましょう。

- 変数に値を代入： x = 値
  - 変数は任意のアルファベットが使えますが、前章の組み込み関数とかち合わないように、小文字を使いましょう。以下のようにして使います。

```
In[1]:= x = N[Pi, 10]    ⇐ 入力
Out[1]:= 3.145926535     ⇐ 出力
In[2]:= x                ⇐ 入力
Out[2]:= 3.145926535     ⇐ 出力
In[3]:= x 2 3            ⇐ 入力
Out[3]:= 18.8496         ⇐ 出力
```

3）情報を引き出す

前章でいろいろな関数、コマンドを列挙しました。しかしそれらを常に覚えていなければならない訳ではありません。

特に数学関数は Mathematica では sin(x) は Sin[x] などというように**普通の**英語の名前が付けられています。そこで、どんな関数があるか、どんな命令があるかは比較的調べ易くなっています。

例えば N の使い方を忘れてしまったとしましょう。そのときは、

- In[1]:= ??N ⇐ 入力

と、すると、

- Out[1]:= N[expr] gives the numerical value of expr. N[expr, n] does computations to n-digit precision

などと出てきて、使い方、内容などを表示します。

ファイル

Mathematica はエディターを内蔵していません。よって、直接、Plot3D[sin[x y], x, 0, 2Pi, y, 0, 2Pi] と打ち込んでしまったら気が滅入って来ます[4]。そこで、普通にファイルをつくってそれを Mathematica に読み込ませましょう。

[4] sin ではなくて Sin と書かなければなりませんよね。

- Mathematica に file を読み込ませる： <<filename
- Mathematica から file に expr を書き込む： expr >> filename
- Mathematica で file の内容を表示： !!filename
- エディタをたちあげて、filename という名前の file をつくって、中に「 N[Pi, 10] 」と、書いてあるものとして説明します。

```
In[1]:= <<filename          ⇐ 入力
Out[1]:= 3.145926535        ⇐ 出力
In[2]:= 2 4 >> filename     ⇐ 入力
In[3]:= !!filename          ⇐ 入力
Out[3]:= 8                  ⇐ 入力
```

もちろん N[Pi, 10] のように簡単な命令ではなく、複雑な命令、打ち間違え易い命令を入力する時に、この方法は便利なものとなります。
何回も出したいグラフィック、式などはこのやり方で file の形で取っておくことをお進めします。

印刷

ノートブックで作った結果に関しては簡単に印刷する事が出来ます。Mathematica の File メニューの下の方にある「Print...」を選択します。すると以下のようなウィンドウが表示されます。

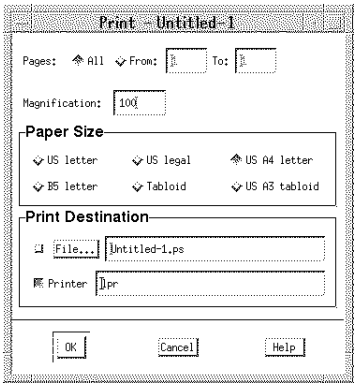


図 5.6 プリントウィンドウ

ここで「Printer」の設定に注意して下さい。まず「Printer」のすぐ左にある小さな四角のボタンが押されている状態[5]になっている事を確認して下さい。もしも押されていないようならマウスでこの四角のボタンをクリックして押して下さい。「Printer」のすぐ上の「File...」の左の小さな四角がもしも押されている状態ならばこれもクリックして今度は押されていない状態にしておきます。次に `lpr` と書いてある右側に以下のようにプリンタの名前を追加しましょう。

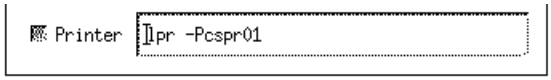


図 5.7 プリントウィンドウ（プリンタ設定の部分）

[5] 判りにくいですが、まわりより暗くなっているのが押されてへこんでいるという意味です。

例えば 21 情報処理教室であれば `lpr` の右に空白を一つ開けて `-Pcspr01` と書き足してプリンタの指定をします。どの部屋でどのプリンタを指定するべきかは 173ページの A.1.4を参照して下さい。上の図のようにできたら「OK」をクリックします。しばらくすればプリンタから結果が印刷されてくるでしょう。

ところでこれでは全文印刷されてしまいますが、ノートブックでは指定した部分だけを印刷する事も出来ます。*In* や *Out* などの行の右端に「]」などのようなカギ型の枠があると思います。印刷したい部分の右の枠をクリックすると、その黒くなった部分が選択された事になります。

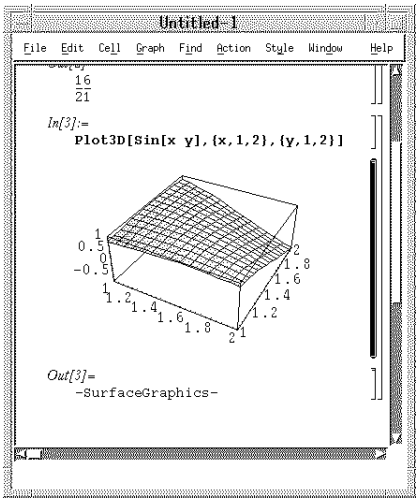


図 5.8 選択されたグラフ（グラフの右の黒い枠に注意）

Mathematica の File メニューの Print Selection... を選択すれば、図 5.6が表示されますので、先ほど紹介した印刷の手順にしたがって操作して下さい。今度は全文では無く選択した部分（上の例ではグラフ）だけが印刷されると思います。

5.5 さらに進みたい人には

このドキュメントは『ともかく Mathematica に触れてみよう』という人を対象に書かれています。ですから、ちゃんと Mathematica でプログラムを書こうという人や、実際に実験や研究で使おうという人は、Mathematica を作った本人、スティーブンウルフラムの書いた *Mathematica* というでっかい本がありますから[6]、それを参考に勉強を進めて下さい。

また、X ウィンドウ環境で利用している人だけに有効な手段ですが、ノートブックウィンドウにある「Help」というメニューで関数などの一覧を表示する事も出来ます。更に `mathbook` コマンドで組み込み関数のマニュアルが表示されます。

[6] C2 情報処理教室に何冊かあります。

第 6 章

GNUPLOT

6.1 概要

GNUPLOT は対話式の操作によって数値データや数学関数のグラフを作ることができるプログラムです。GNUPLOT は誰でも自由に無料で使用できます。

GNUPLOT の最大の特徴は何といってもグラフを出力するために利用できるプリンタなどの出力装置の種類が非常に多いことでしょう。また、GNUPLOT はいろいろなコンピュータシステムの上で動き、16 ビットの小さなパソコンでも、もっと大きな UNIX マシンでも同じように使用することができます。広く使われている物であれば、GNUPLOT が動作可能なシステムと対応可能なプリンタのリストの中に、あなたが使っている物が含まれていると期待してよいでしょう[1]。

GNUPLOT には、いろいろな形式のファイルとしてグラフィックスを出力する機能もあり、そうしたファイルを使ってたとえば LaTeX の文書の中に高品位のグラフを入れることができます。出来上がりはプロの仕事には及ばないかもしれませんが、数式やグラフの入った文書がかなり手軽に費用もかけずにできるのです。

GNUPLOT は線と点を使った 2 次元グラフと、メッシュや等高線を使った 3 次元グラフのいろいろなバリエーションを描くことができます。ただ、パイ・チャートや絵グラフを作ることはできませんし、スケールの異なる座標軸を使った複合グラフも直接には作れません。GNUPLOT は派手なものより、どちらかというと、地味なデータ分析のような実務に向いています。

GNUFIT というプログラムは GNUPLOT を拡張してフィッティングの機能を付け加えるものです。これは数式の中の未知パラメータを自動的に調整して与えられたデータに合わせる機能です。ただし、機能が低く限られた場合にしか使うことができません。GNUFIT が組み込まれている場合には、GNUPLOT を起動したときにオープニングメッセージの中にこのことが表示されます。

興味がわいてきたら、この後の説明に従って GNUPLOT をちょっと使ってみてください。 ここでは cc2000 上の GNUPLOT を ワークステーションやパソコンなどの X 端末から動かすことを想定していますが、21 情報処理教室にある DEC-3300 にも GNUPLOT が入っていて、cc2000 の GNUPLOT と同様にして利用できますから、21 情報処理教室があなたの行きつけの場所ならこちらを使ってください。その場合は、DEC-3300 の前に座ってマシンにログインして使います。ログインしたら自動的に開く DECterm もしくは kterm ウィンドウの中で作業してください。

これらの環境では、処理速度や操作性は十分ですし、かなり良い品質の作品を得ることができます。cc2000 や DEC-3300 で試した後、他のコンピュータでも GNUPLOT を使いたいと思ったら、`sandai.question`

[1] GNUPLOT 3.5 は UNIX、VAX/VMS、OS/2、MS-DOS（PC/AT 機）、MS-Windows、Macintosh などで動作する。NEC の PC-9801 で動作するバージョンもある。対応するプリンタや出力できるファイル形式については、GNUPLOT を起動してから `set term` と打てばリストが出る。

ニュースグループなどに投稿して情報を得ればよいでしょう。

6.2 GNUPLOT を初めて使う

では実際に GNUPLOT を使ってみましょう。X 端末からいつものように cc2000 にログインしてください。(DEC-3300 の GNUPLOT を使おうと思っている方は必要ありません。) エ？ どういうことかわからない？ それならとりあえず、明かりのついたどこかの情報処理教室に入って、そこにいる誰かに教えてもらってください。おっと！ パスワードを忘れた？ それなら計算機センターに行ってなんとかしてもらってください。それからここに戻って先を続けてください。

6.2.1 GNUPLOT の起動

cc2000 (または DEC-3300) にログインした後、一度 `kterm&` と打って (最後はいつもリターンキーを打つ。`kterm` の後に & を忘れないこと。)、もう一つのウィンドウを開いてください。そこで作業します[2]。元のウィンドウもいろいろ使いみちがありますから、そのままにしておいてください。新しいウィンドウで `gnuplot` と打って下さい。これで GNUPLOT が動きだし、オープニング・メッセージと、`gnuplot>` というプロンプトが現れます。オープニング・メッセージの中にこの GNUPLOT のヴァージョン情報とグラフィックスを出力する端末の種類の設定が書かれています。`Terminal type set to 'x11'` と表示されているのが端末の種類です。

万が一そうなっていなければ、`quit` と打って GNUPLOT を終了し、GNUPLOT の個人用の設定をチェックしなければなりません[3]。脚注のようにできたら、再び GNUPLOT を起動してください。

6.2.2 とりあえずグラフを！

うまく起動できたら、GNUPLOT のプロンプト `gnuplot>` に対して次のように入力してみてください。大文字と小文字は区別されますから、注意して使い分けてください。また、式の中の括弧には小括弧 ( ) を使います。

```
plot sin(x)/x
```

図 6.1 のように、ウィンドウが開いてグラフが描かれたでしょう。 これは $y = \sin x/x$ のグラフです。初めは x 軸 (横軸) の範囲は -10 から 10 までになっています。y 軸 (縦軸) の範囲は自動的に調整されます。この関数は　$x = 0$ では 1 で、x が 0 から離れるにしたがって振動しながら小さくなっていきます。今描かれたグラフでその様子がわかりますね。

この新しいウィンドウは決して勝手に閉じないでください。もし勝手に閉じてしまうと、GNUPLOT を起動しなおすまでグラフが描けなくなります。GNUPLOT に次のコマンドを与えるために、マウスの左ボタンで、プロンプト `gnuplot>` が出ている GNUPLOT のウィンドウを選んでください。

x 軸の範囲を変更するには、`set xrange [-20:20]` というように命令を与えますが、まず `set` とだけ打ってみてください。これでは不完全なので、GNUPLOT は何が足りないかを教えてくれます。あらためて `set xrange` と打つと、今度は `[` が足りないと教えてくれるでしょう。`:` のところをわざと間違えて、`set xrange [-20,20]` と打ってみてください。間違いがちゃんと指摘されますね。では、今度こそ正しいコマンドを入力してください。GNUPLOT が何も言ってこなければＯＫです。え？ グラフが変わらない？$\cdots$ 少し待って下さい。範囲を変えて描いたグラフはまもなく見ることができます。

[2] さらに、もしあなたが UNIX に少し慣れていれば、GNUPLOT の練習をするためのディレクトリを `mkdir` で作って `cd` でそこを作業ディレクトリ (カレントディレクトリ) とすることをお薦めします。

[3] ホーム・ディレクトリに `.gnuplot` というファイルがあればそれを消去し、`GNUTERM` という環境変数が設定されていればこれを取り外す。

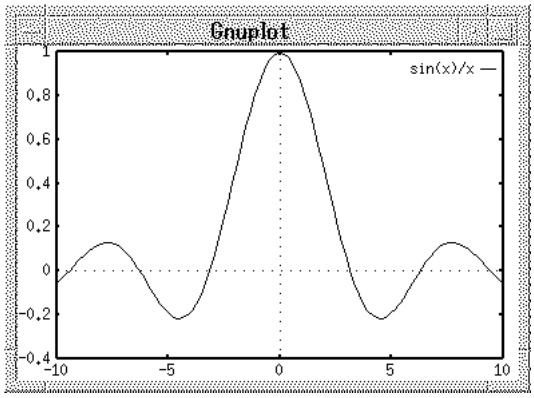


図 6.1 $y = \sin x/x$ のグラフ

x 軸の範囲がうまく変更されたかどうかは、`show xrange` と打ってやればわかります。`set` コマンドにはいろいろなバリエーションがあり、いろいろな設定を変更するのに使います。これは最も頻繁に使用される命令です。一方、`show` コマンドはいろいろな設定状態を調べるのに使います。

では、先程の関数をもう一度グラフにしてみましょう。`replot` と打つだけです。この命令は一番最近グラフにした関数をもう一度グラフにする命令です。縮めて `rep` でもだいじょうぶです。x 軸の範囲が 2 倍になったのが確認できますね。

6.2.3 コマンドライン編集

ここで、我々のような面倒くさがり屋のために GNUPLOT が持っているコマンドライン編集機能というものを紹介しておきましょう。これはタイプを楽にする機能ですが、tcsh シェルや VMS をお使いならお馴染みのものです。

まず ↑ キーを打ってみてください。前に入力したコマンドが呼び出されたのがわかりますね。もう一度打つとその前のコマンドに、さらにもう一度打つと前の前のコマンドに変わります。↓ キーは逆向きの働きをします。前に入力した `set xrange` ... を呼び戻し、←キー、→キー と delete キーを使って範囲を -50 から 50 に書き直してみましょう。delete キーは「左に向かって消していく」ことに注意してください。

書き直したら、それを GNUPLOT に新しいコマンドとして与えるためにリターンキーを押します。その後 `replot` でまたグラフを描いてみてください。

6.2.4 サンプル数の変更

set 命令の別のバリエーションを一つ試してみましょう。さきほど描かれたグラフはちょっと荒っぽく描かれているでしょう？これは、式の値を計算する点の数が少なすぎるのです。`show samples` と打てば、恐らく、`sampling rate is 100,100` と表示されるでしょう。これは、x 軸方向に 100 点しか計算していないことを表します。(前の方の 100 です。後の方の 100 は今は無視してください。)

では、計算する点の数を 200 にするために、`set samples 200` と打ってください。それからまた、`replot` とやってください。今度はずっとましになったと思います。

6.2.5 印刷しましょう

グラフを印刷する方法はシステムによっていろいろ変わってきます。ここでは、「PostScript」という形式のファイルに一旦出力してからそれをプリンタに送って印刷するという方法を説明しましょう。

まず、GNUPLOT で初めて作るグラフですからそれにふさわしいタイトルを入れた方がいいですね。ついでに目盛り線の描きかたもちょっと変えましょうか。次のように入力してください。

```
set title "Kinsan TOYAMA won the first output from GNUPLOT. Here it is!"
set grid
replot
```

タイトルはダブルクォート " で囲みます。タイトルの中身は好きなように書いてください。タイトル入りのグラフが描けましたね。目盛りは格子状 ... grid ... になったでしょう。

次に、グラフを PostScript ファイルに出力します。次のように入力してください。

```
set term postscript
set output "graph1.ps"
replot
set output
set term x11
```

最初の 2 行で、出力形式と出力先のファイル名を指定し、3 行目の `replot` で実際に出力しています。後の 2 行は、出力先を元に（つまり画面に）戻し、出力形式を x 端末に合わせ直しています。ちょっと面倒なようですが、ファイル名以外は `se te pos` というように縮めても認識してくれますし、間違えてもコマンドライン編集の機能で簡単に直せますからだいじょうぶです。

次は、いよいよ印刷です。一番始めに開けておいたウィンドウをマウスの左ボタンで選んでください。`cc2000(81)%` のような シェル・プロンプト が出ている ウィンドウ です。`ls` コマンドで `graph1.ps` というファイルがあることを確認してください。`ls` と打つだけですね。

では、このウィンドウで

```
lpr -Pcspr01 graph1.ps
```

と打って、（`cspr01` は 21 情報処理教室にあるプリンタ。使おうと思うプリンタに置き換えること！）あなたの作品がプリンタから出てくるのを待ってください[4]。

`set output` コマンドでファイル名のかわりに UNIX の「パイプ」というものを指定して、GNUPLOT の中から直接印刷することもできます。

```
set term postscript
set output "| lpr -Pcspr01"
replot
set output
set term x11
```

この場合は `replot` と打った段階でグラフがプリンタに送られます。`graph1.ps` のような PostScript ファイルは作られません。

[4] `lpr` コマンドの `-P` と プリンタ名の間は開けない。

6.2.6 オンラインマニュアル

GNUPLOT は操作画面上でマニュアルを読むことができます。(ただし、GNUFIT が組み込まれていてもフィッティングに関する項目は含まれていません。) 次は、このオンラインマニュアルの使い方を練習しましょう。印刷したマニュアルを作ったり、オンラインマニュアルを日本語にする方法については 6.5 節で説明します。

GNUPLOT のプロンプト `gnuplot>` に対して `help` と打って英語のオンラインマニュアルが出るのを確認してください。`Help topics available:` の下に、`copyright` とか `introduction` といった次に見ることのできる項目のリストも表示されているでしょう。

この中の `expressions` という項目を見るために `Help topic:` というプロンプトに対して `expressions` と打ってみてください。`expr` というように始めの何文字かだけでも大丈夫です。これで、GNUPLOT の式についての説明が出ます。今は説明の中身まで読む必要はありません。さらっとながめるだけで結構です。

さらに、`Subtopic of expressions:` に対して `functions` と打つと関数の説明が出ます。`?` と打つと項目リストを再表示させることができるということも覚えておいてください。

項目名を入れずにリターンキーのみ打てば 1 つ前のレベルに戻ります。リターンキーを何度か打ってプロンプトを `gnuplot>` に戻してください。

次に、`help sin` と打ってみてください。いきなり sin 関数の説明が出たでしょう。また、`help functions` とやってみてください。(この場合最後の `s` まで必要です。`function` では別の項目になります。) これで、オンラインマニュアルの使い方がわかりましたね。なお、`help` のかわりに `h` でもだいじょうぶです。

オンラインマニュアルは英語ですが、これが気に入ってきたら、複数のウィンドウを使って作業するといかにもスマートです。別のウィンドウでもう一つの GNUPLOT を起動して、片方にオンラインマニュアルを表示しながらもう一方で作業するのです。始めに開いた ウィンドウが シェル・プロンプト を表示するだけで遊んでいますが、そこに `kterm&` と打ってもう一つ ウィンドウ を作り、そこで GNUPLOT を起動するのが良いでしょう。新しく起動した GNUPLOT で オンラインマニュアル を表示し、今使っている方でこの後の作業を続けてください。

6.3 GNUPLOT の終了と再開

ちょっと疲れましたね。(え? 疲れちゃいない? じゃあ疲れたということにしてください。) GNUPLOT を一度終了して一休みしたい ...

6.3.1 作業の保存

では GNUPLOT を終了しましょう。でも、ちょっと待った! 今まで行った作業は GNUPLOT を終了したら何も残らないのでしょうか? いいえ、ちゃんと残す方法があります。

`save "gplot1.plt"` と打ってください。`gplot1.plt` という名前のファイルができて、この中に現在の GNUPLOT の状態が記録されます。この種のファイルを GNUPLOT のマクロファイルといいます。これでもう、いつ GNUPLOT を終了してもだいじょうぶです。本当にだいじょうぶかどうかは、次に GNUPLOT を起動したら最初に調べてみることにします。

GNUPLOT を終了するために `quit` と打ってください。`exit` でも同じ意味です。Control-D (コントロール・キーを押しながら D を押す) でも終了できます。終了したら、念のために `ls` コマンドで `gplot1.plt` ファイルがちゃんとあることを確認してください。

他の GNUPLOT も終了し、ウィンドウも全部閉じてログアウトしていただいて結構です。

6.3.2 GNUPLOT の再開

ゆっくり休みましたか? では、再び cc2000 にログインして宿題をかたづけましょう。cc2000 にログインしたら、`ls` コマンドで前に作ったマクロファイル `gplot1.plt` があるのを確認してください。もしなければ、ディレクトリが違うのでしょうから、前に居たディレクトリに移る必要があります。カレントディレクトリ（作業ディレクトリ）に `gplot1.plt` があるのを確認したら、前と同じ様にして GNUPLOT を起動してください。

オンラインヘルプも見たいなら、ウィンドウをもう一つ開いて、そちらでも GNUPLOT を起動すればよいでしょう。

では、GNUPLOT のプロンプトに対して

```
load "gplot1.plt"
```

と打ってください。最後に描いたグラフが再現しましたね。試しに、

```
set xrange [-25:25]; replot
```

と打ってみましょう。

```
set nogrid; replot
```

というのもやってみてください。前の作業を継続できているのがわかるでしょう？

いま `set` コマンドと `replot` コマンドを一行に書いてしまいましたが、これについての説明はまだでした。実は、セミコロン ; で区切って複数のコマンドを一行に書いてもよいのです。ご想像のように、並べられたコマンドは左から順に実行されます。`set nogrid` は `set grid` の逆の働きをするコマンドです。(やってみればわかりますよね？)

6.4 数値データからグラフを描く

測定や調査結果のデータを GNUPLOT でグラフにするには、データを一旦ファイルに収めてからそれを GNUPLOT に処理させます。このファイルをデータファイルといいます。

データファイルは数値をただ並べただけのテキストファイルです。コンピュータで計算処理した結果を GNUPLOT でグラフにすることもできます。この場合は計算処理を行うプログラムは結果を数値として打ち出してデータファイルを作ることができればよいのです。

6.4.1 二次元データのプロット

x の値と y の値の組が複数並んだデータから作ったグラフを散布図といいます。各組みはデータ点と呼びます。データファイルには、1データ点あたり一組の x と y の値をこの順に 1 行に空白で分けて並べます。次のデータは 11 個のデータ点からなっています。

```
0.0     0.0
0.9     1.8
1.8    11.9
2.7    55.3
3.6   108.7
4.5    96.3
5.4    50.0
```

```
6.3     8.2
7.2     2.6
8.1     1.1
9.0     0.5
```

これをそのまま Mule のような テキストエディタ を使って打ち込み、ファイルに保存してください。GNUPLOT とは別のウィンドウで作業すればよいでしょう。ファイル名は `test1.dat` と付けてください。データを打ち込むときに、体裁を整えるために数値と数値の間の空白は適当に入れてかまいませんが、空行を入れるとグラフにしたときにその部分を線でつなぐことができなくなりますので注意してください。

ファイルに保存したら、GNUPLOT で

```
plot "test1.dat"
```

と打ってください。マークだけのグラフが描けましたね。次に、

```
set data style linespoints
```

と打ってから `replot` で描きなおすと、マークの間に線が入ったでしょう。`set data style` の後に続けるパラメータにはほかにも、`lines` や `points` や `boxes` などがあります。これらをデータ・スタイルといいます。

`plot "test1.dat" with lines` というように、`plot` コマンドの中にデータ・スタイルを含めることもできます。`with` が必要です。この場合、指定したデータ・スタイルは以後の `plot` コマンドに影響を与えるわけではなく、その場限りの効果しか持ちません。

6.4.2 エラーバー付きの二次元プロット

次のデータは測定誤差を考慮し、x、yの測定値、y の下限、y の上限を入れたものと思ってください。各データ点について 4 個の数値が入っていますが、一列目と二列目は前と同じです。

```
0.0     0.0     0.0     3.0
0.9     1.8     0.0     5.8
1.8    11.9     6.9    16.9
2.7    55.3    48.3    62.3
3.6   108.7   100.7   116.7
4.5    96.3    88.3   104.3
5.4    50.0    44.0    56.0
6.3     8.2     3.2    13.2
7.2     2.6     0.0     6.6
8.1     1.1     0.0     4.1
9.0     0.5     0.0     3.5
```

`test1.dat` を Mule で開いて書き直した後、同じ名前で保存してください。

GNUPLOT で `plot "test1.dat"` とすれば前と同じグラフになり、データの 3 列目と 4 列目は使われません。3、4 列目も使うために

```
plot "test1.dat" with errorbars
```

と打ってください。エラーバー（誤差の範囲を示す縦棒）がグラフに描き込まれたでしょう。 `errorbars` もデータ・スタイルの仲間です。

6.4.3 列の選択など

少しトリックも紹介しておきましょう。

```
set data style points
plot "test1.dat" using 1:4
```

と打ってみてください。using の後の 1:4 はデータファイルの中の 1 列目を x、4 列目を y として取り出すように指示しています。描かれたグラフを見ただけでそのことがわかるでしょう？ わからなければ、... 1:3 や ... 1:2 もやってみてください。

次に、

```
plot "test1.dat", "test1.dat" using 1:3, "test1.dat" using 1:4
```

と打ってみてください。plot コマンドではコンマ , で区切って複数のグラフを描くことができるのです。ただし、x 軸と y 軸は共通です。つぎのように数式のグラフをいっしょに入れることもできます。

```
plot "test1.dat" with errorbars, 112 * exp(-( (x - 4)/1.5)**2)
```

数式中の括弧は全部小括弧を使います[5]。

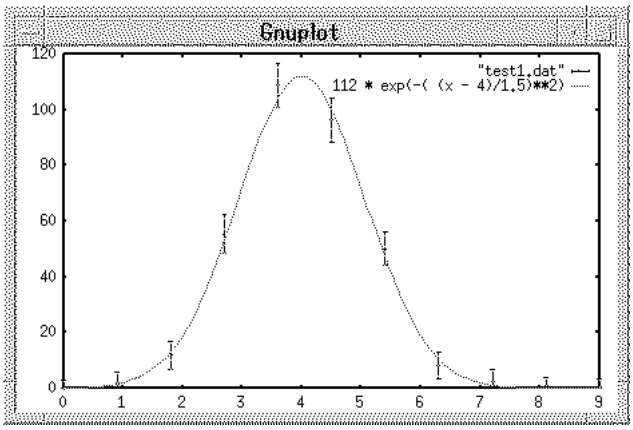


図 6.2 データと数式のグラフを一緒に描く

複数のグラフをいっしょに描く場合は x の範囲に注意してください。x 軸の範囲が思ったようにならないときは、set xrange コマンドを使ってみればよいでしょう。

軸の目盛りを「対数目盛り」にすることもできます。

```
set logscale y
set yrange [0.1:200]
replot
```

とやってみてください。縦軸の目盛りが変わっていますね。対数目盛りでは、0 より大きい範囲を指定しなければならないことに注意してください。対数目盛では 100 と 10、10 と 1、1 と 0.1 は全部間隔が同じですから、縦軸の 0 の位置は無限に下方にあるのです。

では、

[5] exp は指数関数というもので、** はべき乗の演算子。

```
set nologscale
set autoscale y
```

と打って元に戻しておいてください。`set autoscale y` は、`set yrange` コマンドの効果を取り消します。

6.4.4 フィッティング

cc2000 の GNUPLOT には GNUFIT が組み込まれていて、フィッティングを行うことができます。GNUFIT は単純な計算法を使っているので、あらゆる場合に正しい答えが得られると期待してはいけません[6]。 また、どのような数式でも正常に処理できるというわけには行かず、結果を見てうまくいったかどうかをチェックする必要があります。数式が未知パラメータに関して線形の場合は大抵うまくいくようですが、非線形の場合は駄目なことが多いようです。

ではちょっと試してみましょう。次のデータは、直線の方程式

$$y = ax + b$$

に従うはずの 2 つの量、x と y を測定して得たものと思ってください。a と b はある定数です。

```
1.6             4.8
1.0             3.8
1.8             4.8
2.9             6.7
0.2             2.8
0.5             3.3
0.5             3.1
2.5             6.3
3.3             7.6
3.4             7.6
2.2             5.7
2.4             6.2
```

これを `test2.dat` という名前のファイルに入れて、`plot "test2.dat"` でグラフを描いてみてください。データ点はだいたい一直線上に並んでいますね。この測定データから、秘密のパラメータ a と b を推定します。

まず、元の式を一発で呼び出せるようにするために、

```
f(x) = a * x + b
```

と打ちます。これで `f` という名前の関数を新たに定義して登録したことになります。このようなものを「ユーザー定義関数」といいます。`show functions` と打つと、ユーザー定義関数のリストを見ることができます。

次に、グラフを「目で見て」a と b のごく大雑把な推定値を決め、a と b に代入しておきます。a は 1 程度、b は 2 程度といったところでしょうから、

```
a = 1
b = 2
```

[6] GNUFIT は 最小二乗フィッティングを行う。これは異常データが部分的に含まれている場合にその影響を受けやすい。

と打ちます。そして、式とデータをグラフで比較するために

```
plot f(x), "test2.dat"
```

とやってみてください。だいぶん違いますねえ。
後はコンピュータで合わせてやりましょう。次のように打ってください。

```
fit f(x) "test2.dat" via a,b
```

GNUFIT は、`f(x)` とデータとの違いが最も小さくなるようなパラメータを探して計算を繰り返します。その途中経過が表示された後に最終の結果が出てきます。こんな具合です。

```
Final set of parameters            68.3% confidence interval
======================             ==========================

a               = 1.53444          -  0.0382937
b               = 2.37349          -  0.0820193


correlation matrix of the fit parameters:

                a      b
a                1.000 -0.868
b               -0.868  1.000
```

$a = 1.53444, b = 2.37349$ という結果が出ました。confidence interval というのは「信頼区間」といって、推定値がどれぐらいの不確定さを持っているかといういうことをあらわします。すべてのデータ点を一本の直線に乗せることはできないので、推定にはある程度の不確実性が伴うということはおわかりですね。correlation matrix については統計の理論を勉強してもらう必要がありますが、普通は無視してもよいでしょう[7]。

この段階で a と b の値は推定値に変わっています。`show veriables` と打てば、a や b のような「変数」とその値のリストを見ることができるでしょう。あなたの知らない変数も出てくるかも知れませんが、それは無視してください。最後に、推定値に基づいた数式とデータをグラフにしてながめてください。前に一度やっていますから、`replot` と打つだけでＯＫです。

有用な結果が得られたら、`save` コマンドで GNUPLOT の状態を適当な名前のファイルに保存して置くのがよいでしょう。後でそうやって保存したものの中から使えるものを選んで、印刷などの処理に回すのです。

6.4.5 自作プログラムで GNUPLOT を利用する

自分で作るプログラムでグラフを出力したい場合に、描画処理を GNUPLOT に任せるようにすれば手間が省けます。グラフィックスの表示や印刷の処理は使用する端末やプリンタによって違ってきますが、GNUPLOT はその違いも吸収してくれますからあなたのプログラムは GNUPLOT が出力することのできるたいへん多くの装置に出力できることになります。

[7] 推定計算の途中結果は `fit.log` という名前のテキストファイルとして残されます。

実際に試してみましょう。ごく単純な例を使いますので、UNIX やコンパイラの知識は必要ではありません。

漸化式

$$\begin{aligned} x_{i+1} &= -0.97 * x_i + y_i - 5 + 5/(1 + {x_i}^2) \\ y_{i+1} &= -0.995 * x_i \end{aligned}$$

と初期条件 $x_1 = 1,\ y_1 = 0$ で決まる x と y の値の組みの系列をグラフにしてみましょう。始めに $x = 1$、$y = 0$ という状態から出発してこの式で次の値を計算します。同じ計算を次々に繰り返して得られる x と y の列をグラフにするのです。10000 回ほど繰り返します。普通は FORTRAN などのプログラミング言語を使って計算するのでしょうが、ここでは、AWK というのを使ってみましょう。これは C 言語に似ています。

```
BEGIN{ x = 1; y = 0; i = 10000;
       while( i-- > 0 )
       {   print x, y;
           nextx = -0.97 * x + y - 5 + 5 / (1 + x * x);
           y = -0.995 * x;
           x = nextx;
       }
     }
```

テキストエディタでこれをそのまま打ち込んで `test3.awk` という名前のファイルに保存してください。括弧の種類と、コンマやセミコロンといった区切り記号に注意してください。短いので簡単ですね？

`test3.awk` ができたら、UNIX のプロンプトに対して

```
gawk -f test3.awk >test3.dat
```

と打ってください。これで AWK プログラムが走り `test3.dat` というファイルに計算結果が保存されます [8]。便りのないのはなんとやらで、プログラムに間違いがなければ何も表示されずに作業が終了します。念のために `test3.dat` の内容を見てみてください。

```
head test3.dat
```

で始めの数行が表示されます。

もし FORTRAN か Pascal を使うなら、図 6.3 や 図 6.4 のようなプログラムを書けばよいでしょう。`f77 test3.for` などとしてコンパイルした後、`a.out >test3.dat` で実行します [9]。

では、GNUPLOT でグラフにしてみましょう。GNUPLOT のプロンプトに対して

```
set data style dots
plot "test3.dat"
```

と打ってください。`dots` というデータスタイルは、データ点を小さい点でプロットするものですが、小さ過ぎたら `points` を使ってください。最後に前に説明したようにして印刷しましょう。何か芸術的なタイトルを入れるのもよいでしょう。

[8] `>test3.dat` は、プログラムの「標準出力」を端末ではなくファイル `test3.dat` に書き込むように指示している。

[9] cc2000 では、FORTRAN は `f77` コマンド、Pascal は `pc` コマンドでコンパイルできる。特に指定をしない場合、実行ファイルは `a.out` という名前になる。

```
      REAL x, y, nextx
      INTEGER i
      x = 1
      y = 0
      DO i = 1, 10000
        WRITE(*,*) x, y
        nextx = -0.97 * x + y - 5 + 5 / (1 + x * x)
        y = -0.995 * x
        x = nextx
      END DO
      STOP
      END
```

図 6.3 FORTRAN プログラムの例

```
program test3(input, output);
  var
    x, y, nextx : real;
    i : integer;
  begin
    x := 1; y := 0;
    for i := 1 to 10000 do
      begin
        writeln(x:8:2, y:8:2);
        nextx := -0.97 * x + y - 5 + 5 / (1 + x * x);
        y := -0.995 * x;
        x := nextx;
      end
  end.
```

図 6.4 Pascal プログラムの例

6.5 マニュアル類について

ここでは、cc 環境から直接得ることのできる GNUPLOT のマニュアルなどの情報について説明します。

6.5.1 マニュアルを印刷する

印刷したマニュアルがほしい場合は、cc 環境にある次のファイルを自分のところにコピーしておいて後で印刷してください。やりかたについては UNIX ガイドの LaTeX の項を参照するか、親切な友達にやってもらってください。

ディレクトリ	ファイル名
/NF/local/general/lib/gnuplot/	gnuplot-3.5.tex
	gnuplot-3.2j.tex
	gpcard.tex
	tutorial.dvi

`gnuplot-3.2j.tex` はオンラインマニュアルの日本語訳です。cc2000 では jlatex コマンドを使ってコンパイルします。コンパイルには、同じディレクトリにある toc_entr.sty と titlepage.tex が必要です。`gnuplot-3.5.tex` は英語版です。これらの内容はオンラインマニュアルと同じですが、全体を通して読むことができる点が違います。

gpcard.tex はリファレンスカード（英語）で、要領良くまとめてあるので役に立ちます。これは plainTeX 用で、cc2000 では tex コマンドでコンパイルできます。このファイルの始めのほうにある `\columnsperpage=2` という行の数字を 2 ではなく 1 に直してから TeX にかけて印刷してください。2 のままだと、用紙からはみ出すおそれがあります。後で貼り合わせて縮小コピーすれば、りっぱなリファレンスカードになるでしょう。

GNUPLOT には GnuTeX というプログラムが組み込まれていて、その機能を使って LaTeX の文書の中に GNUPLOT で描いたグラフを入れることができます。 GnuTeX の特徴は、グラフ中のタイトルや目盛りラベルなどの文字に本文中と同じく TeX のフォントが使われることで、LaTeX の記法を使って数式やギリシャ文字、記号などを入れることができます。 `tutorial.dvi` は、GnuTeX の使い方を実例をたくさん使って解説しています。これはコンパイル済みの TeX ファイルですから、そのまま `jdvi2ps` と `lpr` コマンドを使って印刷できます[10]。

なお、GNUFIT の使い方を簡単に説明したファイルも 同じディレクトリに入っています。名前は、`gnufit.doc` です。これは、Mule などで表示して読むことができますし、直接 `lpr` コマンドでプリンタに送って印刷することができます。

6.5.2 オンラインヘルプを日本語にする

オンラインヘルプが当面英語でもかまわない方や、UNIX のコマンドに慣れていない方はここは読み飛ばしてください。必要になってから読めばよいでしょう。

先程のマニュアル類の入っているディレクトリに `gnuplot.gih` というファイルと `gnuplot.gih.j` というファイルが入っています。前者は英語のオンラインマニュアルの「種になっている」ファイルです。しかし、次のようにして後者を「種」にすればオンラインマニュアルは日本語になります。

[10] TeX の DVI ファイル（ `.dvi` が付くファイル）は、`xdvi` コマンドで X 端末に表示できる。大きな画面なら十分きれいに表示でき操作も簡単なので、印刷したマニュアルのかわりとして使ってもよい。

空いているウィンドウで作業しましょう。まず cd と打って自分のホームディレクトリに移ってください。ここにある .cshrc というファイル（隠しファイルなので通常は気が付かない）を編集しますが、うまくいかなかったらすぐ元に戻せるように複製を作ります。

```
cp .cshrc cshrc.gplot
```

これで、cshrc.gplot というファイルに .cshrc の複製ができます。オンラインヘルプの日本語化がもしうまくいかなければ、後で

```
cp cshrc.gplot .cshrc
```

と打てば元に戻せます。

複製ができたら Mule などのテキストエディタで .cshrc を開いてください。たとえば、

```
mule .cshrc
```

でＯＫです。.cshrc のいちばん下の行の次に一行追加して

```
setenv GNUHELP '/NF/local/Solaris2J/lib/gnuplot/gnuplot.gih.j'
```

と書いてください。クォーテーションマークは両方ともシングルクォート ' です。行の最後には必ず改行を入れてください。書けたら保存してエディタを終了してください。念のために、cat .cshrc と打って .cshrc を表示し、最後の行をよく見てください。

うまくいったかどうかを確かめるには、いったん cc2000 からログアウトした後、再び cc2000 にログインして GNUPLOT を起動し、help と打ってください。

6.5.3 デモンストレーション・ファイル

GNUPLOT は今まで説明したよりも、もっともっとたくさんの機能を持っています。あなたが必要とする機能を探すために、デモンストレーション・ファイル を利用してください。デモンストレーション・ファイル は、ディレクトリ /NF/local/general/lib/gnuplot/demo に入っています。自分のところにコピーして使ってください。

たくさんのファイルの内どれが必要かはわかりにくいので、とりあえず全部のファイルをコピーしましょう。デモンストレーション・ファイルを収めるサブディレクトリを作って、そこにコピーするのがよいでしょう。たとえば、シェルプロンプトに対して、mkdir gplotdemo と打って、サブディレクトリ gplotdemo を作ってください。それから cd gplotdemo と打ってそこをカレントディレクトリにします。そうしてから、注意して、

```
cp /NF/local/general/lib/gnuplot/demo/* .
```

と打てば、すべてのファイルが手に入ります。最後のピリオド . はカレントディレクトリをあらわす記号で、必ず必要です。

ls コマンドでファイルのリストを見てください。ファイル名の最後が .dem となっているのが、GNUPLOT の マクロファイルで、GNUPLOT のコマンドが説明と共に入っています。Mule などでのぞいてみれば様子がわかるでしょう。説明文には各行の左端に # が付いています。# はその行の残りの部分を全部無視するように指示する GNUPLOT の命令です。

これらのマクロファイルは GNUPLOT の load コマンドを使って自動実行できるようになっています。一つのマクロファイルでたくさんのグラフを次々に表示しますが、次のグラフに切り替えるには リターン・キー を押します。マクロファイルの中に pause -1 というコマンドがところどころ入っていて、これで自動実行を一旦止めてキー入力を待つようにしているのです。

これらのマクロファイルを実行するには、カレントディレクトリがデモンストレーション・ファイルのサブディレクトリになっていなければなりません。GNUPLOT を起動する前に `cd` コマンドでこのディレクトリに移っておけばよいでしょう。GNUPLOT のコマンドにも `cd` がありますが、ディレクトリ名を ダブルクォートで囲って指定しなければなりません。 GNUPLOT の `pwd` コマンドはシェルのコマンドの `pwd` と同じで、カレントディレクトリを表示します。

図 6.5は、デモンストレーション・ファイル `world.dem` を元にして作った立体画像です。地図データから GNUPLOT で生成した図を tgif で合成した後、日本語 LaTeX で書いたこの文章に取り込みました。目を回さないようにお楽しみください。

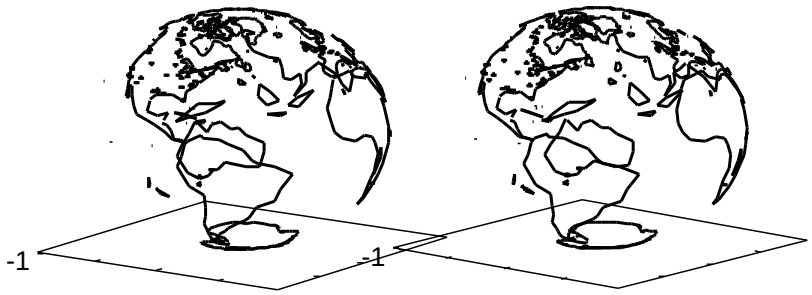
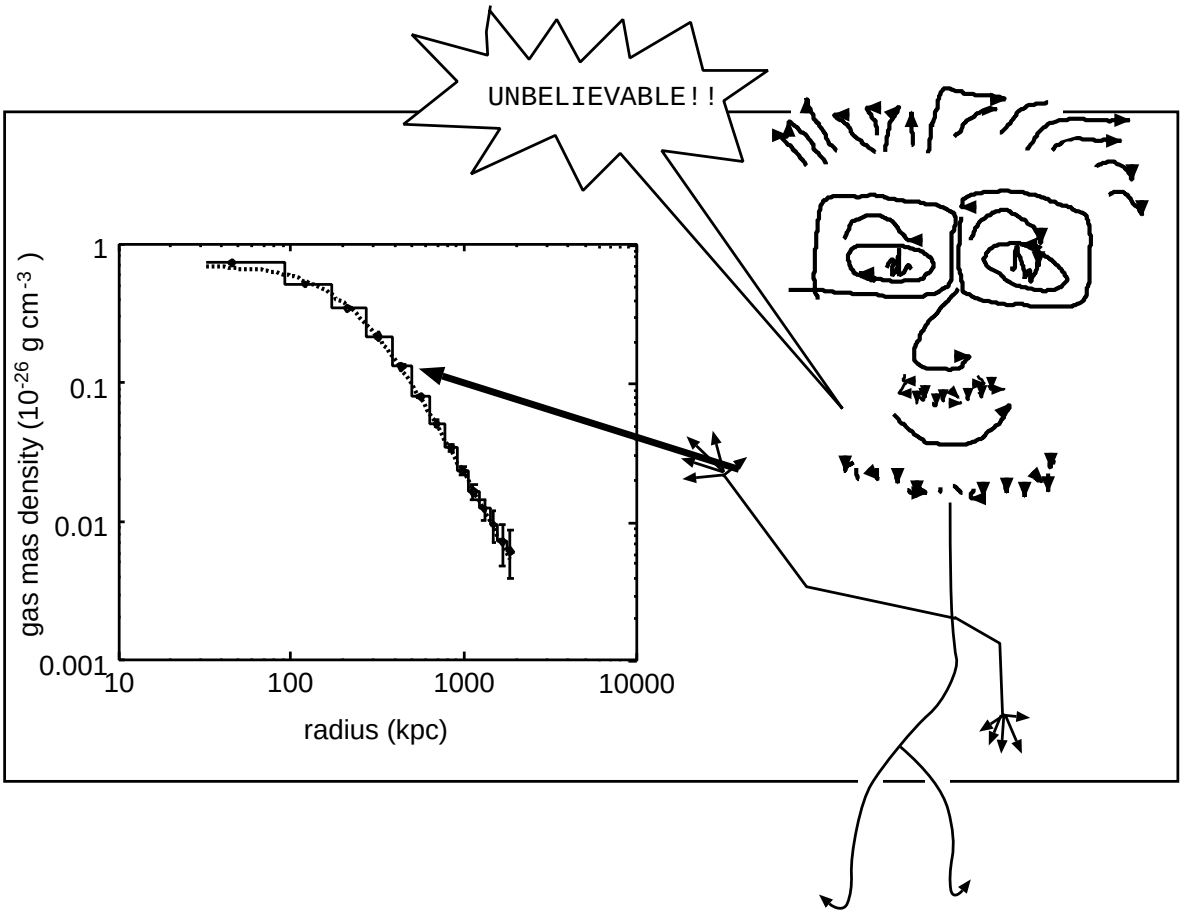


図 6.5 立体画像

第 7 章

tgif



tgif はX上で作動する対話的な作図ツールです.

7.1 tgifのできる事できない事

できる事

- テキスト，四角，楕円，ポリライン，フリーライン，ポリゴン，円弧，角丸四角形という多彩な図形を描画できます．7.4節で説明します．
- XBM，XPM，EPS形式の図形をファイルから読み込むことができます．File Menu(図7.4) のコマンドを使います．
- 日本語入力が可能です．X windowのdrag and pasteを使って日本語を入力します．Muleで日本語の文字を編集したものを利用する方法を薦めます．その方法は7.5節で説明します．
- 作成した図形はtgifの専用形式以外にもXBM，PS，EPS，EPSI，TEXT形式で出力することができます．これらの出力方法は7.4節の最後に説明します．
- カラーをサポートしています．
- 階層的な図形単位管理ができます (が，私は使ったことがありませんので今回は図7.12で機能の一部を紹介するだけで説明はしません)．

できない事

- 90度以外の図形の回転は出来ません．
- ファイルから読み込んだEPS形式の図形の画面上での表示は出来ません．

7.2 tgifの起動とWindowの名称

tgifを起動するにはターミナルから

`tgif [オプション] [ファイル名 (拡張子は obj で省略可)]`

と入力します．基本的にはオプションやファイル名を付ける必要はないでしょう．tgifで作図された図形は**拡張子が“obj”**のテキストファイルとして保存されます．

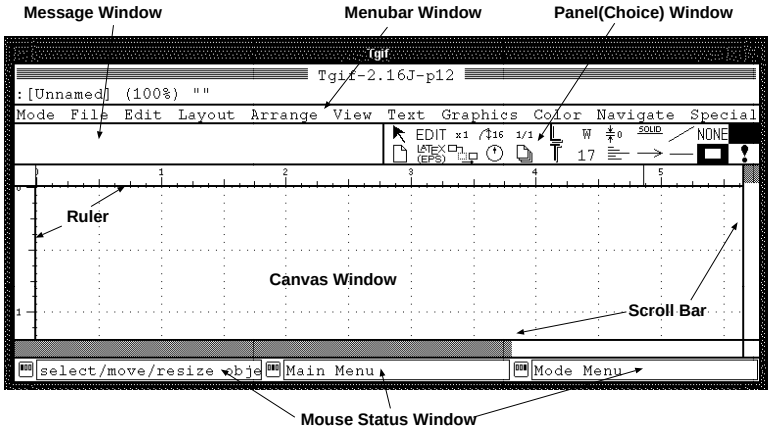
tgifが起動すると図7.1の様なウインドウが現れます． 以下では各Windowを図中の名称で呼ぶことにします．

7.3 Popup menu

Menubar Window(図7.1参照) の項目一覧 (**Main Menu**) の所でマウスの任意のボタンを押すと，Popup Menu(図7.2参照) が表示されます．その状態でマウスを上下にドラッグして，Popup Menuに表示されている項目一覧 (**Sub Menu**) の中の目的のコマンドの位置でボタンを離すと，その**コマンドを選択することができます**．この時，ボタンを離す前に左右に素早くドラッグすると，表示されているPopup Menuが常時表示されるようになります (**Popup Menuの張り付け**)．張り付けたPopup Menuの中のコマンドの選択にはマウスの中ボタンを使います．張り付けたPouup Menuの場所を移動させるにはマウスの左ボタンでドラッグします．張り付けたPouup Menuが不要になればマウスの右ボタンでクリックして消します．

Canvas Window(図7.1参照) でマウスの中ボタンを押すと，Main MenuがPopup Menuとして現れます．そのまま上下にドラッグして項目を選ぶ (つまりボタンを離す) とマウスポインタが ⌜ に変わります．

図 7.1 tgif の各 Window の名称



この状態で，もう一度 Canvas Window 内でマウスのボタンを押し続ける事で，その項目についての Sub Menu の Popuo Window が現れます．そのまま上下にドラッグしてコマンドを選択します．

Panel Window(図 7.1参照) 内の各 Panel をマウスの左右いづれかのボタンでクリックすることで，各項目の設定を変更することが出来ます．Panel 上でマウスの中ボタンを押すと，各 Sub Menu の Popup Menu が現れます．

様々な状態でマウスの各ボタンには異なった操作の意味が割り当てられています．各状態で各ボタンにどんな操作が当てられているのかを覚えておいて，それが今どうなっているのかをいちいち判断するのは，最初のうちはややこしいものです．そのために **Mouse Status Window(図 7.1参照) に，その状態でマウスの各ボタンに割り振られている機能が表示されています**．

7.4 作図の手順

作図は次のよう進めます．

まず，編集環境を適当に変更します．編集環境というのは，例えば，グリッド間隔の変更，ズーム，図形をグリッド単位で移動させる/させないようにする変更，などです．これらの環境設定を変更するコマンドは Main Menu の Layout Menu (図 7.6) の中にあります．とりあえず最初はデフォルトの環境で作業していってもよいでしょう．

次に図形単位を作ります．Mode menu(図 7.3) で，文字を入力するモード，あるいは，図形を描くモード (長方形，円，ポリライン，ポリゴン，円弧，角丸長方形，フリーライン) を選択します．**作図は Canvas Window 上で主にマウスの左ボタンを用いて行ないます**．各図形の具体的な作図方法について説明します．

- 文字入力モードを選択したら，Canvas Window 上をマウスの左ボタンでクリックします．すると文字入力用の窓が開くので文字をタイプしていきます．そのようにして作った文字列を後から適切な場所に移動させます．

- 長方形や円や角丸長方形のモードを選択したら，Canvas Window 上でマウスの左ボタンを押しながらドラッグして，適当な所で離します．縦横の大きさは後から変更できます．

- ポリラインやポリゴンのモードを選択したら，マウスのポインタの場所を変えてはマウスの左ボタンをクリックする，という作業を繰り返します．最後にマウスの右ボタンをクリックします．その点が

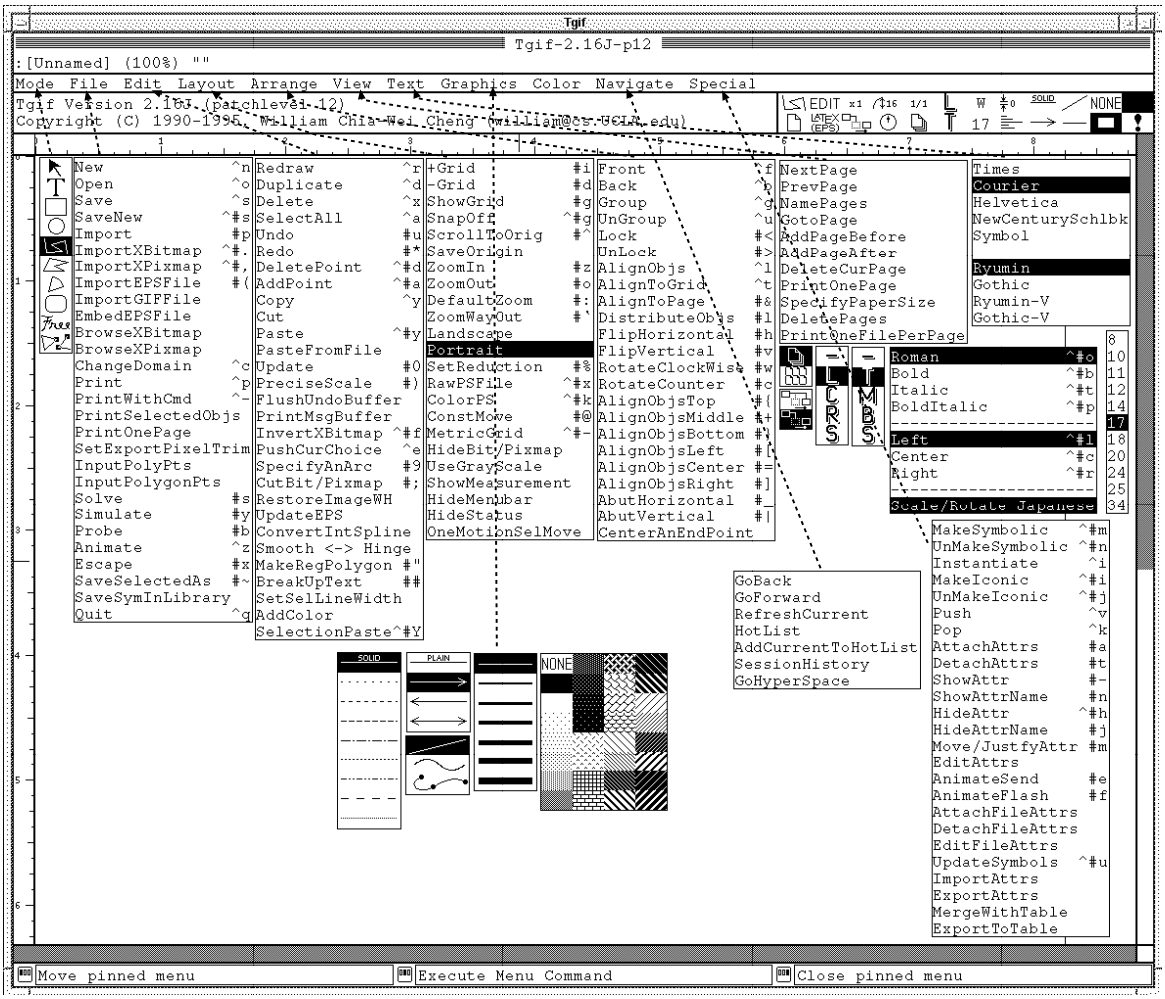
ポリラインの場合は終点に，ポリゴンの場合は最後の頂点になります．頂点や折点の位置は後から変更できます．

- 円弧のモードを選択したら，始めに Canvas Window 上をマウスの左ボタンでクリックして，円の中心の位置を決めます．次に円弧の始点位置にしたいところでマウスの左ボタンをクリック．続けて，円弧の終点の位置でマウスの左ボタンをクリックします．
- フリーラインのモードを選択したら，Canvas Window 上でマウスの左ボタンを押しながらドラッグします．これぱかりは後で移動させたり大きさを変えたり出来ないので，描きたい場所に描きたい大きさで描いてゆきます．

と，一応説明してみましたが，これらについては「百聞は一見にしかず」で，マウスを動かしながら左ボタンを押したりドラッグしたりして色々試してみた方が早いでしょう．

上記のようにして図形単位を幾つか作図したら，**Panel Window や Popup Window のコマンドを使って，選択されている図形に対して様々な加工を施してゆきます**．加工というのは，具体的には，移動，サイズの変更，塗りつぶし，整列，フォントの変更，グループ化，回転，線の太さや種類の変更などです．

図 7.2 Sub Menu の Popup menu 一覧



図形を加工するには，まずは図形を選択するモードに設定します．1 つの図形単位を選択するにはその図形単位をマウスの左ボタンでクリックします．すると図形単位の接点に四角い点が重ねて表示されます．これを以後**操作点**と呼ぶことにします．**複数の図形単位を選択するには，**`<Shift>`キーを押えながら，図形単位をマウスの左ボタンで次々とクリックしていきます．すでに選択されている図形単位に対して，もう一度`<Shift>`キーを押えながら，マウスの左ボタンでクリックすると，選択されていない状態に戻すことができます．まとめて幾つかの図形単位を選択する他の方法もあります．図形単位の無い部分からマウスの左ボタンによるドラッグを行なうと，その始点と終点を対角線とする長方形の中に含まれる全ての図形単位が選択されます．

以上のようにして選択した図形単位に対して, 操作点を左ボタンでドラッグすると，図形単位のサイズや形を変えることが出来, 操作点以外の部分を左ボタンでドラッグすると，その図形を移動させることが出来ます．

他にも, 図形を選択しておいて, Main Menu の Graphics Menu(図 7.10) のコマンドで線の太さや種類を変更したり, 線や円で囲まれた領域や線そのものに模様をつける事が出来ます．Color Menu で色をつける事が出来ます．Text Menu (図 7.9) のコマンドで文字列のフォントを変更できます．Arrange Menu (図 7.7) のコマンドで図形単位の整列，グループ化，回転，反転, などを施す事ができます．

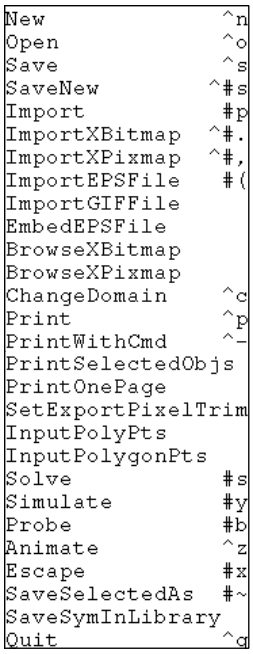
一通りの作画と加工が終ったら，File-Save コマンド (図 7.4) で tgif 専用の形式で保存します．tgif は作図した図形を様々な形式で出力できます．プリンターへ直接出力する以外に，XBM，XPM，PS，EPS，EPSI，TEXT 形式のファイルに出力できます．上記の出力形式を切替えるには，**出力形式の変更は Panel Window(図 7.1) の下の段の左から 2 番目の Panel の所 (最初は LaTeX(EPS) となっていると思います) で，マウスの左右のボタンをクリックする**か，Arrange menu(図 7.7) の下から 9 番目の項目を選択します．この様にして出力形式を決めたら，File menu の Print コマンド (図 7.4) を選択します．これで図形が目的の形式で出力されます．例えば TeX の文書の中で使う場合は EPS を，fvwm のアイコンに使う場合は XPM を選択しておいて Print コマンドを実行します．

図 7.3以降で, 私が把握している Sub Menu のコマンド一覧を示しておきます．

図 7.3 Mode:描画モード

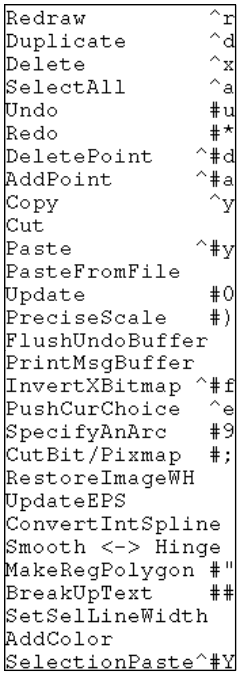
図形を選択するモード
文字を入力するモード
長方形を描くモード
円を描くモード
ポリライン (多節直線やスプライン曲線) を描くモード
ポリゴン (多角形) を描くモード
円弧を描くモード
角の丸い長方形を描くモード
自由曲線を描くモード
ポリラインやポリゴンの操作点を移動させるモード

図 7.4 File:図形ファイルの入出力



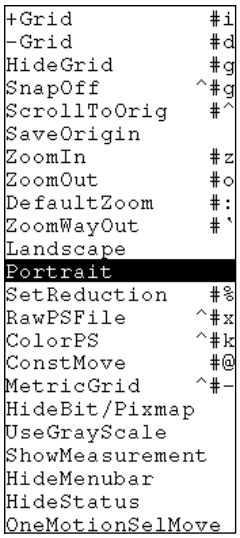
New	^n	新しくobjファイルをオープン
Open	^o	既存のobjファイルをオープン
Save	^s	現在の図形をobjファイルに保存
SaveNew	^#s	現在の図形を新しい名前でobjファイルに保存
Import	#p	objファイルを挿入
ImportXBitmap	^#.	xbmファイルを挿入
ImportXPixmap	^#,	xpmファイルを挿入
ImportEPSFile	#(	epsファイルを挿入
ImportGIFFile		
EmbedEPSFile		
BrowseXBitmap		カレントディレクトリの全xbmファイルの挿入
BrowseXPixmap		カレントディレクトリの全xpmファイルの挿入
ChangeDomain	^c	
Print	^p	指定されているデバイスへ出力
PrintWithCmd	^-	commandを指定してPrintに
PrintSelectedObjs		選択されている図形をPrintに
PrintOnePage		
SetExportPixelTrim		
InputPolyPts		
InputPolygonPts		
Solve	#s	
Simulate	#y	
Probe	#b	
Animate	^z	
Escape	#x	
SaveSelectedAs	#~	
SaveSymInLibrary		
Quit	^q	終了

図 7.5 Edit:図形の編集



Redraw	^r	図形の再描画
Duplicate	^d	選択されている図形の複製
Delete	^x	選択されている図形の消去
SelectAll	^a	全ての図形を選択
Undo	#u	過去の操作の取消
Redo	#*	Undoの取消
DeletePoint	^#d	選択されているポリゴンポリラインの制御点を消去
AddPoint	^#a	選択されている図形に制御点を付加
Copy	^y	選択されている図形をカットバッファにコピー
Cut		選択されている図形をカットバッファに移動
Paste	^#y	カットバッファの図形を挿入
PasteFromFile		テキストファイルを文字列として挿入
Update	#0	
PreciseScale	#)	選択されている図形のサイズを数字入力で変更する
FlushUndoBuffer		Undoバッファを消去
PrintMsgBuffer		Message Windowの表示をファイルやターミナルに出力する
InvertXBitmap	^#f	選択されているビットマップを白黒反転
PushCurChoice	^e	
SpecifyAnArc	#9	円弧を半径と二つの角度を入力する方法で描画
CutBit/Pixmap	#;	選択されているxbm/xpmから入力された幅と高さの領域の外側を刈り取る
RestoreImageWH		
UpdateEPS		選択されているEPS図形を更新する
ConvertIntSpline		内挿スプライン曲線を通常のスプライン曲線形式に変更する
Smooth <-> Hinge		選択されている通常のスプライン曲線を操作点[に拘束する/から緩める]
MakeRegPolygon	#"	選択されているポリゴンやスプライン閉曲線を正規形にする
BreakUpText	##	選択されている文字列を文字単位にバラバラにする
SetSelLineWidth		選択されている図形単位の線の太さを数字入力で変更する
AddColor		
SelectionPaste	^#Y	X windowのcut bufferの文字列を挿入する

図 7.6 Layout:編集環境の変更



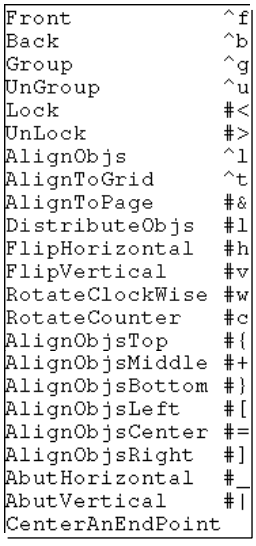
グリッド間隔を増やす
グリッド間隔を減らす
グリッドを[表示しない/表示する]
選択されている図形単位を[グリッド単位で移動/グリッドによらず移動させる]

拡大率を上げる
拡大率を下げる
拡大率をデフォルトに戻す
キャンバス全体が表示される拡大率まで下げる
キャンバスを横長にする
キャンバスを縦長にする

Print の出力先を指定[LaTeXFig(EPS)-RawPS-XBitmap-TextFile-EPSI-Printer]

グリッドの単位を[cmにする/inchにする]
XBitmapXPixmapの内容を[表示しない/表示する]
出力時にタイルパターンを[グレースケール/タイルパターン]にする
ポインターの位置座標を[表示する/表示しない]
コマンドメニューバーの欄を表示[しない/表示する]
マウスステータスの欄を[表示しない/表示する]
[クリック-選択 クリック-移動/クリック-移動]

図 7.7 Arrange:図形の整理整頓



選択されている図形を前に移す
選択されている図形を後ろに移す
選択されている複数の図形をグループ化する
選択されている図形のグループ化を解く
選択されている図形を固定する
選択されている図形の固定を解く
選択されている複数の図形の左上の位置を揃える
選択されている図形の位置をグリッドに合わせる(SnapがOnの時のみ)
選択されている図形の左上の位置をページの左上に置く

選択されている図形を水平方向に反転させる
選択されている図形を垂直方向に反転させる
選択されている図形を時計回りに90度単位で回転させる
選択されている図形を反時計回りに90度単位で回転させる
選択されている複数の図形の上の位置を揃える
選択されている複数の図形の上下の中央の位置を揃える
選択されている複数の図形の下の位置を揃える
選択されている複数の図形の左の位置を揃える
選択されている複数の図形の左右の中央の位置を揃える
選択されている複数の図形の右の位置を揃える
選択されている複数の図形の水平方向の境を合わせる
選択されている複数の図形の垂直方向の境を合わせる
選択されているポリラインの端を選択されている図形の中心に移動させる

図 7.8 View.Page.PageLayout.HoriArign.VertiArign.MoveMode:ページ管理

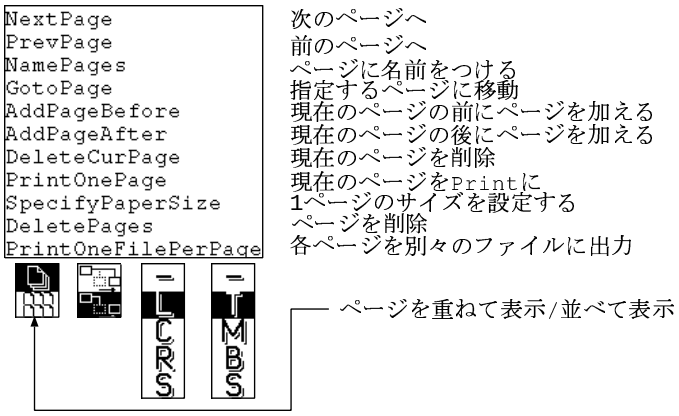


図 7.9 Text:Font.TextStyle.TextSize:文字種の変更

図 7.10 Graphics:LineDash.LineStyle.LineType.LineWidth.Fill.Pen: 線やのパターンの変更

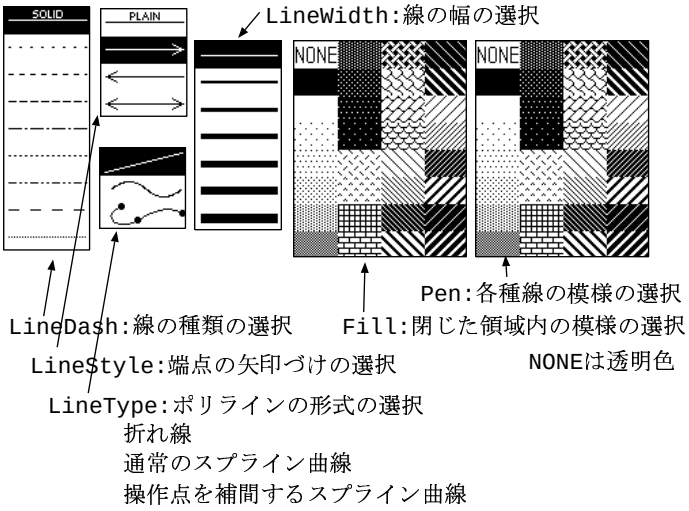


図 7.11 Navigate:

GoBack	戻る
GoForward	進む
RefreshCurrent	現在のページの再描画
HotList	HotListからアクセス先を選択する
AddCurrentToHotList	現在のページをHotListに加える
SessionHistory	ヒストリー機能
GoHyperSpace	tgif HyperSpaceモードに入る

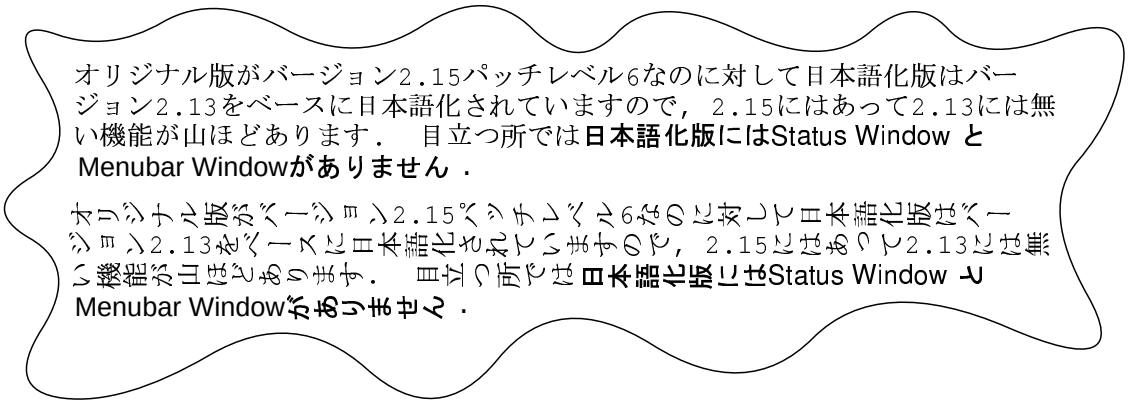
図 7.12 Special:その他の機能

MakeSymbolic	^#m	選択されているグループ化された図形をシンボル化して部品を作成
UnMakeSymbolic	^#n	シンボル化の解除
Instantiate	^i	部品 (シンボル化して保存されている図形) の挿入
MakeIconic	^#i	選択されているグループ化された図形を部品としてアイコン化
UnMakeIconic	^#j	アイコン化の解除
Push	^v	部品の下位図形の編集
Pop	^k	部品の上位図形の編集
AttachAttrs	#a	
DetachAttrs	#t	
ShowAttr	#-	
ShowAttrName	#n	
HideAttr	^#h	
HideAttrName	#j	
Move/JustfyAttr	#m	
EditAttrs		
AnimateSend	#e	
AnimateFlash	#f	
AttachFileAttrs		
DetachFileAttrs		
EditFileAttrs		
UpdateSymbols	^#u	シンボル化している図形の更新
ImportAttrs		
ExportAttrs		
MergeWithTable		
ExportToTable		

7.5 日本語の入力方法

日本語は **X window の drag and paste を使って入力します**. Mule 上で日本語のテキストを書いて必要な部分をドラッグ (マウスを左ボタンを押したまま動かす事) しておいて, **Edit menu(図 7.5) の SelectionPaste** コマンドを選択します (コマンドの選択方法については 7.3節参照). すると, 四角い枠がマウスポインタについてくるので適当な場所でマウスのボタンを押します. 後は普通の図形単位と同じように扱えます.

図 7.13 日本語出力の例



7.6 tgif 関連のツール

- **pstotgif** pstotgif は単純な線画からなる PostScript フォーマットを tgif フォーマットに変換します. 使い方はコマンドの引数に PS ファイルを付けるだけです.

```
cc2000% pstotgif foo.ps
Tgif file written to foo.ps.obj
cc2000% tgif foo.ps.obj
```

成功率はあまり高くないようですが試してみる価値はあるかも知れません. mathematica で作った PostScript file は不完全ながら変換できるようです. 私は PostScript の事は分らないので, 具体的にどういう PostScript ファイルが通ってどういうのが通らないかは分りません.

- **gnuplot** gnuplot はグラフの出力フォーマットとして tgif フォーマットをサポートしています. 以下に使用例を示します. 個々の gnuplot の説明については第 6 章を参照して下さい.

```
cc2000% gnuplot

        G N U P L O T
        unix version 3.5
        patchlevel 3.50.1.17, 27 Aug 93
```

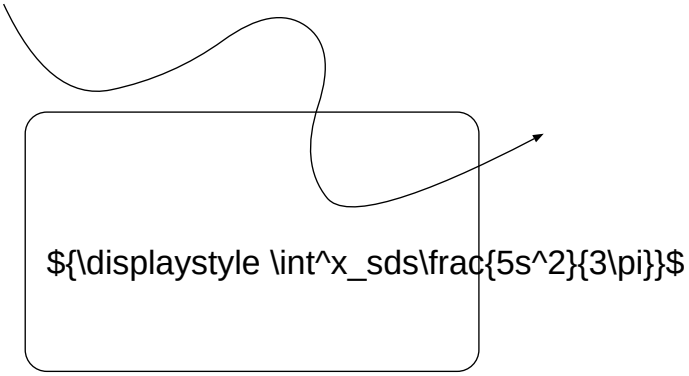
```
        last modified Fri Aug 27 05:21:33 GMT 1993

        *** GNUFIT 1.2                              22 Nov 93 ***
        *** Nonlinear least squares fit added by C. Grammes ***
        *** Send bugs regarding fit to cagr@rz.uni-sb.de    ***

        Copyright(C) 1986 - 1993   Thomas Williams, Colin Kelley

        Send bugs and comments to bug-gnuplot@dartmouth.edu

        Send bugs, suggestions and mods to bug-gnuplot@dartmouth.edu

Terminal type set to 'x11'
gnuplot> plot'foo.dat'
gnuplot> set xlabel'radius (cm)'
gnuplot> set ylabel'weight (g)'
gnuplot> replot
gnuplot> set term tgif
Terminal type set to 'tgif'
gnuplot> set output'foo.obj'  <------------ グラフの出力先のファイル名
gnuplot> replot                             を入力します．拡張子は''obj''
gnuplot> quit
cc2000% ls
... foo.obj ..
cc2000% tgif foo.obj &
...
```

- **tgif2tex**　tgif では書けないけれど LaTeX では書けるような文字列を図中に入れたい時には，tgif の文字入力モードで LaTeX のコマンドを使って文字列を入力してから，tgif2tex を使って LaTeX の出力に変換します．例えば tgif で編集中の図形 (以降 foo.obj というファイル名だとします．) の中に $\int_s^x ds \frac{5s^2}{3\pi}$ という数式を入れたいとします．その場合，tgif を使って図 7.14のように作図します．
これを EPS 形式で出力します (出力の仕方は 7.3節参照)．そうして得られた EPS ファイル (foo.eps) を tgif2tex に掛けることで foo.tps という LaTeX のソースファイルと foo.dps という EPS ファイルが作られます．以下が tgif2tex の実行の様子です．

```
cc2000% tgif2tex foo.eps
tgif2tex foo.eps
tgif2tex version 1.33p1
foo.dps and foo.tps generated.
cc2000% ls
... foo.dps foo.tps ...
```

foo.dps はもとの foo.eps の中から文字列の部分を取り去ったもので，もう一つのファイル foo.tps の中身は

図 7.14 tgif での LaTeX コマンドの使用



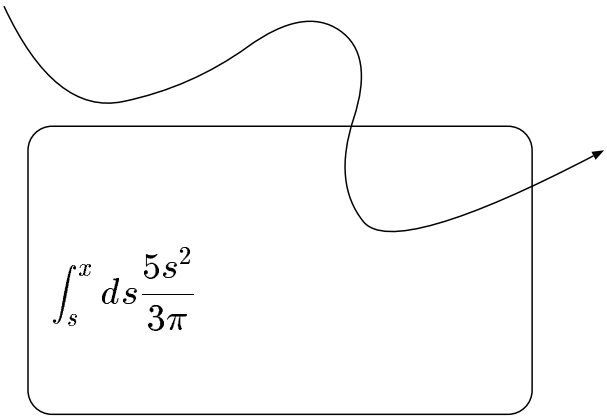

```
%% Generated by tgif2tex version 1.33p1
%% Original: TGIF_test_of_tgif2tps.eps page 1
\unitlength 1pt
 \begin{picture}(388,192)
 \put(0,0){\epsfile{file=foo.dps,scale=1}}
 \put(128.2,111.8){\makebox(0,0)[lb]{\mbox{\LARGE ${\displaystyle \
                                    \int^x_sds\frac{5s^2}{3\pi}}$}}}
 \end{picture}
```

のように抜き取った文字列と foo.dps を適切な位置に記述しているだけです．あとは次のように，この foo.tps を LaTeX のソースファイル (foo.tex) の中で呼び出します．

```
\documentstyle[epsbox]{jarticle}
\begin{document}
\input{foo.tps}
\end{document}
```

最後に foo.tex を jlatex に通せば出来上がり．

図 7.15 tgif2tex の出力例



上付や下付の添字, 矢印, ギリシャ文字などは tgif で描けます．tgif で出来る事まで TEX のコマンドにして tgif2tex を使うと無用の苦労をする事にもなります．tgif2tex の使い過ぎに注意しましょう．

7.7 終りに

私自身も完全に tgif の使い方を把握している訳ではありませんので，この章でも tgif の機能の一部しか紹介できていません．他の tgif のドキュメントとしては，tgif2.2.4 をベースに日本語化された tgif+ついて, その作者が書いたとても親切丁寧なものがあります．それを/NF/local/general/doc/tgif+.ps として置いてあります．cc で使えるもバージョンとかなり開きがあるので細かい所は色々異なりますが，使い始めの頃はとても参考になると思います．また、UNIXmagazine の 93 年 1 月号と 3 月号にやはり tgif+の使い方の記事が載っていて，こちらも参考にされると良いかも知れません．あとの細かいところはオンラインマニュアルを参考にして下さい．

第 8 章

XV

xv を使うと、画像データを表示させる事が出来ます。cc 環境では画面印刷に欠かせない道具でもあります。ある種の画像フォーマット[1]から別のフォーマットへと変換する事も出来ます。*xv* は X ウィンドウ環境で働きますので、利用するためには X 環境が必要です。具体的には cc 環境では 21 情報処理教室の csosf01 〜40 に login するか、C1 情報処理教室などの X 端末ソフトを用意しているコンピュータから cc2000 に login する必要があるでしょう。

この章では *xv* の簡単な使い方を説明します。

8.1 はじめに

画像データの作成、変換といっても具体的にその必要性が思い浮かばないかも知れません。例えば何かの画像を LaTeX の文中に張り付けたい時があったとしましょう。以下のような具合です。

図 8.1 意味もなく現れるユカリさん

このようなふざけた例ばかりでなく、GNUPLOT[2]で作成したグラフや tgif[3]や xpaint[4]で作成した絵なども画像のうちの一つです。さあ、あなたも画像データを扱いたくなってきたでしょう！

[1] フォーマットの概念についてはここでは説明しません。
[2] GNUPLOT については第 6 章を参照して下さい。
[3] tgif については第 7 章を参照して下さい。
[4] xpaint については第 9 章を参照して下さい。

8.2 まずマニアックな人のために

最初に画像関係の事情をよく判っている人のための機能説明を専門用語でささささっとやってしまいます。画像ファイルのフォーマットなどと聞いて良く判らない人は次の章まで読み飛ばして下さい。

xv は以下の画像フォーマットを扱う事が出来ます。

GIF	一般的に流通していますね。256 色までしか表現できないのが今となっては辛い。
PM	良く知りません。
PBM	Portable Bitmap. 良く知りません。
X11 Bitmap	モノクロ二値画像のみです。X 環境では `bitmap` コマンドで扱えます。
Sun Rasterfile	Sun Microsystems 社が規定したほとんどベタのフォーマットです。たしかモノクロだけが規定されていたような、、、
BMP	Microsoft Windows Bitmap. Windows パソコン標準の画像フォーマットです。
JPEG	写真などを綺麗に圧縮するにはこれが最高。互換性も高いですね。
TIFF	これまた一般的に流通しています。NeXT は拡張して色数 256 以上に増やしています。機種依存が激しく互換性に乏しいフォーマットですね。
PGM	Portable Graymap. 同じく良く知りません。
PPM	Portable Pixmap. またまた良く知りません。
RLE	良く知りません。
RGB	Red Green Bule... でしょうね、、良く知りません。
PCX	良く知りません。
IRIS	良く知りません。

プリンタなどで流通している PostScript フォーマットを、*xv* は各種画像の出力フォーマットとしてだけサポートします。PostScript 画像を表示する事は出来ません。

8.3 基本操作

まず基本操作を習得しましょう。

8.3.1 *xv* の起動と終了

xv を起動するには X 環境が利用できないといけません。cc 環境であればあなたは既に 21 情報処理教室の csosf01〜40 に login しているか、C1 情報処理教室などの X 端末ソフトを用意しているコンピュータから cc2000 に login していますね。そこでターミナル（cc 環境であれば Kterm ですね）を一つ表示させて、そこにはコマンドプロンプトが表示されている事を確認して下さい。そこから以下のように `xv &<return>` で *xv* を起動します。以下の例ではコマンドプロンプトは cc2000 マシンのものになっています。

```
cc2000(199)% xv &
[7] 23547
cc2000(200)%
```

すると以下のようなウィンドウが一つ現れると思います。

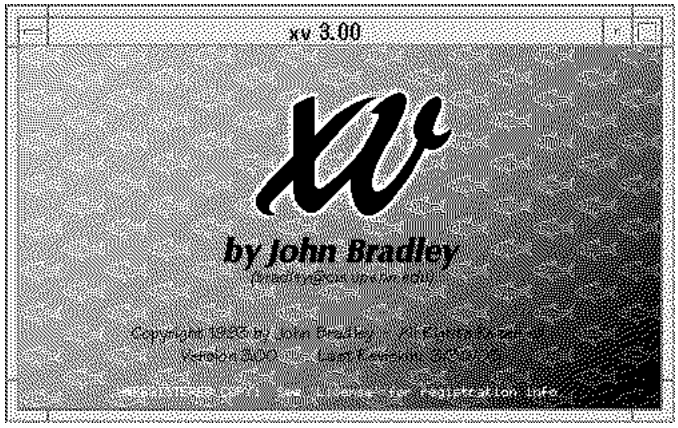


図 8.2 メインウィンドウ

このウィンドウをメインウィンドウと呼ぶことにします[5]。メインウィンドウが表示されたら *xv* の起動に成功しています。とりあえずいろいろな作業をやっていくわけですが、ここではまず一連の操作を覚えると言う意味で何も作業しないで終了してみましょう。メインウィンドウはそこに画像を表示するためのものです。すなわち起動した直後はイニシャルの画像（「*xv*」と格好良く書かれた「絵」）が表示されていると言う訳です。このメインウィンドウでは *xv* を操作することは出来ません。メインウィンドウの絵をマウスの右ボタンでクリックし、コントロール用のウィンドウを表示させましょう。

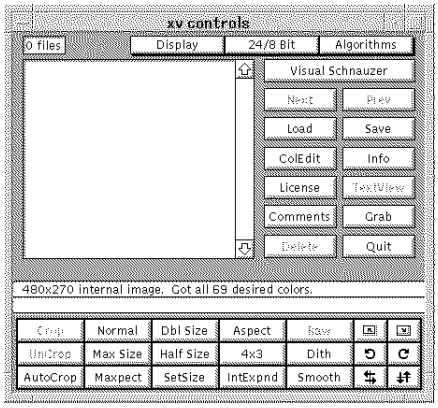


図 8.3 controls ウィンドウ

このコントロールウィンドウには数個のボタンがついていて、このボタンをマウスでクリックすることによって *xv* を操作するのです。*xv* を終了する為にはコントロールウィンドウの Quit ボタンをクリックして下さい。

8.3.2 画像を表示する

xv の終了操作には成功したと思います。今度は画像を表示する為にもう一度 *xv* を起動して下さい。コントロールウィンドウも表示させておいて下さい。但し今度は、後の操作を簡単にする為に画像データが置いてあるディレクトリに `cd` コマンドで移動してから *xv* を起動して下さい。画像データを持っていない人

[5] 正しくは何と呼ぶべきなのでしょう。私は知りません。

は/NF/local/general/lib/picture ディレクトリ以下にサンプルの画像が置いてありますのでこのディレクトリに移動してから起動して下さい。

画像ファイルを画面に表示するにはコントロールウィンドウの Load ボタンをクリックして下さい。以下のウィンドウが表示されるでしょう。

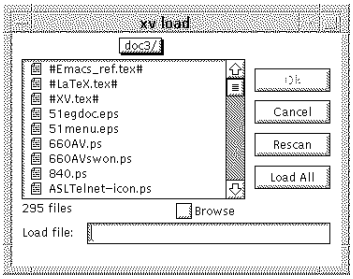


図 8.4 load ウィンドウ

このウィンドウの下側にファイル名を入力する為の枠が取ってありますね。ここにファイル名をタイプして下さい。この枠のすぐ上にはカレントディレクトリのファイルの一覧が表示されていますから、この中から表示させたいファイルをクリックすればいちいちタイプしなくても選んだものをタイプしたのと同じ状態にしてくれます。その状態で Ok ボタンをクリックするか<return>すれば、そのファイルがメインウィンドウに表示されます。

次の画像を表示させたいと思ったらまた Load ボタンを押して、新しい画像データのファイル名をタイプして読み込ませれば良いのです。*xv* は最後に読み込まれた画像だけを表示し、作業の対象とします。

ファイル名の代わりにディレクトリ名をタイプすることでディレクトリの移動が出来ます。「..」とタイプして一つ上のディレクトリ階層に移動することも出来ます。

8.3.3 画像を保存する

表示されている画像をファイルとして保存するには Save ボタンをクリックして下さい。以下のウィンドウが現れるでしょう。

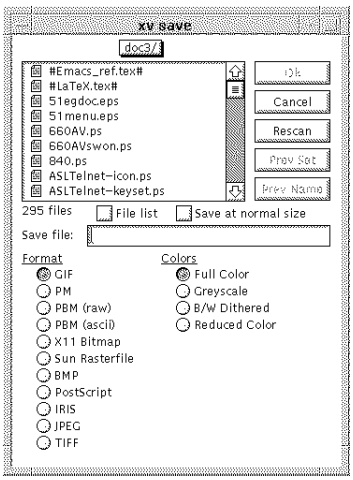


図 8.5 save ウィンドウ

ここでファイル名をタイプして Ok ボタンをクリックするか<return>です。表示されていた画像は指定された名前のファイルとして保存されます。出来上がった画像データファイルがどの程度の大きさになるのか`ls -l`コマンドで確認してみると良いでしょう。個々のファイルの大きさを確認する作業に関しては、インターネット編の **UNIX もっともっと**を参照して下さい。

画像データをフォーマット変換する

保存の時の save ウィンドウのファイル名の為の枠の下に並んでいる丸いボタンの列に注目して下さい。そこには様々な画像ファイルのフォーマットが並んでいると思います。*xv* は標準的には読み込んだ絵のフォーマットで保存しようとしますが、ここで違うフォーマットを選ぶ事も出来ます。例えば読み込んだ GIF フォーマットの画像データを保存する時に JPEG で保存するなどという状況です。これによって画像データのフォーマットを変換する事が出来るという訳です。

LaTeX などで画像を扱う為には PostScript というフォーマットでなければいけません。PostScript での保存については 8.4.3で詳しく説明していますので参照して下さい。

8.4 応用操作

基本を押えたら次は *xv* に満載された機能（の一部）を見ていくことにしましょう。説明はかなりはしょって行ないますので、自分で試しながら理解していって下さい。

8.4.1 画像データを加工する

さて、*xv* を利用すると画像をいろいろ加工できます。一番簡単な加工方法はメインウィンドウの大きさを変えて絵全体の大きさ、縦横比率を変える事でしょうか。また、コントロールウィンドウの下側に並んだボタンの数々をクリックする事でさまざまな効果が得られます。それぞれどういう機能があるのか、各ボタンをクリックして確認しておくと良いでしょう。ここではその中で比較的判りにくい Crop についてだけ説明します。

Crop とは切りとる、刈り取るというような意味で、一般に画像処理関係では要る部分だけを抜き出す作業[6]のことを指します。Crop するにはまずメインウィンドウの中から抜き出す領域を指定します。メインウィンドウの絵の抜き出そうと思う左上の地点から右下の地点までマウスの中ボタンでドラッグします[7]。これで四角い枠がメインウィンドウの画像の中に残ります。領域の指定に失敗したと思ったらもう一度やり直せば良いのです。枠が残っている間にコントロールウィンドウの Crop ボタンをクリックして下さい。メインウィンドウが枠のサイズにまで小さくなったのが判りますね。つまり画像の一部を切りとることが出来たのです。切り取りに失敗したら UnCrop ボタンを押す事で元に戻せます。

さて、様々な画像の変換作業をしたとしても、それは画面の上で行なっているだけで実際の画像データが入っていたファイルには何の影響もありません。加工した画像を画像データファイルに反映させるには現在の画像をファイルに保存してやらなければなりません。保存の仕方は既に説明しましたね。

8.4.2 画面に表示されているウィンドウを取り込む

画面に表示されているウィンドウを絵として画像データに残したいと思う時があるでしょう。例えばこのガイドのように、何かの説明の時に今画面に表示されているものを LaTeX に取り込んで例示としたいなどです。単純に画面に表示されているものをそのまま印刷したいと思う事もあるでしょう。

[6] 同じ作業をトリミングなどと表現する場合もあります。
[7] マウスの中ボタンを押して、押したまま右下に移動してから離す。

そのような場合に役に立つのがコントロールウィンドウにある Grab ボタンです。画像データにしたいウィンドウをよく見える位置に移動して、Grab ボタンをクリックします。「ぴっ」と音が鳴ったら目標のウィンドウをクリックします。その後数秒して「ぴぴっ」と鳴ったら目標のウィンドウのそっくりさんがメインウィンドウに画像として取り込まれているでしょう。まさに「ぴっ」と「ぴぴっ」の間に「ひっつかんで」来た訳ですね。あとは好きに加工、保存して利用すれば良いのです。

8.4.3 印刷する

cc 環境ではプリンタは PostScript フォーマットだけを受け付けます。画像データを印刷するにはとにかく PostScript にフォーマット変換して保存してやらなければなりません。印刷したい画像データがメインウィンドウに表示されているとします。この画像をまず保存する為に Save ボタンをクリックして、フォーマットの指定をします。私のお勧めは PostScript - B/W Dithered で保存する事です。Full Color や Grayscale などで保存しても最終的に印刷できない事はありませんが、一枚印刷するのに数分以上掛かってしまいます。それでは他のプリンタ利用者に迷惑ですからね！フォーマットの指定が済んだら Ok ボタンをクリックするか、**<return>**すると以下のウィンドウが表示されます。

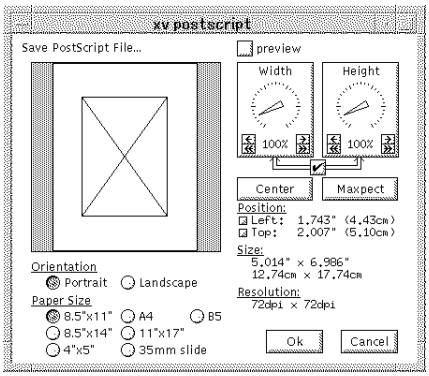


図 8.6 postscript ウィンドウ

ここで印刷する時のイメージ調整が出来ます。いろいろ試して下さい。大抵の事は気にしないで良いのですが、右半分に表示されているダイヤルで行なう倍率指定は有効です。これで印刷する時の縮尺を変えられます。ダイヤルの下にある小さな←ボタンをクリックする事で調整します。調整が決まれば Ok ボタンをクリックするか**<return>**です。

これで PostScript の画像データファイルが出来ました。プリンタで印刷するのに特殊なテクニックは必要ありません。以下のように `lpr` コマンドを普通に利用するだけです。

```
cc2000(257)% lpr -Pcspr01 window_title.ps
cc2000(258)%
```

`lpr` コマンドなど印刷については 173ページの A.1.4を参照して下さい。

8.4.4 壁紙を貼る

コントロールウィンドウの上の方にある Display ボタンをプレスして下さい。するとそこから小さなメニューが下がってくるでしょう。そのままマウスを下にドラッグして、Root: tiled と書かれているところでボタンを離して下さい。メインウィンドウがなくなり、そこにあった画像が画面の背景いっぱいに表示

されたでしょう。このような画像の事を壁紙などと呼んでいます。Root: tiled 以外を指定したらどうなるか、試してみると良いでしょう。

このまま画像を新しく Load ボタンをクリックして表示しようとすると、その画像は直接壁紙となって表示されます。また、面白い事に壁紙は *xv* を終了しても消えません。

壁紙を貼るもう一つの方法は、以下のようにして *xv* を起動する事です。

```
cc2000(274)% xv -root -quit heavymetal.jpg
cc2000(275)%
```

これでカレントディレクトリにある `heavymetal.jpg` という画像ファイルを壁紙として貼ってくれます。*xv* そのものは`-quit` オプションによってすぐ終了してしまうのですが、壁紙は残ってしまう性質を持っているから大丈夫と言う訳です。

8.4.5 色あいを変える

時々画像が全体的に暗いのでもっと全体を明るくしたいとか、ある特定の色を別の色に変えてしまいたいと思う時があります。要は色調を変えると言う事ですが、*xv* はある程度までならこれを可能にしています。コントロールウィンドウにある ColEdit（Color Edit の略でしょう）ボタンをクリックして下さい。新しいウィンドウが一つ開いて、そこで全体の色調、ある特定の色の変更、モノクロ化、色反転などが出来ます。この Color editor ウィンドウを消すにはもう一度 ColEdit ボタンをクリックすれば良いのです。

Color Edit 機能は説明し出すときりがないので止めておきます[8]。RGB, HSV, Intensity, Saturation と言った色編集についての基礎的な知識がある人には使い方は大体想像が付くと思います。適当に試しながら使って下さい。

8.4.6 Visual Schnauzer を使う

何と説明したら良いのでしょうか。余り X ウィンドウ関係の操作に慣れていない人はやらない方が良いと思いますが、慣れている人にとっては結構便利な機能だと思います。

コントロールウィンドウにある Visual Schnauzer ボタンをクリックして下さい。新しく一つのウィンドウが開き、そこにはカレントディレクトリのファイルやディレクトリの一覧がアイコン（絵文字）になって表示されます。そのファイルのアイコンをドラッグしてディレクトリ移動するとか、ダブルクリックして表示させるとか、その種の操作に慣れた人なら違和感なく操作できると思います。Visual Schnauzer ウィンドウの右上にある Misc. Commands ボタンをプレスすれば、そこからメニューが表示されますが、その中でも Update Icons は面白い機能です。実行すると以下のような一段と判りやすい表示となります。

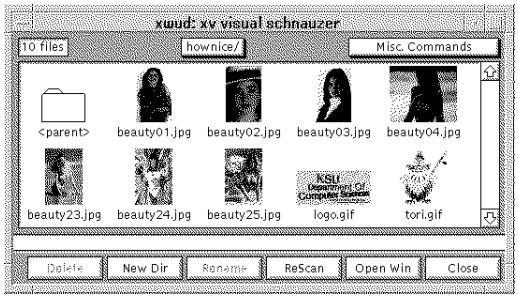


図 8.7 Visual Schnauze ウィンドウ

[8] 実は私も良く判っていません。

Visual Schnauzer ウィンドウを閉じるには、Visual Schnauzer ウィンドウの右下にある Close ボタンをクリックします。

8.5 マニュアルなど

xv はまったくもって機能満載です。それに見合う非常に詳細なマニュアルが用意されていますが、それは `man` コマンドで表示されるものではありません。`/NF/local/Solaris2J/lib/xv/xvdocs.ps` もしくは `/NF/local/OSF1/lib/xv/xvdocs.ps` というファイルに PostScript 形式で置いてあります。8.4.3で紹介したように PostScript ファイルを印刷するのに特にテクニックは必要ありません。`lpr` コマンドで直接印刷すれば良いのです。`lpr` コマンドなど印刷についてはインターネット編 **UNIX それから**の**印刷**を参照して下さい。

但しこのファイルの印刷には非常に時間が掛かります。全部で 100 ページ近くありますし、一ページそのものを印刷するのにも結構時間が掛かるものが含まれています。印刷する時はプリンタを長い間自分が占領してしまっても迷惑が掛からないかどうか確かめて行なって下さい。

8.6 法律に注意

画像を扱う時に常に注意するべき事に肖像権の問題があります。あなたが自分の作っているドキュメントに誰かのポートレート写真を使いたい時があるのは判ります。しかしその写っている相手はそうやって自分の写真があなたのドキュメントの中で使われる事を承知していますか？特にコンピュータで出版するような場合は簡単に複製、配布できたりします。作ったドキュメントがあなた以外の人の目に触れるような場合は肖像権の問題は非常に重要です。図 8.1の写真は勿論このガイドに使って無料配布すると言う事を本人に確認し、承諾して貰ってから使っています。

同様の意味であなたが扱っている画像データはその絵の作者の著作権を侵害していませんか？自分が生み出したものだけを扱っている時はこのような問題はほぼ発生しないと言えるのですが、人が作ったものを扱う時には非常に大切な事です。注意して下さい。

第 9 章

xpaint

xpaint を使うと、画像データを作成、加工、表示させる事が出来ます。xpaint は X ウィンドウ環境で働きますので、利用するためには X 環境が必要です。具体的には cc 環境では 21 情報処理教室の csosf01〜40 に login するか、C1 情報処理教室などの X 端末ソフトを用意しているコンピュータから cc2000 に login する必要があるでしょう。

この章では xpaint の簡単な使い方を説明します。

9.1 はじめに

画像データの作成、加工といっても具体的にその必要性が思い浮かばないかも知れません。例えば何かの画像を LaTeX の文中に張り付けたい時があったとしましょう。以下のような具合です。

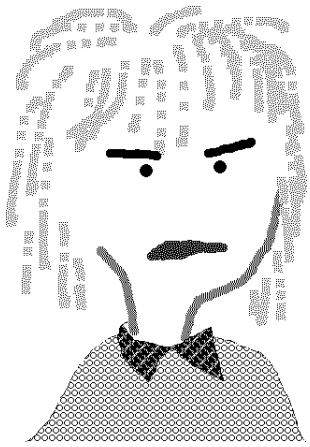


図 9.1 意味もなく現れる現在のユカリさん

このようなふざけた例ばかりでなく、tgif[1]で作成した絵なども画像のうちの一つです。さあ、あなたも画像データを扱いたくなってきたでしょう！

[1] tgif については第 7 章を参照して下さい。

9.2 まずマニアックな人のために

最初に画像関係の事情をよく判っている人のための機能説明を専門用語でささささっとやってしまいます。画像ファイルのフォーマットなどと聞いて良く判らない人は次の章まで読み飛ばして下さい。

xpaint は以下の画像フォーマットを扱う事が出来ます。

TIFF	これまた一般的に流通しています。NeXT は拡張して色数 256 以上に増やしています。機種依存が激しく互換性に乏しいフォーマットですね。
PPM	Portable Pixmap. またまた良く知りません。
GIF	一般的に流通していますね。256 色までしか表現できないのが今となっては辛い。
XBM	X Bitmap. モノクロ二値画像のみです。X 環境では `bitmap` コマンドで扱えます。
XPM	良く知りません。
XWD	X Window Dump. X 環境では `xwd, xwud` コマンドで扱えます。

プリンタなどで流通している PostScript フォーマットを、xpaint は各種画像の出力フォーマットとしてだけサポートします。PostScript 画像を表示する事は出来ません。

9.3 基本操作

まず基本操作を習得しましょう。

9.3.1 xpaint の起動と終了

xpaint を起動するには X 環境が利用できないといけません。cc 環境であればあなたは既に 21 情報処理教室の csosf01〜40 に login しているか、C1 情報処理教室などの X 端末ソフトを用意しているコンピュータから cc2000 に login していますね。そこでターミナル（cc 環境であれば Kterm ですね）を一つ表示させて、そこにはコマンドプロンプトが表示されている事を確認して下さい。

そこから以下のように `xpaint &<return>` で xpaint を起動します。以下の例ではコマンドプロンプトは cc2000 マシンのものになっています。

```
cc2000(199)% xpaint &
[7] 23547
cc2000(200)%
```

すると以下のようなウィンドウが一つ現れると思います。

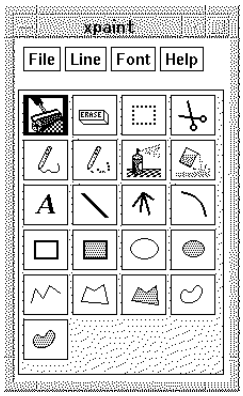


図 9.2 Toolbox ウィンドウ

このウィンドウを Toolbox(道具箱) ウィンドウと呼びます。この Toolbox ウィンドウには多くのボタンがついていて、各ボタンをマウスでクリックすることによって xpaint を操作するのです。Toolbox ウィンドウが表示されたら xpaint の起動に成功しています。とりあえずいろいろな作業をやっていくわけですが、ここではまず一連の操作を覚えると言う意味で何も作業しないで終了してみましょう。

Toolbox ウィンドウの上の方には File, Line, Font, Help のメニューボタンが用意されています。この File メニューボタンをマウスの左ボタンでプレスして下さい。以下のようなメニューが表示されます。

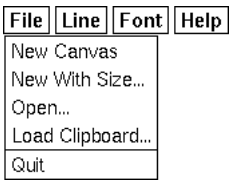


図 9.3 File メニュー

このメニューの一番下にある Quit の位置までマウスをドラッグし、Quit 項目の色が黒くなったらマウスのボタンを離して下さい。この一連の操作を「File メニューから Quit を選択する」と表現します。それで xpaint は終了します。

9.3.2 画像を新規に作成する

xpaint の終了操作には成功したと思います。今度は画像を作成する為にもう一度 xpaint を起動して下さい。

新規に画像を作成するには File メニューの New Canvas を選択して下さい。以下のウィンドウが表示されるでしょう。

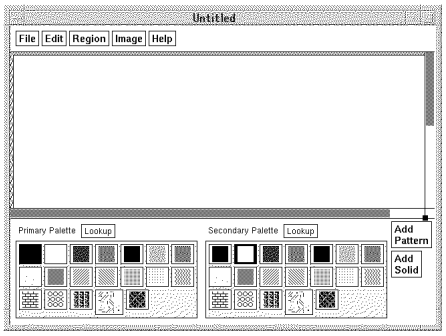


図 9.4 Painting ウィンドウ

さあ後はこのキャンバスはあなたのものです。自由に好きな絵を描きましょう。絵を描くには Toolbox ウィンドウから道具を選択し、キャンバスにマウスでクリックしたりドラッグしたりして行ないます。どの道具がどのような機能を持っているか一つ一つ試して行くのが良いでしょう。色は Painting ウィンドウの下半分に表示されているパレットをクリックして選択します。基本的には道具、色などを選択しておいてからキャンバスをクリックなどする事によってその色でその機能が働き出すようになっています。

直前の描画動作に関しては Edit メニューの Undo でやり直しが利きます。

9.3.3 画像を保存する

キャンバスの画像をファイルとして保存するには Painting ウィンドウの File メニューから Save を選択して下さい。以下のウィンドウが現れるでしょう。

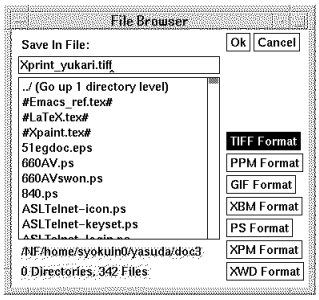


図 9.5 save ウィンドウ

ここでファイル名をタイプして Ok ボタンをクリックするか`<return>`です。表示されていた画像は指定された名前のファイルとして保存されます。出来上がった画像データファイルがどの程度の大きさになるのか `ls -l` コマンドで確認してみると良いでしょう。個々のファイルの大きさを確認する作業に関してはインターネット編 **UNIX もっともっと**を参照して下さい。

ファイル名の代わりにディレクトリ名をタイプすることでディレクトリの移動が出来ます。「`..`」とタイプして一つ上のディレクトリ階層に移動することも出来ます。

ファイル名を指定して、Ok ボタンを押す前に save ウィンドウの右に並んでいるフォーマットのボタンを押して保存する画像ファイルのフォーマットを選ぶ事も出来ます。

9.3.4 既存の画像ファイルを読み込む

既存の画像ファイルをキャンバスに読み込む事も出来ます。Toolbox ウィンドウの File メニューから Open... を選択します。図 9.5の save ウィンドウに酷似したウィンドウが現れるでしょう。保存の時と同じようにここでファイル名をタイプして Ok ボタンをクリックするか`<return>`すればそのファイルがキャンバスに表示されます。

こうして既存の画像ファイルをキャンバスに読み込んで手を加える事により、画像の加工が出来るのです。加工されたファイルを既存のファイルに上書きする形で保存するのならば、Painting ウィンドウの File メニューから Save を、元のファイルはそのままに別のファイルに保存したければ Save As... を選択して下さい。

既存の画像ファイルが xpaint で扱えないファイルフォーマットであった場合は *xv*[2] を利用して xpaint が扱えるフォーマットに変換してから処理し、再び *xv* を利用して元のファイルフォーマットに戻せば大丈夫です。

9.4 応用操作

xpaint にはなかなかどうして多くの機能があります。書くとキリがありませんし、習うより慣れろというタイプのものでもあります。自分でいろいろ試していくのが良いでしょう。ここではとりあえず直感的にイメージできない操作方法によって実行される機能に付いてだけ紹介しています。

[2] *xv* については第 8 章を参照して下さい。

9.4.1 道具の設定

線の太さ

鉛筆の太さや、四角や丸を書く時の枠線の太さを変えるにはToolboxウィンドウのLineメニューで行ないます。

とにかくダブルクリック

Toolboxウィンドウの道具ボタンをダブルクリック(二回続けてクリック)すると、幾つかの道具に関してはその道具の機能設定の為のウィンドウが表示されます。例えばローラー()をダブルクリックするとローラーの大きさを設定できるウィンドウが表示されます。

また、Paintingウィンドウの色パレットをダブルクリックすると今度は色を設定する為のウィンドウが開いて、あなたはそこで自由に色を作る事が出来ます。

9.4.2 Region

Regionという概念があります。これからある操作をしようと思っている領域の事です。幾つかの機能はこのRegionを設定してからでないと実行できません。

Regionを設定するには矩形切り出しもしくはハサミ()を選択して、Paintingウィンドウで指定したい領域をドラッグして決めます。うまく設定できたら目印としてRegionの上下左右、四隅の合計8点に黒い小さな四角が表示されます。

こうしてRegion設定が出来たら、以下のような操作が可能になります。

移動 マウスの左ボタンでRegionをドラッグしてください。

削除 Paintingウィンドウの Editメニューから Cutを選択すると削除できます。

複写 Paintingウィンドウの Editメニューから Duplicateを選択すると複写でき、それが新しいReginonとして扱われます。新しいRegionを移動しても元のRegionの図形はその場に残っています。

貼付 Paintingウィンドウの Editメニューから Pasteを選択すると直前に Cutもしくは Copyした Regionを張り付ける事が出来ます。

回転 マウスの中ボタンでRegionの端の方をつまんでドラッグして下さい。

更にPaintingウィンドウのRegionメニューが選択できるようになり、そこでは反転、回転、色反転、エッジ抽出、エンボス加工、油絵風加工など色々な加工が出来ます。

9.4.3 拡大表示

Paintingウィンドウの Imageメニューから Change Zoom... を選択すれば拡大倍数が選べます。ここでの数値は倍数でパーセントではありません。間違って200%のつもりで200などと指定すると大変な事になりますので注意しましょう。

また、写真の加工などをしようとすると、一画素ごとに加工したいと思う時が多いでしょう。同じく Imageメニューで Fat Bitsを選択すると Paintingウィンドウとは別に Fat Bitsウィンドウが表示され、そこでは非常に大きく拡大された表示が行なわれます。Fat Bitsウィンドウでは Paintingウィンドウと同じ描画、加工の操作が行なえます。

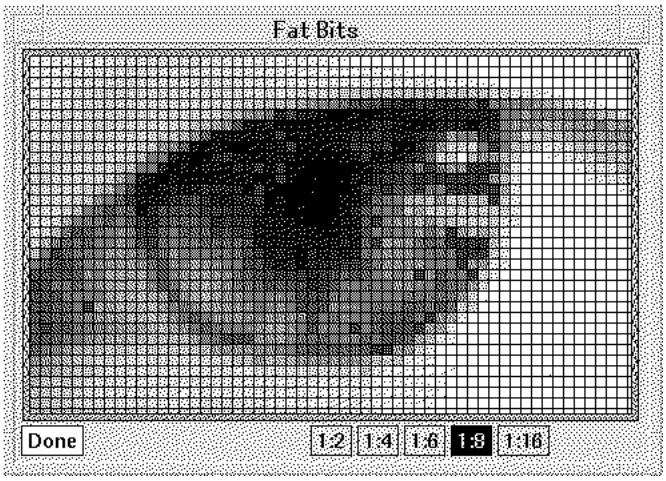


図 9.6 Fat Bits ウィンドウ (目を拡大表示したところ)

FatBits で表示されている領域が Painting ウィンドウの中に小さな四角い枠で表示されているのが判るでしょう。この枠をマウスでドラッグする事によって FatBits による拡大表示の場所も移動します。

9.5 マニュアルなど

シェルプロンプトから `man xpaint<return>` でマニュアルは表示されますが、そこでは xpaint の機能に付いてはほとんど何も教えてくれません。そのかわり Toolbox ウィンドウや Painting ウィンドウの Help メニューから以下のような Help ウィンドウを表示させる事が出来ます。

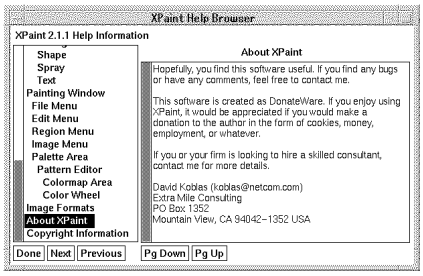


図 9.7 Help ウィンドウ

このウィンドウを閉じるには左下にある Done ボタンをクリックして下さい。

第 10 章

NQS

大勢の人たちで一つのコンピュータを共用していて、科学技術計算などの大規模な処理を実行したい人がたくさんその中に含まれていたとします。例えば 10 人の利用者がそれぞれ 3 時間かかるような大規模な計算を同時に同じコンピュータに掛けたとすると、全員の処理が終了するのが合計 30 時間よりも遥かに長くなったりする場合が往々にしてあります。これは一時に負荷のかかる計算をやらせた為に、計算機が過負荷状態になってしまうことから来ています。これを回避する為には、一つずつプログラムを実行し、先のプログラムの実行が終ったら次のプログラムの実行に掛かるようにするのが最善です。

このようなシステムは一般にバッチシステムと言われ、科学技術計算を良く行なうコンピュータ[1]には大抵実装されているもので、そのような要求のある人たちにとっては大変便利なもので、VMS[2]や MSP[3] などの OS には実装されています。

しかし、その生まれの経緯から Unix にはもともとバッチキューという概念がありません。NQS はこのバッチキューを Unix 上で実現するソフトウェアです。cc 環境では cc2000 マシン（SPARCcenter2000）に実装されています。ここでは cc2000 上の NQS を例にとって、NQS の使い方を説明します。

10.1 今どんなバッチキューがあるか？

バッチキューの確認には `qstat -b` とします。

```
% qstat -b
```

です。以下に例を示しておきます。

```
cc2000(411)% qstat -b
==========================================
NQS Version: 2.3  BATCH QUEUES on cc2000
==========================================
QUEUE NAME      STATUS   TOTAL   RUNNING  QUEUED   HELD TRANSISTION
------------------------------------------------------------------------------
BATCH           AVAILBL    0      0/4/4      0       0       0
I               AVAILBL    0      0/2/2      0       0       0
cc2000(412)%
```

[1] もしくは古典的なコンピュータ
[2] DEC-3500 などの OS です。
[3] FACOM の OS です。

STATUS のところが available になっていますから、利用可能なキューはこの時点で BATCH と I の二つですね。また、RUNNING 表示に出てくる A/B/C のそれぞれの数字は、各キューで

A : 実行されているジョブの数

B : 一人のユーザが同時に走らせることが出来るジョブの数

C : 同時に実行可能なジョブの数

をそれぞれ示しています。SPARCcenter2000 に限って言えば、このマシンは 4 つの CPU を同時に並行動作させられますから、同時に 4 つ以下の数のジョブが流れるぶんには全く負荷は掛かりません。(もちろんメモリを食いすぎると負荷になりますが。)

このうち BATCH は、デフォルトキューとなっていますから、これ以降の全てのコマンドでキュー指定をしなかった場合は、この BATCH という名前のキューが対象となります。(どのキューがデフォルトキューかという情報は表示されません。)

10.2 バッチジョブの投入

バッチジョブの投入には qsub コマンドを利用します。

```
% qsub [ -q QUEUE ] [ SHELLSCRIPT ]
```

です。QUEUE にはキュー名を、SHELLSCRIPT には、実行したいコマンド列を自分の login shell の文法で書かれたファイルの名前を書きます。

デフォルトキューで実行する場合は-q QUEUE を外してください。スクリプトが簡単に済むなら省略が可能です。その場合は標準入力にコマンド列を書くように要求してきますから、

```
% qsub [ -q QUEUE ]
```

だけで実行し、そのあとにずらずらとコマンド列を書くという形になります。

例として bb というファイルをデフォルトキューで実行します。以下の例を見て判るように、今回のスクリプトは非常に単純で、単に自分で作った resolv というプログラムをホームディレクトリのすぐ下の projectX という名前のサブディレクトリに置いていて、それにパラメタを与えて実行するだけです。date コマンドを前後に入れて時間を表示させるようにしています。

```
cc2000(360)% cat bb
cd projectX
date
resolv 100 x10 y20 z30
date
cc2000(361)% qsub bb
Request 16.cc2000 submitted to queue: BATCH.
cc2000(362)%
```

このように、16 番と言うエントリ番号を貰いました。

10.2.1 ジョブの始まりと終りにお知らせを貰う

自分が投入したジョブの前に、誰かが非常に時間の掛かるジョブを同じキューに投入していたりすると、自分のジョブはいつ実行されるか判りません。また、いつ終わるのかも判りません。

ジョブを投入するときに -mb , -me オプションを付けておけば、それぞれジョブが始まったときと終わったときに mail をくれます。両方のオプションを付けた例を示しておきます。

```
cc2000(402)% qsub -mb -me bb
Request 24.cc2000 submitted to queue: BATCH.
cc2000(403)%
```

こうしておくと、Subject が 24.cc2000 beginning. などというメイルをお知らせに貰う事が出来ます。

10.3 ジョブの標準出力

ジョブの結果がどのように残ってプログラマに与えられるのかはプログラム次第ですが、普通に shell script を実行したときに発生する標準出力と標準エラー出力は自動的にファイルとして生成され、残されます。投入したジョブが終わると、ジョブを投入したときのスクリプトファイルの名前の後に ".o" とエントリ番号を付けたファイルが標準出力の内容を残します。また、".e" とエントリ番号を付けたファイルが標準エラー出力の内容を残しています。もしもスクリプトファイルを利用せずに標準入力からジョブの投入を行った場合は、STDIN に続いて".o"と".e"が付いたファイルが生成されます。

例を示しておきます。まずはエラーファイル。

```
cc2000(425)% more bb.e16
cc2000(426)%
```

空っぽでした。続いて標準出力。

```
cc2000(426)% more bb.o16
Sun Microsystems Inc.   SunOS 5.2       Generic March 1993
1994 年 01 月 26 日 (水) 19 時 43 分 01 秒 JST
Start
End.
1994 年 01 月 26 日 (水) 19 時 43 分 19 秒 JST
cc2000(427)%
```

ああ、うまく行ったようです。安心安心。

これらのファイルは自動的に、qsub でジョブを実行した時のディレクトリに生成されます。

10.3.1 標準出力、エラー出力のファイルを変更したい

もしもファイル名や生成される場所を変えたい場合は、qsub に-o , -e オプションを加えます。

```
% qsub [ -o STDOUT ] [ -e STDERR ] ...
```

です。STDOUT には標準出力の宛先ファイルを、STDERR には標準エラー出力の宛先ファイルを指定します。例として両方のオプションを付けたものを示します。

```
cc2000(431)% qsub -o cc -e ee bb
Request 29.cc2000 submitted to queue: BATCH.
cc2000(432)%
```

標準出力ファイルが cc に、エラーファイルが ee に作成されました。それぞれ cc.o29 や ee.e29 にはならない事に注意してください。

また、エラー出力と標準出力を同じファイルに出したいときは-eo オプションを使います。

```
% qsub -eo [ -o STDOUT ] ...
```

-eo オプションを付けて、なおかつ-o オプションでファイルを指定すると、エラー出力も、標準出力も共に STDOUT に指定したファイルに出力されます。

10.4 ジョブの状態表示

自分がエントリーしたジョブの状態を見るには、qstat コマンドを利用します。

```
% qstat { ENTRY | QUEUE }
```

です。ENTRY には上記のエントリ番号がはいります。QUEUE にはキュー名が入ります。例として先ほどの 16 番の状態を見てみます。

```
cc2000(362)% qstat 16
============================================================
NQS Version: 2.3  BATCH DEVICE PIPE REQUESTS on cc2000
============================================================
REQUEST       NAME            OWNER       QUEUE       PRI NICE CPU     STATE
   16.cc2000  bb              yasuda      BATCH       31   20  36000   RUNNING
cc2000(363)%
```

最後の STATE が RUNNING ですから、無事に今処理されている最中という事です。

ところで他の全てのジョブの状態を知るには qstat -a とします。

```
% qstat -a [ QUEUE ]
```

です。QUEUE にはキュー名が入ります。省略すると全てのキューの情報が表示されます。例を示しておきます。

```
cc2000(365)% qstat -a
============================================================
NQS Version: 2.3  BATCH DEVICE PIPE REQUESTS on cc2000
============================================================
REQUEST       NAME            OWNER       QUEUE       PRI NICE CPU     STATE
   17.cc2000  bb              yasuda      BATCH       31   20  36000   RUNNING
   18.cc2000  bb              yasuda      BATCH       31   20  36000   QUEUED
   28.cc2000  bb              yasuda      I           31   20  36000   RUNNING
cc2000(366)%
```

現在 17 番が BATCH キューで実行中で、18 番は待ち状態にあるということが判ります。また、28 番が I キューで実行中です。

10.5 流れているジョブの中身を確認する

qcat コマンドで、キューに入っているジョブの内容などの確認が出来ます。

10.5.1 ジョブの記述を確認する

既に投入したジョブの記述を確認するには qcat コマンドを利用します。

```
% qcat ENTRY
```

です。ENTRY には上記のエントリ番号がはいります。例として先ほどの 16 番の中身を見てみます。

```
cc2000(363)% qcat 16
>>> Input file  is /usr/spool/nqs/private/root/data/+++++F/++++F+++++0+++.
#
date
cd projectX
resolv 100 x10 y20 z30
date
cc2000(364)%
```

という表示が出てきました。勿論 qcat コマンドは現在実行中、もしくは実行待ちのジョブに対してしか効きません。既に実行が終ってしまったジョブについてはどうにもなりません。

10.5.2 流れているジョブの途中経過を確認する

現在実行中のジョブの出力を実行の最中に確認するには qcat -o コマンドを利用します。

```
% qcat -o ENTRY
```

です。ENTRY には上記のエントリ番号がはいります。例として先ほどの 16 番の中身を見てみます。

```
cc2000(364)% qcat -o 16
Sun Microsystems Inc.   SunOS 5.2       Generic March 1993
1994 年 01 月 26 日 (水) 19 時 43 分 01 秒 JST
Start
cc2000(365)%
```

という表示が出てきました。勿論 qcat -o コマンドは現在実行中のジョブに対してしか効きません。既に実行が終ってしまったジョブについては出力ファイルを見ることになります。

非常に長いジョブを実行する場合は処理の切れ目などで時々標準出力にメッセージを出すようにプログラムを書いておくと qcat -o 機能をうまく利用できて便利でしょう。

なお、-o オプションの代わりに -e オプションを与えることによって（もしあれば）エラー出力の内容を実行途中に確認できます。

10.6 ジョブの実行を保留する

待ち状態にあるジョブの実行を一時保留するには qhold コマンドを利用します。

```
% qhold ENTRY
```

です。ENTRY には上記のエントリ番号が入ります。以下に例を示します。

```
cc2000(459)% qhold 33
Request 33 has been held.
cc2000(460)%
```

hold されたジョブを再び解放するには、qrls コマンドを利用します。

```
% qrls ENTRY
```

です。ENTRY には上記のエントリ番号が入ります。以下に例を示します。

```
cc2000(464)% qrls 33
Request 33 has been released.
cc2000(465)%
```

qhold は、現在実行中のジョブには適用出来ません。qstat -a でチェックしたときに、QUEUED になっているものだけです。以下に qstat -a の例を示します。

一番右の STATE の列を見てください。RUNNING が現在実行中で、QUEUED が現在実行待ちのものです。31 番は HOLDING になっていますね。これは先ほど実行した qhold のせいで、実行が保留されているという事です。

```
cc2000(462)% qstat -a
============================================================
NQS Version: 2.3  BATCH DEVICE PIPE REQUESTS on cc2000
============================================================
REQUEST       NAME            OWNER       QUEUE       PRI NICE CPU     STATE
   32.cc2000  bb              yasuda      BATCH       31   20  36000   RUNNING
   33.cc2000  bb              yasuda      BATCH       31   20  36000   QUEUED
   31.cc2000  bb              yasuda      BATCH       31   20  36000   HOLDING
cc2000(463)%
```

10.7 ジョブの実行を停止、削除する

ジョブの実行を停止するには qdel コマンドを利用します。

```
% qdel [ -SIGNAL ] ENTRY
```

です。現在実行中のジョブの実行を停止させるには、SIGNAL に 9 を入れると良いでしょう。待ち状態のジョブの実行を取り消すだけなら、-SIGNAL は省略可能です。ENTRY には上記のエントリ番号が入ります。以下に例を示します。

```
cc2000(561)% qdel -9 54
Request 54 is running, and has been signalled.
cc2000(562)%
```

10.8 もっと詳しいキューの情報を調べる

最初に qstat -b コマンドによってキューの情報を調べる例を挙げましたが、もっと詳しく、それぞれのキューに割り当てられている最大メモリ量、最大 CPU タイムなどを調べるには qstat コマンドに-bl オプションを与えます。

```
% qstat -bl [QUEUE]
```

です。QUEUE にはキュー名が入ります。省略すると全てのキューの情報が表示されます。例を示しておきます。

```
cc2000(562)% qstat -bl BATCH
=====================================================
NQS version: 2.3        BATCH QUEUE: BATCH.cc2000       status: AVAILBL
=====================================================
                                        Priority: 0
ENTRIES:
        Total: 0        Running: 0
        Queued: 0       Held: 0         Transiting: 0

COMPLEX MEMBERSHIP:

RESOURCES:
  Per-proc core file size limit= 1 Mbytes <DEFAULT>
  Per-process data size limit  = 10 Mbytes
  Per-proc perm file size limit= 10 Mbytes
  Per-proc execution nice value= 0 <DEFAULT>
  Per-process stack size limit = 10 Mbytes
  Per-process CPU time limit   = 300.0
  Per-process working set limit= 10 Mbytes

ACCESS
  Unrestricted access
```

上記の Per-proc data size limit などが一つのジョブ当たりの最大メモリ量です。Per-process CPU time limit などが一つのジョブ当たりの最大 CPU タイムとなります。これらの制限を越えると、その時点でジョブは強制的に終了させられます。

10.9 エラーメッセージ

いつもの通りにジョブが動くはずのところが、以下のようなエラーが出てくる時があります。

10.9.1 あなたの所在地はどこですか？

これは tty 情報が取れない環境で biff を起動したときに表示されるエラーです。biff コマンドが $HOME/.login ファイルなどに含まれているのではありませんか？対処方法はありません。無視するか biff コマンドを使わないようにするしかありません。

10.9.2　警告: tty にアクセスできないため、このシェルでジョブ制御はできません ...

```
Warning: no access to tty; thus no job control in this shell...
```
もしくは
```
Warning: no access to tty (Bad file number).
Thus no job control in this shell.
```

これは tty 情報が取れない環境で `csh` もしくは `tcsh` を起動したときに表示されるエラーです。対処方法はありません。無視してください。

10.10　マニュアルなど

cc2000 にインストールされている NQS に関するドキュメントとしては cc2000:/NF/local/Solaris2J/lib/nqs 以下に troff の mm マクロ形式と、それを PostScript 形式に落したものが用意してあります。

また、一般的なオンラインマニュアルがインストールされているはずですので、`man -k nqs` などとして探せば大抵のコマンドの詳細情報が得られます。

付録 A

リファレンス

A.1 UNIX コマンド

ここでは UNIX のコマンドのうち、代表的なものを示します。例示は以下のような体裁をとります。

command : コマンドの働き (full spec of command)

`command [option]` *string...*

`option`	(full spec of option)	オプションの働き
string		引数の意味

一行目にコマンド名とその概略を、二行目に実際にコマンドを実行する時のスタイルを、三行目以降に二行目で使われた記号の説明をしています。

以下にそれぞれの意味を説明します。

command	まさにコマンド名です。
コマンドの働き	そのコマンドがどのような働きをするかを簡単に書いています。
(full spec of command)	コマンド名はその働きを示す単語の省略形である事が良くあります。その元となった単語です。
`option`	そのコマンドに適用できるオプションです。代表的なものだけを書いています。
(full spec of option)	オプション記号はその働きを示す単語の省略形である事が良くあります。その元となった単語です。
オプションの働き	そのオプションがどのような働きをするかを簡単に書いています。
string	そのコマンドに適用できる引数です。
引数の意味	与えられた引数がどのようなものとして解釈されるかを簡単に書いています。

オプションや引数の与え方のスタイルを示している二行目の部分では、`[ ]` | `{ }` などの記号を使って表現しています。以下のルールに従って解釈して下さい。

- オプションや引数が `[ ]` で囲まれている場合がありますが、これは囲まれた部分はなくてもいいと言う事を意味しています。
  特にオプション記号が並んでおり、それが `[ ]` で囲まれていた場合は囲まれた部分から任意の記号を組み合わせて書いても良い事を示しています。例えば `[-abc]` と書かれていた場合は `-a -b -ab -abc -cb` などのいずれの書き方をしても構わないのです。

- オプション記号が並んでいる時に | で仕切られている場合がありますが、これは仕切られた両側の記号のいずれか一つを選ぶ事を示しています。
  例えば `[-a|-b]` と書かれていた場合はオプション無しか `-a` か `-b` かいずれかで実行しなければなりません。もしも `[-a|-b]` ではなく `{-a|-b}` と書かれていた場合はオプションなしは許されず、`-a` か `-b` かいずれかを選ばなければなりません。

- 引数など斜体文字で書かれた部分については、その部分をファイル名やキーワードなど何か別の文字列で置き換えて与える事を意味しています。
  特に引数の後ろに ... と書かれていた場合は引数を空白で区切りながら複数個書いても良い事を示しています。例えば *string...* と書かれていた場合は `sample test try` と書いても構わないのです。

A.1.1 ファイル管理に関するコマンド

ls : ファイルの一覧を表示する (list)

`ls [-lagdF]` *[names...]*

`-l`	(long format)	詳細情報を表示する
`-a`	(all)	隠しファイルも含めて表示する
`-g`	(group)	グループ情報も表示する
`-d`	(directory)	ディレクトリ以下をたどらない
`-F`		ファイルの属性が判るようにディレクトリには/を、実行ファイルには*をつける
names		ファイル名

pwd : カレントディレクトリの表示 (print working directory)

`pwd`

cd : カレントディレクトリの移動 (change directory)

`cd` *directory*

directory	移動先のディレクトリ名

mkdir : ディレクトリの作成 (make directory)

`mkdir` *directory...*

directory	作成するディレクトリ名

rmdir : ディレクトリの消去 (remove directory)

`rmdir` *directory...*

directory	消去するディレクトリ名

cp : ファイルのコピー (copy)

`cp [-i|-f]` *source-file destination-file*
`cp [-i|-f]` *source-file... destination-directory*
`cp -r[i|f]` *source... destination-directory*

`-i`	(inquiry)	コピー先でファイル名が既存の場合、上書きするかを問い合わせする
`-f`	(force)	コピー先でファイル名が既存であっても問い合わせをせずに上書きする
`-r`	(recursive)	ディレクトリ構造ごと階層的にコピーする
source-file		コピー元ファイル
destination-file		コピー先ファイル
destination-directory		コピー先ディレクトリ
source		コピー元ファイルもしくはディレクトリ

mv : ファイルの移動 (move)

`mv [-i|-f]` *source-file destination-file*
`mv [-i|-f]` *source... destination-directory*

`-i`	(inquiry)	移動先でファイル名が既存の場合、上書きするかを問い合わせする
`-f`	(force)	移動先でファイル名が既存であっても問い合わせをせずに上書きする
source-file		移動元ファイル
destination-file		移動先ファイル
source		コピー元ファイルもしくはディレクトリ
destination-directory		移動先ディレクトリ

rm　：ファイルの消去 (remove)

`rm [-[r][i|f]]` *[name...]*

`-r`	(recursive)	ディレクトリ構造ごと階層的に消去する
`-i`	(inquiry)	消去するかどうか確認する
`-f`	(force)	消去するかどうか確認しない

A.1.2　ファイルに関する雑多なコマンド

cat　：ファイルの内容を表示する (catalog)

`cat [-n]` *[name...]*

`-n`	(number)	行番号をつける
name		ファイル名

more　：ファイルの内容を一ページずつ表示する

`more` *[name...]*

name	ファイル名

head　：ファイルの先頭を表示する

`head [-`*number*`]` *[name...]*

`-`*number*	先頭から *number* 行めまで表示する
name	ファイル名

tail　：ファイルの末尾を表示する

`tail [-`*number*`|+`*number*`]` *[name...]*

`-`*number*	末尾 *number* 行を表示する
`+`*number*	*number* 行めから末尾まで表示する
name	ファイル名

file　：ファイルの種類を表示する

`file` *[name...]*

name	ファイル名

touch　：ファイルの更新日付を新しくする

`touch [-c]` *name...*

name が存在していない場合にはその名前で空ファイルを作成する。

`-c`	*name* が存在していない場合にはファイルを作成しない。
name	ファイル名

od　：ファイルの内容をダンプする (octal dump)

`od [-bcx]` *[name...]*

`-b`		8 進数で表示する
`-c`		可視コードは文字表示する
`-x`	(hex)	16 進数で表示する
name		ファイル名

split　：ファイルを行単位で分割する

`split [-`*number*`]` *[name [prefix]]*

number 行ごとに *name* ファイルを分割し、*prefix* に aa から zz までアルファベット順に合成した番号を付けた複数のファイルとして出力する

`-`*number*	分割する行単位。デフォルトでは 1000 が与えられる
name	元ファイル名
prefix	出力ファイルの頭に付く名前

du　：ファイルの量を表示する (disk use)

`du [-sk]` *[name...]*

`-s`	(size)	*name* それぞれの総合計のみを表示する
`-k`	(kilobyte)	表示単位をキロバイト単位にする
name		ファイル名

ln　：リンクを作成する (link)

`ln [-fns]` *link [name...]*

`-f`	(force)	リンク先 *link* が書き込み禁止であっても確認の問い合わせをしない
`-n`		リンク名 *link* が既存であれば上書きしない
`-s`	(symbolic link)	シンボリックリンクを作成する
link		リンク名
name		ファイル名

A.1.3　テキスト処理に関するコマンド

wc　：ファイルの単語数などを調べる (word count)

`wc [-clw]` *[name...]*

`-c`	(character)	文字数（但しバイト数）を数える
`-l`	(line)	行数を数える
`-w`	(word)	単語数を数える
name		ファイル名

diff　：二つのファイルの差分を表示する (difference)

`diff [-biw]` *file1 file2*

`-b`	(ignore blank)	行末の空白を無視する
`-i`	(ignore case)	大文字と小文字を区別しない
`-w`		空白及びタブ文字を無視する
file1		ファイル名
file2		ファイル名

diff3　：三つのファイルの差分を表示する (differences of 3 files)

`diff3` *file1 file2 file3*

file1	ファイル名
file2	ファイル名
file3	ファイル名

sort　：ファイルの内容を行単位で順に並べ変えて表示する

`sort [-cunfr]` *[name...]*

sort には非常に多くのオプションがある。`man sort` などして確認するのが良い。

`-c`	(check)	正しくソートされているか否かのチェックのみ行なう
`-u`		同一行を出力しない
`-n`	(number)	数値表現として解釈してソートする
`-f`		大文字と小文字を無視する
`-r`	(reverce)	並び順を逆にする
name		ファイル名

grep　：パターンをファイルから検索して表示する (general regular expression)

`grep [-[AB]`*number*`] [-cinlv]` *pattern [name...]*

pattern には正規表現が利用できる。

-number		マッチした行の前後 *number* 行を含めて表示する
`-A`*number*	(after)	マッチした行の後ろ *number* 行を含めて表示する
`-B`*number*	(before)	マッチした行の前 *number* 行を含めて表示する
`-c`	(count)	マッチした行数を数えるだけ
`-i`	(ignore case)	大文字と小文字を無視する
`-n`	(number)	マッチした行を行番号とともに表示する
`-l`		少なくとも一つはマッチした行を持つファイルの名前を表示する
`-v`	(invert)	マッチしなかった行を表示する
pattern		検索するキーワード
name		ファイル名

egrep : grep の完全版 (expression grep)

`egrep [-cinlv]` *pattern [name...]*

pattern には全ての正規表現が利用できる。

`-c`	(count)	マッチした行数を数えるだけ
`-i`	(ignore case)	大文字と小文字を無視する
`-n`	(number)	マッチした行を行番号とともに表示する
`-l`		少なくとも一つはマッチした行を持つファイルの名前を表示する
`-v`	(invert)	マッチしなかった行を表示する
pattern		検索するキーワード
name		ファイル名

fgrep : grep の高速版 (fast grep)

`fgrep [-cinlv]` *pattern [name...]*

pattern には正規表現は利用できない。簡略化された表現だけを提供する。

`-c`	(count)	マッチした行数を数えるだけ
`-i`	(ignore case)	大文字と小文字を無視する
`-n`	(number)	マッチした行を行番号とともに表示する
`-l`		少なくとも一つはマッチした行を持つファイルの名前を表示する
`-v`	(invert)	マッチしなかった行を表示する
pattern		検索するキーワード
name		ファイル名

tr : 文字を置き換える (translate character)

`tr [-ds]` *[string1 [string2]]*

string1, *string2* には`"\012"`などとして 8 進数表記も可能。`[a-z]` などと `a` から `z` の連続した文字を意味する省略記法も可能。

`-d`	(delete)	標準入力から *string1* に含まれる文字を削除して標準出力に出す
`-s`		標準入力の連続する *string1* を一つの *string2* に置換して標準出力に出す

sed : ストリームエディタ (stream editor)

`sed [-n][-e` *script*`][-f` *scriptfile*`]` *[name ...]*

`-n`	デフォルトの出力を抑制する
`-e`	編集スクリプト *script* のバリエーションは多数あるのでマニュアルを参照の事
`-f`	編集スクリプトを *scriptfile* ファイルから読みとる
name	ファイル名

spell : 単語のスペルチェックをする

`spell [-bvx]` *[name]*

spell コマンドは非常に高機能だがここでは基本機能のみ載せる

`-b`	(British)	British 綴りをチェックする
`-v`	(verbose)	その綴りでスペリングリストに入っていないすべての単語を表示し，その単語から可能性のある派生語が示する
`-x`		可能性のある語幹を等号 (=) と共に表示する
name		ファイル名

A.1.4 プリンタに関するコマンド

プリンタ一覧

プリンタ名	設置場所
ccpr01	計算機科学研究所 2 階ミニコン室（白くて小さな方）
ccpr02	計算機科学研究所 2 階ミニコン室（茶色の大きな方）
cgpr01	1 号館 2 階 11 情報処理教室
cgpr02	1 号館 2 階 11 情報処理教室
cgpr03	1 号館 2 階 11 情報処理教室
cgpr04	1 号館 2 階 11 情報処理教室
cspr01	2 号館 4 階 21 情報処理教室
cspr02	2 号館 4 階 21 情報処理教室
clpr01	3 号館 2 階 31 情報処理教室
clpr02	3 号館 2 階 31 情報処理教室
cepr01	5 号館 1 階 51 情報処理教室
c1kpr01	第 1 研究室棟 2 階共同利用室
c2kpr01	第 2 研究室棟 1 階共同利用室
c3kpr01	第 3 研究室棟 1 階共同利用室
c9pr01	9 号館

lpr　：プリンタに出力する (line printer)

`lpr` [`-P`*printer*] *[name...]*

cc 環境でプリンタに出力できる内容は単純なテキストファイルか、PostScript ファイルに限られる。

`-P`*printer*　　*printer* で示されるプリンタに出力する

name　　ファイル名

lpq　：プリント待ちキューの内容を表示する (line printer queue)

`lpq` [`-P`*printer*]

`-P`*printer*　　*printer* で示されるプリンタの待行列を表示する

lprm　：プリント待ちエントリを消去する (line printer remove entry)

`lpq` [`-P`*printer*] [`-`] *[job...]*

job は lpq コマンドであらかじめ確認する。

`-P`*printer*　　*printer* で示されるプリンタのエントリを処理対象にする

`-`　　自分が最後に出力したエントリを消去する

job　　*job* で示されるエントリを消去する

A.1.5 アクセス権、アクセス制御に関するコマンド

whoami　：現在の利用者名を表示する (who am i)

`whoami`

id　：現在の利用者名、現在のグループ名などを表示する

`id`

id コマンドは各 OS によって相違が見られるので、`man id` などして確認するのが良い。

groups　：所属するグループ名の一覧を表示する

`groups` *[username...]*

username　　表示させたい利用者名。デフォルトでは現在の利用者の利用者名が与えられる

newgrp　：新しいグループへのログイン (new group)

`newgrp` *[group]*

group　　グループ名。デフォルトでは現在の利用者のデフォルトグループが与えられる

chmod　：ファイルのアクセス権を変更する (change mode)

`chmod [-R]` *mode name...*

`-R`	(recursive)	*name* がディレクトリであった場合は階層的に処理を行なう
mode		下部参照
name		ファイル名

相対指定における *mode* は`{u|g|o|a}{+|-|=}{r|w|x}`となる。

対象		オペレータ		設定内容	
u	所有者	+	追加	r	読みだし
g	グループ	-	取消	w	書き込み
o	その他の人	=	強制	x	実行
a	全ての人				

絶対指定における *mode* は以下の数値の和となる。

値	対象	設定内容	値	対象	設定内容	値	対象	設定内容
0400	所有者	読みだし	0040	グループ	読みだし	0004	その他の人	読みだし
0200	所有者	書き込み	0020	グループ	書き込み	0002	その他の人	書き込み
0100	所有者	実行	0010	グループ	実行	0001	その他の人	実行

chgrp　：ファイルのグループを変える (change group)

`chgrp [-R]` *group name...*

`-R`	(recursive)	*name* がディレクトリであった場合は階層的に処理を行なう
group		変更したいグループ名
name		ファイル名

A.1.6　マニュアルに関するコマンド

man　：コマンドなどのマニュアルを表示する (manual)

`man [-s` *section*`]` *title*

`man -k` *keyword...*

`-s` *section*	*title* のセクションを *section* に限定する
`-k` *keyword*	*keyword* にマッチするタイトルの一覧を表示する
	keyword はコマンド名などの一部でも良い
title	コマンド名など

whatis　：コマンドなどの要約を表示する (what is)

`whatis` *title...*

title	コマンド名などのキーワード

which　：コマンドファイルの位置を表示する

`which` *command...*

command	コマンド名

whereis　：コマンドファイル、マニュアルファイルの位置を表示する

`whereis` *command...*

command	コマンド名

A.1.7　雑多なコマンド

date　：現在の日付を表示する

`date`

sleep　：実行を保留する

`sleep` *[time]*

time	待ち時間（秒）

cal : カレンダーを表示する (calender)

`cal` *[[month] year]*

オプションを全て省略すると今月のカレンダーを表示する。*year* だけを与えるとその年のカレンダーを表示する。*month* も与えるとその年のその月のカレンダーを表示する。

month	月の指定を 1 から 12 までで与える
year	年の指定を西暦で与える

echo : 引数を表示する

`echo [-n]` *[argument...]*

`-n`	(no return)	出力に改行文字を加えない
argument		引数

banner : 引数を花文字で表示する

`banner` *string...*

string	10 文字までの花文字にしたい文字列

clear : 画面を消去する

`clear`

bc : 電卓

`bc` *[name...]*

name	ファイル名。但しファイルの最後には quit を bc へのサブコマンドとして書く

look : 英単語を辞書検索する

`look [-df][-t`*c*`]` *string*

`-d`	(dictionary order)	文字、数字、タブ及びスペースが比較される
`-f`	(fold case)	大文字と小文字を区別しない
`-t`*c*	(termination character)	*c* とその後に続く文字を無視する
string		検索単語

tee : 標準入力を標準出力とファイルに書く

`tee [-ai]` *[name...]*

`-a`	(append)	ファイルへの出力を追加書きにする
`-i`	(ignore interrupts)	割り込みを無視する
name		ファイル名

script : 端末の操作をファイルに記録する

`script [-a]` *[name]*

script コマンドは新しくシェルを起動する。このシェルを終了することによって script コマンドも終了する。シェルを終了するには exit コマンドを利用すれば良い。

`-a`	(append)	*name* ファイルに対する出力を追加書きで行なう。デフォルトは上書き
name		ファイル名。省略するとデフォルトとして `typescript` が与えられる

df : 利用可能なディスクブロックを表示する (disk free)

`df` *[directory...]*

df コマンドは各 OS によって相違が見られるので、`man df` などして確認するのが良い。

directory	*directory* が存在するファイルシステムだけに限定して表示する

yppasswd : パスワードを変更する (YP password)

`yppasswd`

ypchfn : フルネームを変更する (YP change full name)

`ypchfn`

ypchfn コマンドは Sun OS 独自のコマンドで、cc 環境では cc2000 でしか動かない。これによってメイルの From: 行や finger などの表示に現れるローマ字の名前を変更できる。

ypchsh : login シェルを変更する (YP change shell)

`ypchsh`

ypchsh コマンドは Sun OS 独自のコマンドで、cc 環境では cc2000 でしか動かない。これによって login した後に実行されるシェルを変更できる。この表現が理解できない場合はこのコマンドは実行しない方がよい。変更できるシェルには限りが有り、`cat /etc/shells <return>`などして確認できる。

A.1.8 ファイル圧縮などに関するコマンド

tar : テープ用集積ファイル (tarfile) を扱う (tape archiver)

`tar [-][c|r|t|u|x][v][-b block][-f` *device*`]` *[name...]*

テープにファイルを書き込む時などに使うが、ブロックサイズなどはテープドライブに依存するので各デバイスなどのマニュアルで確認する事。

`-c`	(creat)	tarfile を *name* から作成する
`-r`	(replace)	tarfile に *name* を追加する
`-t`	(title)	tarfile の内容一覧を表示する
`-u`	(update)	tarfile を更新する。最後に tarfile へ *name* ファイルを書き込んでから変更があれば tarfile に追加する
`-x`	(extract)	tarfile からファイル *name* を抽出もしくはレストアする
`-v`	(verbose)	現在の状態を表示しながら実行する
`-b`	(block size)	ファイルのブロック化係数を *block* にする
`-f`	(device file)	tarfile を格納するデバイスを *device* にする
block		ブロックサイズ。デフォルトは 20
device		デバイスファイル名。-で標準入出力、通常のファイル名の指定も可能
name		ファイル名もしくはディレクトリ名

uuencode : バイナリファイルを可視コード文字列に変換する

`uuencode` *[name] label*

結果は標準出力に出る。*name* を省略すると標準入力から読んだデータを変換する。

name	ファイル名
label	変換したファイルを uudecode コマンドで戻す時のファイル名

uudecode : uuencode によって変換されたファイルをバイナリファイルに逆変換する

`uudecode` *[name]*

uuencode 時の *label* によって指定されたファイル名で結果が作成される。*name* を省略すると標準入力から読んだデータを変換する。

name	ファイル名

compress : ファイルを圧縮する

`compress [-cv]` *[name...]*

圧縮されたファイルは *name*.Z と言う名前で作成され、元の *name* ファイルは消去される。

`-c`		圧縮結果を *name*.Z ファイルに作成せずに標準出力に出力する *name* ファイルは消去されない
`-v`	(verbose)	圧縮率を表示する
name		ファイル名

uncompress : compress コマンドで圧縮されたファイルを復元する

`uncompress [-cv]` *[name...]*

name は最後が.Z でなければならない。復元されたファイルは *name* から.Z が外された名前で作成され、元の *name* ファイルは消去される。

`-c`		復元結果を *name* から.Z を取り除いたファイルに作成せずに標準出力に出力する *name* ファイルは消去されない
`-v`	(verbose)	圧縮率を表示する
name		ファイル名

gzip : ファイルを圧縮する (GNU zip)

`gzip [-cdhlv]` *[name...]*

圧縮されたファイルは *name*.z と言う名前で作成され、元の *name* ファイルは消去される。

`-c`		圧縮結果を *name*.z ファイルに作成せずに標準出力に出力する *name* ファイルは消去されない
`-d`	(decompress)	圧縮ファイルを復元する
`-h`	(help)	オプション一覧を表示する
`-l`	(list)	圧縮ファイルの内容一覧を表示する
`-v`	(verbose)	圧縮率を表示する
name		ファイル名

gunzip　: gzip および compress コマンドで圧縮されたファイルを復元する (GNU unzip)

`gunzip [-cv]` *[name...]*

name は最後が.Z もしくは.z でなければならない。復元されたファイルは *name* から.Z もしくは.z が外された名前で作成され、元の *name* ファイルは消去される。

`-c`		復元結果をファイルに作成せずに標準出力に出力する *name* ファイルは消去されない
`-d`	(decompress)	圧縮ファイルを復元する
`-h`	(help)	オプション一覧を表示する
`-l`	(list)	圧縮ファイルの内容一覧を表示する
`-v`	(verbose)	圧縮率を表示する
name		ファイル名

zcat　: compress コマンドで圧縮されたファイルを表示する (cat Z file)

`zcat` *[name...]*

name は最後が.Z でなければならない。`gzip` 圧縮ファイルを扱える場合もある。`gunzip` 同様のオプションが使える場合もある。

name	ファイル名

A.1.9　プロセスに関するコマンド

ps　: 現在処理中のプロセス一覧を表示する (process)

`ps [[-]aux]`

ps コマンドはオプションもその表示も各マシン、その OS 種類によって大きく異なる。`man ps` などして確認するのが良い。

`a`	(all)	全ての利用者のプロセスを表示する
`u`	(user)	利用者情報指向に整形して表示する
`x`		端末制御を持たないプロセスも含めて表示する

kill　: プロセスを終了させる

`kill [-l][-`*signal*`]` *process-id...*

process-id についてはあらかじめ ps コマンドで確認しておく。*signal* の種類によっては終了ではなくプロセスの再起動などが行なわれる場合がある。

`-`*signal*	*process-id* によって示されるプロセスに対して送られるシグナルの種類。
`l`	*signal* に利用可能な記号の一覧を表示する デフォルトでは-15 (-TERM) が与えられ、大抵これで終了させられる。強制終了の為には-9 (-KILL) を与える
process-id	終了させたいプロセスの番号

A.1.10　現在使っているコンピュータに関するコマンド

tty　: 端末回線名を表示する (tele type terminal)

`tty`

hostname　: ホスト名を表示する (host name)

`hostname`

uname　: OS に関する情報を表示する

`uname [-apsv]`

`-a`	(all)	全ての情報を表示する
`-p`	(processor)	プロセッサ型を表示する
`-s`	(operating system)	OS 名を表示する。これはデフォルトで与えられる
`-v`	(version)	OS のバージョンを表示する

uptime　: 起動されてからの時間と CPU 負荷率を表示する

`uptime`

A.1.11　利用者に関するコマンド

who　：現在利用している利用者の一覧を表示する

`who [-Hq]`

`-H`	(header)	見出しを出力する
`-q`		簡略化された形式で出力する

w　：現在利用している利用者と作業内容の一覧を表示する

`w [-hls]` *[username]*

見出しに uptime コマンドで表示されるのと同じ CPU 負荷率などが表示される

`-h`		見出しを表示しない
`-l`	(long format)	長い出力形式。これはデフォルトで与えられる
`-s`	(short format)	短い出力形式
username		利用者名 *username* に関する情報だけに限定する

finger　：利用者情報を表示する

`finger [-lms]` *[keyword...]*

`finger [-l]` *[username]@hostname...*

finger コマンドによって表示される最後にメイルを読んだ日付については cc 環境では正しく表示されない場合がありますので無視して下さい。

`-l`	(long format)	長い出力形式
`-m`		*keyword* を利用者名に限定して検索する
`-s`	(short format)	短い出力形式
keyword		利用者名、氏名などの断片
username		利用者名。省略した場合は *hostname* コンピュータを利用している利用者一覧が表示される
hostname		リモートコンピュータのホスト名

whois　：利用者情報を表示する (who is)

`whois [-h` *hostname*`]` *keyword*

whois サービスの内容や使い方はそれが行なわれているサーバに大きく依存する。

`-h` *hostname*	*hostname* コンピュータを whois サーバとする
keyword	検索するキーワード

A.1.12　ネットワークサービスに関するコマンド

telnet　：TELNET プロトコルによるリモートログインを行なう

`telnet` *[hostname]*

hostname	リモートログインするホスト名

rlogin　：リモートログインを行なう (remote login)

`rlogin [-8] [-l` *username*`]` *hostname*

`-8`	(8 bit)	通信に 8 ビットを利用する。デフォルトは 7 ビットの可能性が高い。
`-l` *username*		リモートログイン用の利用者名として *username* を使う デフォルトでは現在の利用者名が与えられる
hostname		リモートログインするホスト名

rsh　：リモートマシンにコマンドを実行させる (remote shell)

`rsh [-l` *username*`]` *hostname command*

`-l` *username*	リモートログイン用の利用者名として *username* を使う デフォルトでは現在の利用者名が与えられる
hostname	*command* を実行させるホスト名
command	実行させたいコマンド行

rcp　：リモートマシンのファイルをコピーする (remote copy)

`rcp` *[[username@]hostname:]source-file [[username@]hostname:]destination*
`rcp -r` *[[username@]hostname:]source... [[username@]hostname:]destination-directory*

コピー元、先のファイル名の記述の先頭に *hostname:*を付加する事によって「:」以降に記述されているファイルはそのホストに存在する事を意味する。*hostname:*を省略すればデフォルトとして現在のホスト名が与えられる。更に *username:*を与える事によって「@」以降に記述されているホストに対するアクセスは利用者名 *username* で行なわれる事を意味する。*username:*を省略すればデフォルトとして現在の利用者名が与えられる。

`-r`	(recursive)	ディレクトリ構造ごと階層的にコピーする
username		@以降に記述されるホストでの利用者名
hostname		:以降に記述されるファイルが存在するホスト名
source-file		コピー元ファイル
destination		コピー先ファイルもしくはディレクトリ
destination-directory		コピー先ディレクトリ
source		コピー元ファイルもしくはディレクトリ

A.1.13　シェル (tcsh) のサブコマンド

`cd`, `which`, `kill` などもシェルのサブコマンドだが、これらは普通のコマンドとして紹介している。ここでの記述は tcsh 特有のものを含んでいる事に注意。

シェル変数、環境変数に関するサブコマンド

set　：シェル変数を定義する

`set` *[variable[=string]]*

引数なしで現在定義されているシェル変数を表示する。*variable* を与えながら *string* を省略すると *variable* で示されるシェル変数を空文字列とする。

variable	シェル変数名
string	文字列

unset　：シェル変数の定義を解除する

`unset` *variable*

variable	シェル変数名

setenv　：環境変数を定義する (set environment variable)

`setenv` *[variable [string]]*

引数なしで現在定義されている環境変数を表示する。*variable* を与えながら *string* を省略すると *variable* で示されるシェル変数を空文字列とする。

variable	環境変数名
string	文字列

unsetenv　：環境変数の定義を解除する (unset environment variable)

`unsetenv` *variable*

variable	環境変数名

ジョブ制御に関するサブコマンド

jobs　：現在実行中のコマンドの一覧を表示する

`jobs [-l]`

`l`	(long)	プロセス番号も表示する

fg　：停止中のジョブを再開する (fore ground)

`fg` *[%job]*

job	`jobs` コマンドで表示されたジョブ番号

bg　：停止中のジョブをバックグラウンドで再開する (back ground)

`bg` *[%job]*

job	`jobs` コマンドで表示されたジョブ番号

stop　：バックグラウンドで実行中のジョブを一時停止する

`stop` *[%job]*

job　　`jobs` コマンドで表示されたジョブ番号

notify　：バックグラウンドで実行中のジョブの状態変化を知らせる

`notify` *[%job]*

job　　`jobs` コマンドで表示されたジョブ番号

wait　：全てのバックグラウンドジョブの実況終了を待つ

`wait`

%job **番号の指定方法**

`100`	プロセス番号 100 番
`%1`	ジョブ番号 1 番
`%`	直前に操作したジョブ
`%-`	一つ前のジョブ
`%cc`	実行コマンドが `cc` で始まるジョブ
`%?sort`	実行コマンドに `sort` を含むジョブ

プロセス番号は `ps` コマンドもしくは `jobs -l` コマンドで確認できる
ジョブ番号は `jobs` コマンドで確認できる

雑多なサブコマンド

alias　：コマンドの別名を定義する

`alias` *[name [string]]*

引数なしで現在定義されているエイリアス一覧を表示する。*name* を与えながら *string* を省略すると *name* で定義されているエイリアスを表示する。

name　　エイリアス名
string　　定義する文字列

unalias　：エイリアスの定義を解除する

`unalias` *name*

name　　エイリアス名

rehash　：コマンド参照の為の内部ハッシュテーブルを更新する (re-assign hash table)

`rehash`

unhash　：コマンド参照の為の内部ハッシュテーブルを使わなくする (unuse hash table)

`unhash`

login　：login シェルを終了し、新たに login する

`login`

logout　：login シェルを終了する

`logout`

exit　：シェルを終了する

`exit` *(expr)*

expr で与えられた数値は $status シェル変数に与えられる

expr　　数値もしくは数値になる式。() は省略できる

exec　：コマンドを実行する (execute)

`exec` *name*

name を実行する。実行が終っても制御は返ってこない。

name　　コマンド名もしくは実行可能なファイル名

source　：実行するコマンドの指定をファイルから読む

`source` *name*

ファイル名 *name* に書かれたコマンドを現在のシェルで実行する。

name		ファイル名

history　：実行したコマンドの履歴を見る

`history [-hr]` *number*

ファイル名 *name* に書かれたコマンドを現在のシェルで実行する。

`h`		イベント番号を表示に付けない
`r`	(reverse)	履歴から最新の *number* 個のイベントを逆順に表示する
number		表示するイベントの数

A.1.14　索引

A.2 UNIX でよく使われる記号など

ここでの記述は tcsh 特有のものを含んでいる事に注意。

A.2.1 シェル変数の一覧

argv	シェルスクリプトに渡された引数の列
status	直前のコマンド終了時の返り値を示す
cwd	Current Working Directory. カレントディレクトリを示す
home	ホームディレクトリを示す
path	コマンドパスを示す。set サブコマンドで設定可
user	現在のシェルプロセスのユーザ名を示す
uid	現在のシェルプロセスのユーザ id を示す
gid	現在のシェルプロセスのグループ id を示す
term	現在利用しているターミナル種別を示す。set サブコマンドで設定可
tty	現在利用しているターミナルの回線番号を示す
prompt	シェルプロンプトの形式を示す。set サブコマンドで設定可
autologout	セットした場合、この秒数の間入力がなければ tcsh は自動的に終了する
ignoreeof	セットした場合、端末から C-d を読んでもシェルは終了しない。終了には exit を使う
noclobber	セットした場合、シェルは既存のファイルに出力をリダイレクトする事を許さない。この設定を無視してコマンドを実行するには!を使う
noglob	セットした場合、*,?などのワイルドカードによるファイル名の補間を行なわない
nonomatch	セットした場合、ファイル名の補間に失敗してもエラーとせずコマンドを起動する
verbose	セットした場合、シェルはエイリアス、コマンド、ファイル名、変数などの置換えをした後のコマンドを表示しながら実行する
history	コマンド履歴の最大数を示す。set サブコマンドで設定可
histchars	履歴置き換え文字を示す。未設定の場合!を使う
savehist	ファイルに残すコマンド履歴の最大数を示す。set サブコマンドで設定可
shell	現在のシェルを示す
tcsh	tsh のバージョンを示す
version	現在の tcsh のバージョンを示す

A.2.2 環境変数の一覧

PATH	コマンドパスを示す。path シェル変数と連動している
HOME	ホームディレクトリを示す
PWD	カレントディレクトリを示す
SHELL	現在のシェルを示す
HOST	ホスト名を示す
HOSTTYPE	ホストコンピュータの種別を示す
LOGNAME	現在のシェルプロセスのユーザ名を示す
USER	現在のシェルプロセスのユーザ名を示す
LANG	言語環境を示す
TERM	現在利用しているターミナル種別を示す。term シェル変数と連動している
MANPATH	man コマンドが検索するマニュアルファイルの置き場所を示す
DISPLAY	X ウィンドウアプリケーションの表示画面先を示す
EDITOR	標準のエディタを示す
PAGER	標準のページャを示す
PRINTER	標準のプリンタ名を示す
TEXFONTPATH	TEX のフォントディレクトリを示す
TEXTFMPATH	TEX のフォントディレクトリを示す
ARCH	ホストコンピュータのアーキテクチャを示す。cc 環境特有
ENVIRON	ホストコンピュータの環境種別を示す。cc 環境特有
XENVIRON	ホストコンピュータの X ウィンドウの環境種別を示す。cc 環境特有
COMMON	共通設定ディレクトリを示す。cc 環境特有

A.2.3 リダイレクション記号など

`<`	標準入力をファイルから読む
`<< string`	文字列 `string` が入力行の先頭に現れるまで標準入力を読む
`>`	標準出力をファイルに書く
`>>`	標準出力をファイルに追加書きする
`>&`	エラー出力をファイルに書く
`>>&`	エラー出力をファイルに追加書きする
`>!`	標準出力をファイルに書く (`noclobber` シェル変数による保護を無視)
`>>!`	標準出力をファイルに追加書きする (`noclobber` シェル変数による保護を無視)
`>&!`	エラー出力をファイルに書く (`noclobber` シェル変数による保護を無視)
`>>&!`	エラー出力をファイルに追加書きする (`noclobber` シェル変数による保護を無視)
`|`	標準出力をパイプに書く
`|&`	エラー出力もパイプに書く

リダイレクション記号などを使ったコマンド実行の例

`command`	通常のコマンド実行
`command &`	バックグラウンドでのコマンド実行
`command1 ; command2`	`command1` の実行が済めば `command2` を実行する
`(command1 ; command2)`	`command1 ; command2` に同じ。但し単一コマンドのようにシェルは扱う
`command1 | command2`	通常のパイプ付き実行
`command1 |& command2`	エラー出力を含めたパイプ付き実行
`command1 && command2`	`command1` の実行が成功すれば `command2` を実行する
`command1 || command2`	`command1` の実行が失敗すれば `command2` を実行する
`(command > outfile) >& errorfile`	標準出力とエラー出力を分ける

A.2.4 ファイル指定のワイルドカードなど

`*`	任意のゼロ個以上の文字
`?`	任意の一文字
`[characters]`	`[]` に囲まれた文字列 `characters` に含まれる任意の一文字
`[char1-char2]`	文字 `char1` から `char2` までの範囲に含まれる任意の一文字
`{string1,string2,...}`	文字列 `string1` もしくは `string2` などのいずれか
`.`	カレントディレクトリ
`..`	カレントディレクトリの一つ上の階層のディレクトリ
`~`	自分のホームディレクトリ
`~username`	ユーザ名 `username` のホームディレクトリ

A.2.5 コマンド履歴を扱う為の表記法

`!!`	直前のコマンド行
`!n`	n 番めのコマンド行
`!-n`	n 番前のコマンド行
`!string`	`string` から始まる最近のコマンド行
`!?string`	`string` を含む最近のコマンド行
`!$`	直前のコマンド行の最後の単語
`!*`	直前のコマンド行の 1 番めから最後の単語（つまりコマンド以外の全ての引数）
`!n:$`	n 番めのコマンド行の最後の単語
`!n:^`	n 番めのコマンド行の最初の単語
`!n:m`	n 番めのコマンド行の m 番めの単語
`!n:m-l`	n 番めのコマンド行の m 番めから l 番めの単語
`!n:*`	n 番めのコマンド行の 1 番めから最後の単語（つまりコマンド以外の全ての引数）
`^str1^str2^`	直前のコマンドの `str1` を `str2` に置き換える

A.2.6 正規表現

a	a（通常の文字）にマッチする
.	任意の一文字にマッチする
^	行頭にマッチする
$	行末にマッチする
^.....$	5 文字の行にマッチする
|	前後の正規表現のいずれかにマッチする
ab|cd	ab もしくは cd のどちらかにマッチする
ab|cd|ef	ab、cd もしくは ef のいずれかにマッチする
(ab|cd)(12|34)	ab12 ab34 cd12 cd34 のいずれかにマッチする
[abc]	abc のどれか一文字にマッチする
[^abc]	abc のどれか一文字以外にマッチする
[a-z]	a から z までの範囲のどれか一文字にマッチする
[a-hxyz0-9]	abcdefghxyz0123456789 のどれか一文字にマッチする
?	?直前の正規表現のゼロ個ないしは一個にマッチする
ab?c	ac abc にマッチする。abbc などにはマッチしない
*	*直前の正規表現のゼロ個以上の繰り返しにマッチする
ab*c	ac abc abbbc などにマッチする。ab1c などにはマッチしない
a.*c	ac abc abbbc a123c などにマッチする
+	+直前の正規表現の一個以上の繰り返しにマッチする
ab+c	abc abbc abbbc などにマッチする。ac ab1c などにはマッチしない
a.+c	abc abbc abbbc a1c などにマッチする。ac にはマッチしない
\{num\}	直前の正規表現の num 個の繰り返しにマッチする
\{num,\}	直前の正規表現の num 個以上の繰り返しにマッチする
\{num1,num2\}	直前の正規表現の num1 個から num2 個までの繰り返しにマッチする

\ ^ $. [ ] ( ) | * + ?は普通の文字ではなく、意味を持ったメタキャラクタである。メタキャラクタに使われている記号をそのまま表現したい時は\$などのように \ 記号に続けて書く。\ 記号を表現したい時は\\である。

A.3 Mule コマンド

表記方法

C-h	コントロールキーを押しながら h を押す
M-x ABC	エスケープキーを押した後に x を押して普通に ABC と打つ
M-C-x	エスケープキーを押した後にコントロールキーを押しながら x を押す
<Space>	スペースキーを押す
<return>	リターンキーを押す
<Delete>	デリートキーを押す

A.3.1 絶対覚えておいた方がいいもの

mule <return>	Mule を起動する
C-x C-c	Mule を終了する
C-h T Japanese <return>	Mule(日本語) のチュートリアルを表示する
C-x C-f *filename* <return>	ファイルを読み込む
C-x C-w *filename* <return>	ファイル名を変更して保存する
C-g	指示途中のコマンド操作を取り消す
C-l	カーソルのある行を中央へ移動する
C-x u または C-_	直前の編集操作を取り消す (Undo)
M-x goto-line <return>	指定した行にジャンプする

カット＆ペースト

C-<Space>	カーソルの位置にマークをセットする
C-x C-x	カーソルの位置とマークの位置を入れ替える
C-w	マークの位置からカーソルの前までを記憶して消去する（カット）
M-w	マークの位置からカーソルの前までを記憶する（コピー）
C-y	記憶した文字列をカーソルの位置に挿入する（ペースト）
C-k	カーソルの位置から行末までを消去する

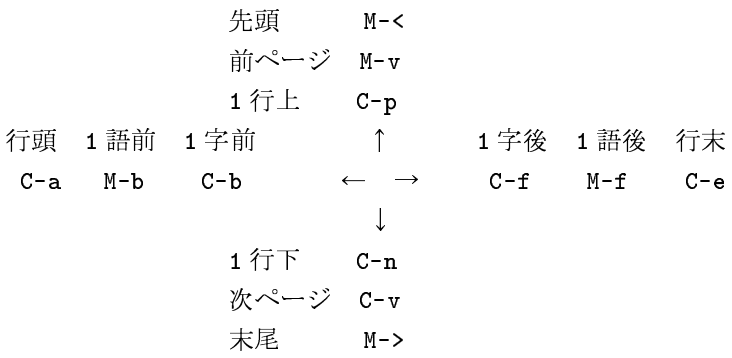
検索

C-s *String*	カーソル位置より下方向に向かって検索する
C-r *String*	カーソル位置より上方向に向かって検索する
C-s	次を検索する
C-r	前を検索する
C-g	検索を終了しカーソルを検索開始前位置に戻す

Wnn

C-\	Wnn を起動／終了する
<Space>	変換する、次候補を表示する（C-n でも可）
C-p	前候補を表示する
<return>	変換文字を確定する
C-o	文節を伸ばす
C-i	文節を縮める
C-f	右の文節へ移動する
C-b	左の文節へ移動する

カーソル操作



A.3.2 必要に応じて覚えるもの

起動時

mule *filename* <return>	Mule を起動して *filename* をバッファに読み込む
mule -q	.emacs を無視して Mule を起動する
mule -u *usr_name*	*usr_name* の人の.emacs の設定で Mule を起動する

ファイル操作

C-x i *filename* <return>	別ファイルをカーソルの位置に差し込む
C-x C-s	カレントバッファを保存する
C-x s	すべてのバッファを保存する
C-x k	カレントバッファを保存せずにクローズする
C-x C-v	カレントバッファにファイルを読み込む (カレントバッファの内容はクローズする)
C-x b	バッファを切替える
C-x C-b	バッファリストを表示する
C-x C-q	書き込みモードを変更する
M-x recover-file <return>	autosave された内容を読み込む

コードの変更

C-x C-k d　画面入出力コード変更
C-x C-k i　キーボードからの入力コード変更
C-x C-k f　ファイルの入出力コード変更

その他

全角、半角文字

M-x zenkaku-region	リージョン範囲を全角にする
M-x hankaku-region	リージョン範囲を半角にする

確定後の再変換

M-x henkan-region	リージョン範囲を変換する。
M-x gyaku-henkan-region	リージョン範囲を変換し直す。

文字の入れ換え

C-t	カーソルの位置の文字とその左の文字を入れ換える

繰り返し

C-u *n Command*	*Command* を *n* 回繰り返す
または M-*n Command*	例: C-u 5 C-_ アンドゥ5回

カーソル操作

M-a	文の先頭へ移動する
M-e	文の末尾へ移動する

カーソルの位置

M-x what-line **<return>**	今カーソルが何行目にあるか表示する
C-x l	全部の行数と現在のカーソル位置を表示する

改ページ

C-q C-l	改ページ文字^L を入力する

消去、カット &ペースト

M-x kill-rectangle **<return>**	マークセット位置からカーソル位置までのブロックを消去する
M-x clear-rectangle **<return>**	マークセット位置からカーソル位置までのブロックを空白に置換する
M-x yank-rectangle **<return>**	消去したブロックをカーソル位置に挿入する

置換

M-% *search-string* **<return>***change-string* **<return>**
検索文字列を置換文字列に確認しながら置換する
<Space>または y で置換を行なう
<Delete>または n で置換を行なわない
! で残り全部を確認せずに置換を行なう
^ で一つ前にもどる

M-x replace-string **<return>***search-string* **<return>***change-string* **<return>**
検索文字列を置換文字列にすべて置換する

画面分割

C-x 2	上下に二分割する
C-x 3	左右に二分割する (Emacs では C-x 5)
C-x o	カーソルを別ウインドウに移動する
C-x 1	カーソルのあるウインドウ以外のウインドウを隠す
C-x 0	カーソルのあるウインドウを隠す

ウインドウのリサイズ

C-x ^	カーソルのあるウインドウを縦方向に拡大する
C-x }	カーソルのあるウインドウを横方向に拡大する

バッファリスト

C-x C-b	バッファ一覧を表示する
?	バッファリスト簡易ヘルプを表示する
f	カーソルの行のバッファをウインドウに表示する
1	カーソルの行のバッファだけをウインドウに表示する
q	バッファ一覧を終了する

ヘルプ

C-h	ヘルプを呼び出す
C-h C-h C-h	ヘルプオプションとその説明を表示する
C-h k *Command*	*Command* の引数説明を表示する
C-h a *String*	*String* を含むコマンドの一覧を表示する
C-h b	現在のキー割当を表示する

オンラインマニュアル

C-h i	オンラインマニュアルを起動する
q	オンラインマニュアルを終了する
m	メニューを選択する
u	前のメニューに戻る
`<Space>`	続きを読む
`<Delete>`	前に戻る
n	次の項目に進む
p	前の項目に戻る
d	オンラインマニュアルの最初のメニューに戻る

Wnn

変換

M-h	(変換途中に) ひらがなにする
M-k	(変換途中に) カタカナにする
q	アルファベット入力モードにする
C-q	アルファベット入力モードをやめる
C-k または C-c	変換をキャンセルする (ひらがなに戻る)
	(C-f C-b で前後して訂正可能)

変換候補の一覧

M-s	変換候補一覧をエコー行に表示する
C-n	次の一覧部分を表示する
C-p	前の一覧部分を表示する

記号の入力

C-^	記号一覧のメニューをエコー行に表示する
または	0.JIS 入力
M-x special-symbol-input	1. 記号
	2. 英数字
	3. ひらがな
	4. カタカナ
	5. ギリシャ文字
	6. ロシア文字
	7. 罫線
	8. 部首入力
	0. 画数入力
	a. 第一水準
	b. 第二水準
	c. 補助漢字
C-n	次の一覧部分を表示する
C-p	前の一覧部分を表示する

単語登録

M-x toroku-region	リージョン (矩形) 指定した単語を登録する
M-x edit-dict-item	単語を登録した辞書を編集する

その他

x?	小さい文字を出す (例:「xa」で「ぁ」)
z?	特殊記号を出す (例:「z(」で「【」)

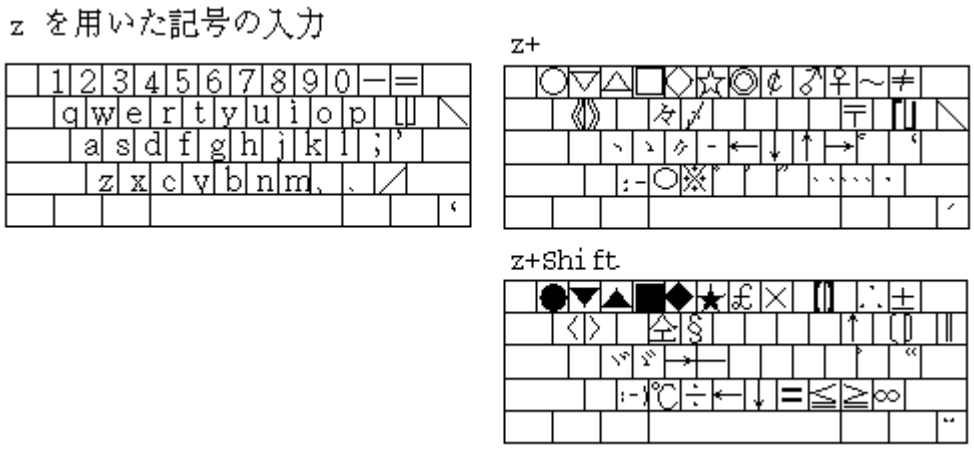


図 A.1 z キーとの組合せによる記号

MHE

読む

M-x mh-rmail <return>	MHE メイルリーダを起動する
q	MHE メイルリーダを終了する
.	メイルを読む
<Space>	続きの部分を読む
<Delete>	前の部分を読む

書く

M-x mh-smail <return>	メイルを書く
C-c C-c	メイルを送信する
C-c C-q	メイルを送信するのをやめる

返事

a	メイルに返事を書く
C-c C-y	メイルの内容を引用する
C-c C-c	返事を送信する
C-c C-q	返事を送信するのをやめる

整理

o *foldername*	メールを~/Mail 以下のフォルダに振り分ける

メールボックス

M-f *folder_name*	~/Mail 以下のフォルダのメールを読む
M-r	現在のフォルダを読み直す。古いメールを読み返す時に便利。

GNUS

M-x gnus <return>	GNUS を起動する
q	GNUS を終了する

ニュースグループ選択画面

<Space>	カーソル位置のニュースグループを読む
c	すべて既読にする
u	次の起動から表示しないようにする
L	すべてのニュースグループを表示する

記事画面

<Space>	カーソル位置のニュースを読む
<Space>	記事の続きを読む
q	ニュースグループ選択画面に戻る
<Delete>	記事の前の部分を読む
d	記事に既読マークを付ける
f	表示中の記事に対してフォローする
F	表示中の記事に対して引用付きでフォローする
a	投稿のための原稿を編集する
C-c C-c	（フォロー、投稿記事編集画面で）記事を投稿する
o	記事を保存する

Directory モード

M-x dired <return>	Dired モードを起動する
q	Dired モードを終了する
f	カーソル位置のファイルを読み込む。ディレクトリなら移動
v	カーソル位置のファイルを見る。元に戻るのは C-c
^	親ディレクトリに移動
~	バックアップファイル filename に削除マークを付ける
d	カーソル位置のファイルに削除マークを付ける
u	カーソル位置のファイルのマークを取り消す
x	マークファイル（削除など）を実行
C	カーソル位置のファイルのコピー
D	カーソル位置のファイルの削除
R	カーソル位置のファイルのリネーム
M	カーソル位置のファイルの chmod

C モード

M-x c-mode	C モードにする
M- C-\ (または M-x indent-region)	リージョンの範囲をインデントする

コンパイル

M-x compile <return>	コンパイラを起動する

.emacs の設定例

```
(setq enable-double-n-syntax t)
```

「nn」で「ん」と変換するようにします。

```
(load "/NF/local/general/lib/mule/19.28/lisp/its/hira.el")
(its-defrule "string" "string2")
```

但し、emacs の場合は

```
(defrule "string" "string2")
```

string をローマ字入力すると string2 になるようにします。
例
(load ”/NF/local/general/lib/mule/19.28/lisp/its/hira.el”)
(its-defrule ”dhi” ”でぃ”)
(its-defrule ”thi” ”てぃ”)

```
(setq-default case-fold-search nil)
```

case-fold-search という検索時に大文字小文字を区別するかどうかのデフォルト値を設定します。この場合は全てのバッファにおいて区別します。

```
(global-set-key "\C-x@" 'compile)
```

C-x @のキー操作に対して compile のコマンド操作を割り当てます。因みに M- C-a なら"/e/C-a"と表します。

```
(autoload 'gnus  "gnus" "Read Network News" t)
```

起動時に gnus 関数を自動的に読み込みます。

```
(setq kill-whole-line t)
```

ただし emacs の場合は

```
(defun kill-line-twice (&optional numlines)
  "Acts like normal kill except kills entire line if at beginning"
  (interactive "p")
  (cond ((or (= (current-column) 0)
     (> numlines 1))
  (kill-line numlines))
(t (kill-line))))
(global-set-key "\C-k" 'kill-line-twice)
```

通常、一行削除は行頭で C-k を 2 回行なう必要がありますがこれで一回で済むようになります。

A.4　京都産業大学 FAQ(抄)

A.4.1　目次

A.4.2　はじめに

Q.　FAQってなんですか？

A.　何度も繰り返される質問をまとめたＱ＆Ａ集です。

　Frequentry asked question の略です。何度も同じ質問が出ると答える方も疲れるし、質問する方も気が引けてくるので、まずこれを見ることでそれを解決しようという意図があります。ここでは学内にある計算機の環境に対するさまざまな質問に答えています。

Q.　この FAQ リスト（の最新版）はどうしたら得られますか？

A.　次の内からお選び下さい。

・計算機センター相談窓口（計算機運用補助員待機場所）まで取りに行く。
・World Wide Web の各種ドキュメントにあるＦＡＱを見る。
・sandai.question に定期的に投稿されるものを見る。

Q.　補助員、または MiCS 補助員ってなんですか。

A.　アルバイトの学生によるコンピュータ環境のお助け部隊です。

　皆さんは学内のコンピュータ施設を使っていて、何か困った事があったことはないでしょうか。プリンタから印刷されてこない、コンピュータが使用中に止まってしまった、このアプリケーションの使い方が分からない、等々。そんなときにはぜひ補助員を呼んでください。
　補助員とは正確には「計算機運用補助員」という名前で、計算機センターで学内のコンピュータのトラブルに対応するために待機している、その方面の知識を持った学生達のアルバイトです。「計算機運用補助員」ではイメージが固いので MiCS 補助員というニックネームを付けました。以下 MiCS 補助員と呼びます。
　MiCS 補助員は学内で授業が行われている間中、複数の人間が勤務しており、少なくとも一人は計算機センターが管理しているＣ１・１１・Ｃ３・２１・３１・５１・５２情報処理教室、それぞれの部屋を何かトラブルはないかと巡回を続けています。また少なくとも一人は計算機センターで常時待機しており、いつでもコンピュータ施設のトラブルに対する電話を受け付けています。あなたがもしどこかの情報処理教室でトラブルに遭遇した時、その部屋にタイミング良く補助員がいれば解決しますし、いなくても、各情報処理教室に設置されている内線電話を使って補助員に連絡すれば電話で、もしくはかけつけて問題を解決してもらえるというわけです。
　MiCS 補助員を呼ぶには先ほども書いたように各情報処理教室に設置されている内線電話を使って「2578」をプッシュしてください。すると「MiCS 相談窓口」というところにつながり、補助員がトラブルの状況を聞いてきます。もしそれが簡単な対処法で解決するような問題ならば電話で補助員が解決方法の指示を出します。そうでなければ補助員が何らかの方法で解決しますので指示を聞いて下さい。
　また補助員は年度の初めに新たにスタッフの募集をしております。コンピュータをやってみようという意欲のある方、応募をお待ちしています！
　－ MiCS 補助員－
業務内容：計算機センターが管理しているコンピュータ施設のトラブル全般の解決
業務期間：学内で通常講義のある期間全て
特　　典：計算機に対する知識が得られる
連絡方法：各情報処理教室に設置してあるコードレスホンで 2578 をプッシュしてください

A.4.3　UNIX 編

各種のコマンドに関すること

Q.　FAQ に載っていないトラブルなんですが、どうすれば良いですか？

A. 補助員や良く知っている人に聞きましょう。

計算機について分からないことがあれば、各情報処理教室にあるコードレス電話で 2578 をダイヤルして下さい。MiCS 補助員というナイスガイ&ナイスギャル達がお答えします。また、お急ぎでない場合や、専門的な質問は京都産業大学ローカルニュースグループの sandai.question に投稿して下さい。誰か知っておられる方からフォローが入るでしょう。

Q. 21 情報処理教室の DEC3300(csosfxx) でフロッピーを使うには?

A. fdio コマンドを使用します。

メディアとしては 2ED, 2HD(1.44), 2DD(720) が利用できます。(2ED というのは最近東芝などが作っている 2.88MB 容量のものです。ワークステーションは最近良く使っていたりします。)
通常は 2HD と 2DD の DOS フォーマットを扱います。fdio コマンドを利用して下さい。

```
csosf03(81)% fdio

DOS Floppy Read/Write Utility  Version 1.2
fdio>
```

というところですかね。どのようなサブコマンドが利用できるかは help サブコマンドで調べることが出来ます。

```
fdio> help
 binary      dir          cd          read
 text        type         mkdir(md)   write
 bye         pwd          rmdir(rd)   ren
 quit        help(?)      find        del
fdio> help dir
 dir -- ディレクトリの一覧を表示する.
 使い方: dir [-awl] [dos-directory or dos-file ... ]
```

終了するには quit コマンドを使用します。

```
fdio> quit
```

UNIX ファイルをそのままフロッピーに詰めたい場合は tar コマンドを使うのがいいのでしょうね。デバイス名は /dev/rz5c です。

Q. コマンドの実行結果をプリントアウトしたいのですが?

A. いくつか方法があります。

まず script コマンドを利用する方法。例えば、

```
cc2000(121)% script filename
```

とすると exit を実行するまでのことが filename と言う file に書き込まれます。これは画面に出力される事がそのまま書き込まれます。
もう一つ、リダイレクトを利用する方法。

```
cc2000(127)% a.out > filename
```

とすると実行結果が filename と言う file に出力されます。
他には、既に実行されて画面に結果が表示されているならば、それを Cut&Paste で

```
cat > filename
```

を使って filename のファイルにする事も出来ます。
後は lpr コマンドでプリンターに出力して下さい。

Q. grep の使い方、正規表現が良く分かりません。

A. grep はある文字列を含む行の一覧を出力するコマンドです。たとえば,

```
% grep hello sample.txt
```

または

```
% cat sample.txt | grep hellow
```

とすれば，sample.txt ファイル中の”hello”という文字列が出現する行が全部画面（正確には標準出力）に出てきます.
文字列にはワイルドカード (「＊」や「？」) も使用可能です。
サポートしている正規表現は

^：行頭に Match
$：行末に Match
.：CR（改行 Code）以外の任意の 1 文字に Match、2 Byte Code も Document の Script に応じ 1 文字として認識

です。このように使います。

例：
```
1: sample document 12345
2: 12345
3: 123456789 this is a test
4: abcdefg9
```

grep '^123' で、	2: 3: がヒット	(先頭から 123 があるもの)
grep '123'	1: 2: 3:	(123 があるもの)
grep '9$'	4:	(9 で終るもの)
grep '9'	3: 4:	(9 を含むもの)
grep 'a...e'	1: 4:	(a に続いてどんな文字でもいいから 3 文字あって、e があるもの)
grep 'ae'	全くヒットしない	(ae があるもの)
grep 'a*e'	全くヒットしない	(ae,aae,aaae,... を含むもの)
grep 'a.*e'	1: 3: 4:	(a に続いてどんな文字でもいいから 何文字かあって（ゼロ文字でも良い） e があるもの)

Q. 実行中のプロセスの終了

A. ps コマンドと kill コマンドを使用します。

とりあえず ps コマンドを実行します。すると以下のようなものがでてきます。

```
cc2000(82)% ps
  PID TT       S  TIME COMMAND
25830 pts/10   S  0:00 -tcsh
25138 pts/12   O  0:00 ps
29506 pts/12   S  0:01 -tcsh
25931 pts/13   S  0:00 -tcsh
26171 pts/13   S  0:41 mule
```

これで目的とする物がでてこないなら次のようにします。但しこれは cc2000 の場合ですのでそれ以外の機種は下記の表に対応するオプションを指定して下さい。

ホスト名	ＯＳ	オプション	備考
cc2000	Solaris2	-axu	/usr/ucb/ps の場合 /bin/ps の場合は -ef
csosf??	OSF/1	-ef	
ccns0??	BSD4.2	-axu	
--	標準的な SVR4	-ef	
--	標準的な BSD	-axu	SunOS なら -xu で良い

```
cc2000(83)% ps -aux | grep ozaki
ozaki    24908  0.2  0.1  972  836 pts/12   O 14:06:57  0:00 ps -aux
ozaki    24909  0.1  0.1  708  484 pts/12   S 14:06:57  0:00 grep ozaki
ozaki    29506  0.1  0.2 1080  996 pts/12   S 09:11:17  0:01 -tcsh
ozaki    25830  0.0  0.2 1084  948 pts/10   S 08:30:45  0:00 -tcsh
ozaki    25931  0.0  0.2 1080  940 pts/13   S 08:31:09  0:00 -tcsh
ozaki    26171  0.0  0.5 4472 3280 pts/13   S 08:31:19  0:41 mule
```

```
ozaki    26196  0.0  0.2 1080  952 pts/14   S 08:31:52  0:01 -tcsh
```

このようにすると全てのプロセスから ozaki さんのプロセスを表示します。パイプより前の部分でシステム中の全プロセスをリストして、パイプより後ろの部分で前半の結果から自分のユーザ名（例では ozaki）の文字列を含む行だけ抜き出して表示させている訳です。これで自分のプロセスが表示されます。ですから上記の ozaki を自分のユーザ名に置き換えると自分のプロセスを表示してくれます。

そして一番左の数字がプロセス番号ですので、殺したい（終了させたい）プロセス番号（PID）を指定した kill コマンドを実行します。

```
cc2000(83)% kill 26171
```

これで、もし止まらない場合はもう少し強力にして止めるオプションをつけます。

```
cc2000(83)% kill -1 26171
```

単なる kill （実は kill -15 と同じ）, kill -1 , kill -2 , kill -9 の順に強力になりますので、順に試してください。

Q. LINKって何？

A. ファイルを扱う方法の一つです。

LINK にはシンボリックリンクとハードリンクという 2 種類が存在します。
ファイルというのはシステム的に見ると、2 つに分割して考えることが出来ます。
・現実にディスクに書き込んである内容そのもの。
・その開始位置を指し示しているポインタ。
普段私たちが目にしているファイル名は 2 つめのポインタにあたります。

シンボリックリンクとはファイル名を指すファイル名です。下記のように file1 を指す file2 のようなものです。file2 を参照すると、file1 を見にいき、それは file の内容を指しているので無事 file の内容を参照できる訳です。

```
File の内容
↑
file1 ← file2
```

ハードリンクとはファイルの内容を指し示すもう一つの名前をつけてやる事をいいます。

```
 File の本体
 ↑ ↑
file1 file2
```

通常使うのはシンボリックリンクの方です。リンクの利点は主に二つあります。一つはディレクトリ構造の離れた所にあるファイルを指定してアクセスし易くする事。もう一つはコピーと違ってファイルの実態は一つなのでディスク容量を節約することが出来ることです。ハードリンクは普通ディレクトリに対して使うことが多いです。例えば、プログラムのコンパイル等でライブラリの指定をリンクで行っていれば、それを変える事でライブラリをを簡単に切り換える事もできます。

シンボリックリンクの設定方法

```
% ln -s file1 file2
```

ここで file1: 元のオリジナルファイル名
file2: 元ファイル名 (file1) を指し示すようにしたいファイル名

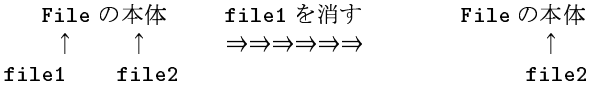
この時に file1 を消すと file2 は指し示すファイルが無くなるので名前だけ残って使えないファイルになってしまいます。

```
File の本体        file1 を消す       File の本体
   ↑              ⇒⇒⇒⇒⇒⇒
file1 ← file2                        ??? ← file2
```

それに対してハードリンクは

```
% ln file1 file2
```

と実行します。この場合、file1 と file2 の立場は同等で、実体は同じで名前を二つ持っている状態になります。この時は file1、file2 のどちらかを消してももう一つの側でファイルにアクセスすることが出来ます。



Q. テキストのファイルがぐちゃぐちゃで読めません

A. ファイルの文字コードを変換します。

漢字やひらがな等のアルファベット 2 文字分の大きさのある文字（全角文字）は規格によりその文字のコード（文字を記号とみなした通し番号）が違うので違う規格で読むと全然違う文字が表示されてしまいます。主に本学の環境では EUC コードを用いていますがメールやニュースでは JIS コードも用いられています。cc 環境では nkf という Network Kanji code conversion Filter があります nkf コマンドを使ったコード変換は次のようにして行ないます。

1. ファイルを一旦別の名前にして、
2. nkf で 目的のコードに変換し、それを 元のファイル名に書き出し、
3. 別の名前にしておいたファイルを消去する

というところです。以下に具体的な手続きを。

1. cc2000(20)% mv filename1 filename2

2. 目的に合わせて次の何れかを選んで下さい。
それぞれ、j(JIS) e(EUC) s(ShiftJIS) に変換します。

```
cc2000(21)% nkf -j filename2 > filename1
cc2000(21)% nkf -e filename2 > filename1
cc2000(21)% nkf -s filename2 > filename1
```

3. cc2000(22)% rm filename2

Mule に関すること

Q. mule でのかな漢字変換で「nn」で「ん」と変換するにはどうすれば良いのですか？

A. .emacs に一行書き加えます。

まず

```
cc2000(01)% mule .emacs
```

として下さい。これで.emacs という mule の環境設定ファイルを読み込み mule が起動します。今まで何も変更していない人は

```
;;
;;       Emacs common settings.
;;

(load "/NF/home/common/settings/_emacs.load")
```

このような内容になっていると思います。
ここで、(load... の行の次の行に

```
(setq enable-double-n-syntax t)
```

を書いて下さい。

```
;;
;;       Emacs common settings.
;;

(load "/NF/home/common/settings/_emacs.load")
(setq enable-double-n-syntax t)
```

このようになりますね。これを保存して終了します。すると次に mule を起動した時には「nn」で「ん」が出るはずです。
なお、emacs ならば上記の代わりに

```
(defrule "nn"  "ん")
```

と書いて下さい。

Q. Mule で単語登録したものの一覧は得られますか？

A. dtoa コマンドを使います。

```
cc2000(80)% dtoa ~/Wnn/private
```
とすれば一覧を得られます。

また、大がかりに辞書を編集したい時は

```
% dtoa ~/Wnn/private > private.txt
% private.txt を mule などで編集する。
% atod ~/Wnn/private < private.txt
```
という手順を用いることもできます。ただし、この方法は間違うと辞書を壊してしまうかもしれませんので注意して下さい。

メイルに関すること

Q. メイルを相手が読んだかどうかを確認したいのですがどうすればよいですか？

A. 自動的に相手が読んだら確認できるようなシステムはありません。

もし必要ならメールに読んだら返事をしてくれるよう書いておけば良いでしょう。
また、NeXT メイルなら受領証という機能がありますのでそれで確認することができます。

Q. シグネチャを付けるにはどうすれば良いのでしょうか？

A. あらかじめファイルを作っておいてそれを挿入します。

前もってシグネチャをファイルに書いて作っておきます。そして必要に応じてその内容を取り込みます。そうすると毎回タイプする必要がなくなりますね。

mule を使ってメールを書いている人を対象に説明します。例えば sig というファイル名で 1 ～ 4 行程度のシグネチャを作っておきます。そしてメールを書き終わった時に C-x i と（コントロールキーを押しながら x キーを押して、コントロールキーを離して i キーを押す）すると、下に Insert file: ~/ の様なメッセージが出ると思います。出たら、そのメッセージに続けて sig とタイプします。

```
Insert file: ~/sig
```
となりましたか？なったらリターンキーを押して下さい。無事にシグネチャがカーソルのあった位置へ書き込まれたと思います。なお、余り長いシグネチャはマナーに反しますので 4 行までにしましょう。

Q. 相手のメイルアドレスが判らないのですが、調べる方法はありますか？

A. 基本的に調べる方法はありません。

ただし cc 環境ならば whois コマンドを利用して調べる方法があります。

Q. 特定の人から来たメイルだけ別のフォルダに入れる方法はありませんか？

A. 読み終えたメールを振り分けることが出来ます。

mule でメールを読んでいる時に o を押すと

```
Destination folder? +
```
と聞いてきます。ここで、そのメールを保存したいフォルダを入力して<return>を押すと振り分けてくれます。指定したディレクトリが存在しない場合は

```
Folder +hogehoge does not exist. Create it? (y or n)
```
と聞いてきますので y と答えて下さい。

まとめて行ないたい場合は、コマンドで行ないます。例として、suzu3 から来たメイルをフォルダ suzu3 に入れる場合

```
cc2000(81)% refile ‘pick -from suzu3‘ +suzu3
```
（‘ と ’ を間違えないように！）

これで ~/Mail/inbox のメールのうち、suzu3 から来たメールが全て ~/Mail/suzu3 に移されます。~/Mail/suzu3 が存在しなければ

```
Create folder "/NF/home/syokuin0/ozaki/Mail/suzu3"?
```

と聞いてきますので y と答えれば OK です。

Q. ニュースやメールの返事を書く時に引用符が付けられなくなっちゃいました。

A. supercite がうまく動いていないのです。

具体的にはメールを読んだ後、それを引用してメールを書こうとして C-c C-y とすると

```
Wrong type argument: listp, " . "
```

と表示される場合です。

ホームディレクトリ以下に、.scrc.el というファイルがあると思いますが、これが supersite という引用符号を付けるプログラムが使う設定ファイルです。

今まで定義した引用記号の設定が失われてもいいのであれば、このファイルを削除してやればうまく行くようになると思います。

設定が失われるのが嫌であれば、このファイルの中を覗いて、怪しげな部分を手で修正する (mule で開いて修正、保存する) というのが良いと思います。

Q. フォルダ内のメールの番号を日付順にするにはどうしたらいいのですか?

A. mule + mh-e の機能を使います。

mule or emacs + mh-e を使ってるのでしたら、そのフォルダを visit している状態 (メールの subject の一覧がでている状態) で

```
M-x mh-sort-folder <return>
```

をすると日付順に番号をつけ直してくれると思います。

mule や emacs で何かする機能が無いかどうかを調べるには「並びかえる」は「sort」なので C-h a sort <return>とすると 「sort」と言う文字列を含んだ関数の一覧が表示されます。その一覧の中で「mh」 や 「folder」 と言う文字列を含むものを探すと「mh-sort-folder」を見つけることが出来ます。

ニュースに関すること

Q. ニュースグループのソートの仕方 (GNUS)

A. 直接順番を入れ換えるか、ファイルをソートします。

ニュースグループ選択画面のニュースグループを編集して行を入れ変えるとその順番になります。

```
          :
    9: sandai.comp.announce
    4: sandai.comp
    1: sandai.general
   11: sandai.rec
          :
```

例えばこの様なニュースグループの並びになっていて、 sandai.rec を sandai.comp の上に持ってきたい、というような場合は、まずカーソルを sandai.rec の行の先頭（１カラム目）に移動させます。ここで C-k（コントロールキーを押しながら k）を二回実行すると sandai.rec の行が消えます。消えたら今度は sandai.comp の行の先頭にカーソルを移動させて C-y を実行します。

```
          :
    9: sandai.comp.announce
   11: sandai.rec
    4: sandai.comp
    1: sandai.general
          :
```

こうなりましたね。これで今後もこの順序でニュースグループが表示されます。
他にはコマンドラインからソートする方法もあります。

```
sort <元ファイル名> -o <出力ファイル名>
```

でアルファベット順にソートできます。そしてニュースグループが書いてあるファイルは~/.newsrc-cc2000 なので、自分のホームディレクトリで

```
cc2000(20)% sort .newsrc-cc2000 -o .newsrc-cc2000
```

というコマンドを実行してやればニュースグループをソートすることが出来ます。

!!! やってはいけないこと !!!

```
cc2000(20)% sort .newsrc-cc2000 > .newsrc-cc2000
```

こういう書き方は一見出来そうな気がするのですが、> .newsrc-cc2000 の部分で書き込み場所を先に確保してしまうので .newsrc-cc2000 が初期化され無くなってしまうのです。無くなった物をソートしても何も残るはずは無く、.newsrc-cc2000 というファイルは中身なしのファイルになってしまいます。

Q. ニュースを読んでいたら、ニュースグループの頭に「*」が付いてしまいました。偶然の産物なので、消し方を知りません。どうすれば消えるのでしょうか。

A. gnus で読んでいる時に、u を押してしまったのではないでしょうか？

記事を読んでいる時に u を押すと、mule の上の画面のその NG の記事一覧の一番左にあるマークが - になって、既読で記事を消去するのを一時保留します。（普通は、ここが D になって q で抜けると、D マークがついた記事は既読として消去されてしまいますよね。）それで、NG 一覧の画面では、そういうファイルが NG の中にあると、その横に*をつけて、以上の処理をした記事がありますよと、知らせます。だから、もういちどその記事を読んだことにしてやるために記事の場所で'd' を押すと既読マークがつくので、すべて解決となるはずです。

Q. 時々相手が文頭や文末に自分のことを「尾崎@計算機センターです」などというように書いていますが、これはどういう意味ですか？

A. これは「尾崎」さんが「計算機センター」の人ですという意味です。

要は「@」マークが区切りとなっていて、個人@所属を表します。こういう書き方の元はメイルアドレスからきています。メイルアドレスは honyarara@cc.kyoto-su.ac.jp という風になっていますね。これは honyarara さんが jp(日本) の ac(大学・研究機関) の kyoto-su(京都産業大学) の cc(ホストマシン名) の人ですという意味です。きちんと個人@所属になっていますね。
尚、「個人%コメント等@所属」 という表記もあるようです。これにも由来はあるのですがここではふれません。参考程度に覚えておいてもいいでしょう。

Q. シグネチャを付けるにはどうすれば良いのでしょうか？

A. あらかじめファイルを作っておいてそれを挿入します。

前もってシグネチャをファイルに書いて作っておきます。そして必要に応じてその内容を取り込みます。そうすると毎回タイプする必要がなくなりますね。
mule を使ってメールを書いている人を対象に説明します。例えば sig というファイル名で１～４行程度のシグネチャを作っておきます。そしてメールを書き終わった時に C-x i と（コントロールキーを押しながら x キーを押して、コントロールキーを離して i キーを押す）すると、下に Insert file: ~/ の様なメッセージが出ると思います。出たら、そのメッセージに続けて sig とタイプします。

```
Insert file: ~/sig
```

となりましたか？なったらリターンキーを押して下さい。無事にシグネチャがカーソルのあった位置へ書き込まれたと思います。なお、余り長いシグネチャはマナーに反しますので４行までにしましょう。

Q. ニュースに記事を投稿したのですが、うまく投稿できたかどうかを確認するには、どうすれば良いですか？

A. その記事を自分で読みましょう。

GNUS を使っているならニュースグループ選択画面で g キーを押す、または GNUS を一度終了して、もう一度起動すると新しく投稿された記事を読み込むのでそれで確認して下さい。だたし投稿後しばらくしないと記事が処理されません（最大１５分）ので注意して下さい。

Q. いろんなニュースグループがありますが、それぞれどんなものなのですか？

A. 以下に一覧を挙げておきます。

京都産業大学に限ったニュースグループ

sandai	京都産業大学からのみ読み書き出来るニュースグループ。まずはここで慣れてから fj.* 等を読み書きすると良いでしょう。

京都産業大学以外の一般的なニュースグループ

fj	主に日本語によって論議をするためのもの。特定の組織や団体が管理・運営をしている訳ではなく、利用者全員の合意に基づいて運営されている。現在のところ営利目的の記事の投稿はできない。
tnn	IIJ が主催する営利目的の記事を投稿できるニュースグループ
jp	日本の IP 接続組織用のニュースグループ
comp	USENET のうちコンピュータに関する話題用
news	USENET のうち NetNews に関する話題用
rec	USENET のうち趣味に関する話題用
sci	USENET のうち科学に関する話題用
soc	USENET のうち社会問題に関する話題用
talk	USENET のうち各種の論議用
misc	上記以外の USENET の話題用
alt	USENET ニュースグループ群に代わる、各種の話題用。非常に緩いルールしかなく、かなり自由にニュースグループを作成して利用できる。
bionet	医学・生物学の話題用
biz	ビジネスの話題用
gnu	FSF(フリーソフトウェアファウンデーション) やその製品 (GNU プロダクト) に関する話題用
k12	アメリカにおけるネットワークを使った教育の実験のためのニュースグループ
vmsnet	DEC VAX/VMS OS に関する話題用

Q. クロスポストってなんですか?

A. 複数の NG に同じ記事を投稿する時に使用する機能です。

クロスポストとはニュースの記事をポストする時、ポスト先のニュースグループを複数指定（通常２つか３つ）して、同じ記事を各ニュースグループに投稿するものです。各ニュースグループに個別に投稿するのに比べて、記事の実体は一つで済みますので、ネットワーク資源の節約になります。
方法は、通常のポスト時に Newsgroup の指定行を

```
Newsgroups: sandai.test, sandai.junk
```

のようにカンマで区切って複数のニュースグループを書きます。この状態で投稿すると指定した各ニュースグループに投稿します。この状態の記事にフォローを入れた場合、Newsgroups の指定はそのまま受け継がれるので全てのニュースグループに同じフォローを入れる事になります。(このフォロー先を一意に決めたい時はフォローアップを使います。)
通常クロスポストを使うのは何箇所かのニュースグループで同時に議論したいときと、そのニュースグループで議論を続けるのにふさわしくない記事に対して、他のニュースグループにふる時につかいます。前者の場合は議題が幾つかのニュースグループにまたがっていて、どうしてもそれら全てのニュースグループの読者の意見を交えて議論したい、という場合です。通常のポストの際に Newsgroups に上記のように二つ以上のニュースグループを指定して下さい。
後者の場合は、フォローして、そのニュースグループと新しいニュースグループとのクロスポストにして、フォローアップを新しいニュースグループにします。例えば sandai.question のある記事に対して sandai.junk の方が適切だ、と思ったらフォローする際に次のようにします。まず F キーを押して引用符を決めるとこの様になりますね。

```
－－－ここから－－－
In-reply-to: ozaki@cc.kyoto-su.ac.jp's message of 26 Jan 1995 12:39:48 +0900
Newsgroups: sandai.question
Subject: Re: .emacs
Distribution: local
References: <TANIMA.95Jan26114834@csosf20.kyoto-su.ac.jp>
        <OZAKI.95Jan261239945@cc2000.kyoto-su.ac.jp>
--text follows this line--
（引用した本文）
－－－ここまで－－－
```

通常、「–text follows this line–」より上の部分は書き換えないのですが、Newsgrops: 行の変更と、Followup-To: 行の追加を行ないます。

Newsgroups: sandai.question, sandai.junk
Followup-To: sandai.junk

この様に修正、追加します。次のようになりましたね。

```
－－－ここから－－－
In-reply-to: ozaki@cc.kyoto-su.ac.jp's message of 26 Jan 1995 12:39:48 +0900
Newsgroups: sandai.question, sandai.junk
Followup-To: sandai.junk
Subject: Re: .emacs
Distribution: local
References: <TANIMA.95Jan26114834@csosf20.kyoto-su.ac.jp>
        <OZAKI.95Jan26123945@cc2000.kyoto-su.ac.jp>
--text follows this line--
（引用した本文）
－－－ここまで－－－
```

二行以外は変更ありません。これでいつものようにポストするとその記事は sandai.question と sandai.junk に投稿され、それぞれの記事を見た人がそれにフォローしようとするとニュースグループに sandai.junk が選択されます。

そして、一言、「sandai.junk に振ります」と書き添えて下さい。こうすると sandai.question の読者は sandai.junk に議論の場が移るんだ、ということが分かりますし、sandai.junk の読者は sandai.question から移ってきた議論だ、ということが分かります。

Q. Followup-To: ってなんですか？

A. 記事に対してのフォロー先のニュースグループを決めるものです。

例えば sandai.test にある記事に

```
Followup-To: sandai.junk
```

となっていたらその記事のフォローは sandai.junk に投稿されます。

X に関すること

Q. リモートログインした機械でアプリケーションを立ち上げようとすると Can't open displayって出ます。

A. 貴方が使っている機械の環境を変えなければいけません。

これはそのリモートログインしている機械が貴方の使っている機械に表示しようとしたけれども出来ませんでした、という事です。Xウインドウはネットワークを介して繋がっている機械に画面を表示する為に作られたウインドウシステムで、その為、何処に表示するのか、何処から自分の画面への表示を許すのか、を指定してやる必要があります。Can't open display とは表示しようとしている機械に表示出来ませんでした、ということです。他からの画面の書き込みを許すには xhost というコマンドを使います。

```
% xhost 機械名
```

とするとその機械からの表示を許可します。何処に表示するかは環境変数の DISPLAY で指定するのでログインした先のコマンドラインで

```
% setenv DISPLAY 自分の使っている端末の名前:0.0
```

と入力します。これで、目の前の端末が、ログイン先の機械の画面を表示出来る様になります。

例

```
csosf11(81)% xhost cc2000
cc2000 being added to access control list
csosf11(82)% rlogin cc2000
```

```
Last login: Thu Sep 22 11:00:56 from csosf41
tcsh: using dumb terminal settings.
Sun Microsystems Inc.   SunOS 5.3       Generic September 1993

cc2000(81)% setenv DISPLAY csosf11:0.0
```

（下線部が入力部分）

setenv の 0:0 は 0 番目のディスプレイの 0 番目の画面という意味で、この意味のよく解らない人はおまじないとして打っていれば O.K.

それ以外のこと

Q. パーミッションって何ですか?

A. ファイルに対するアクセス権の事です。

パーミッションとはファイルに与えられるアクセス権で、オーナー、グループ、他人の三種類の人に対してリード権、ライト権、実行権をあるか無いかで決めます。詳しくはコンピュータガイド インターネット編 「UNIX もっともっと ファイルのアクセス権」を見て下さい。

Q. ディレクトリを他の人からも見られるようにしたいんですが

A. ディレクトリのアクセス権を変更します。

そのディレクトリに対するアクセス権を変えます。読めるようにするにはリード権と実行権を与えます。さらに書き込めるようにするにはライト権を与えます。あるグループに対して解放するなら所属グループを変更します。

この文章を読んで意味が分からない人はセキュリティ上、危険ですので少し勉強した方が良いと思われます。例えば自分宛のメールを覗かれたり大切なデータをいじられたりする危険性があります。自信のない人は分かっている人に見て貰って下さい。

Q. キーボードを打っても文字が化ける、あるいは何も表示されずまともに動かないのですが

A. 状況として３通り考えられます。

一つは何かバイナリデータなどの普通表示できないデータを画面に表示してしまった場合、端末設定が変わってしまい、以後の文字がすべて化けてしまう場合。そしてカナキーや CAPS キーなどの特殊なキーを押してしまった場合。もう一つは C-s(コントロールキーを押しながら s キーを押す）の場合です。順に対処法を。

一番めの場合
- 端末状態をリセットする
- 端末エミュレータを再起動してやる

二番目の場合
カナキーや CAPS キーを解除する。(もう一度押す)

三番目の場合
C-q（コントロールキーを押しながら q キーを押す）をする。

Q. 印刷したいのですが、どのプリンタを指定したら良いのでしょうか?

A. ccinfo コマンドで参照できます。

% ccinfo<return>で ccinfo コマンドを起動して、「cc 環境の設備について」の「プリンタの配置について」を見て下さい。
C3 情報処理教室の NeXT についてはモノクロの印字は綺麗で速いレーザープリンターの TAKE や UME で、どうしてもカラーでないといけないものは遅いけどカラー印刷のできる MATSU で行って下さい。プリンターの場所は MATSU と TAKE が C3、UME が C4 情報処理教室です。

Q. 家のパソコンとデータをやりとりしたいのですが

A. コンピュータガイド -インターネット編- FTP の節をお読み下さい。

Q. 21 情報処理教室の DEC3300 でセッション休止から復帰できません

A. Lock キーを調べて下さい。

login panel から login した後にルートメニューのアクセサリからセッション休止を選ぶと画面を真っ暗にして休止状態になりますが、このとき、Lock キー（CapsLock キー）を押して Lock 状態（キーボード右上の Lock ランプが点灯）にしていても、休止状態になったと同時にキーボードの Lock ランプも消えてしまいます。（しかし Lock は効いたまま）

休止状態から復帰する際にはパスワードの入力を必要としますが、このとき、Lock ランプは消えていますが Lock 状態のままです。小文字モードと思ってパスワードを入力してもアルファベットは大文字のままで入力されますから、復帰に失敗します。

Lock キーはちゃんと効いていますので、Lock キーを一回押してもう一度パスワードを入力してみましょう。復帰に失敗した原因がこれであったならば、復帰に成功します。

A.4.4 Mac 編

Q. マックにリセットスイッチはついていないのですか？

A. キーボードによるリセットがあります。

ctrl とアップルキー (花文字の奴) を押しながら電源を入れる時に押す、キーボードの上にある四角いボタンを押します。（一部機種では使えません）

Q. Mac でフロッピーが取り出せない。

A. フロッピーに保存された書類からアプリケーションを立ちあげていませんか?

作業中に書類を保存して、一旦アプリケーションを終了させてください。取り出せるようになります。もし、フロッピーが入ったままハングしたときは、アップル+シフトキー+数字の 1 を試してください。それで駄目なら、マウスのボタンを押したままリブートしてください。大概はこれで吐き出されるはずです。

ちなみに、この 2 つは Macintosh での、フロッピーディスクの強制 eject 方法です。もし、これでも駄目なら、もう一度、アップル+シフトキー+数字の 1 を押してください。

こうしても無理なら MiCS 相談室 (内線 2578) までお電話下さい。補助員が対処しに行きます。

Q. Mac でフロッピーディスクがロックされている。フロッピーディスクに保存できない。

A. フロッピーディスクが書き込み禁止状態になっていませんか？

フロッピーディスク (差し込む方向に持って、そのまま手を返してください。向かって右側にあります) のスライド式の黒い小さなノブ (ライトプロテクトノッチ) を書き込み禁止状態 (穴が空いている状態) にしていませんか?

一旦フロッピーを取り出して、確認してください。もしそうなっていなくとも、これで直る場合もあります。もう一度お試し下さい。

どうしても無理なら、MiCS 相談室 (内線 2578) までお電話下さい。補助員が対処しに行きます。

Q. Mac でことえり入力時にカタカナしか出ない。

A. キーボート左下の Capslock キーが押されています。

もう一度 CapsRock キーを押して、上に飛び出た状態 (押し込まれていない状態) にして下さい。それでも直らなければ、鉛筆メニューから「操作パレット表示」を選んで、操作パレットの「あ」という文字のボタンを押して下さい。シフトキーを押しながら打つと、カタカナが出ます。

A.4.5 Program 編

Q. math.h を使ったらコンパイルできない。

A. UNIX 上で C 言語で、sin 関数などのいわゆる math.h を include するようなプログラムを cc しようとした時 NeXT ではそのままでコンパイル出来るのですが、cc2000 や csosf や SUN では数学ライブラリを組み込む-lm オプションが必要です。

例：`cc test.c -lm`

＃それでも出来ない時... プログラムミスかな？(^_^;)

A.4.6　その他

Q.　フロッピーディスクを買いたいのですが、、

A.　基本的に電気屋さんやパソコンショップで売っています。種類は 3.5’ 2HD（サンテンゴインチ ニイエイチディー）と呼ばれるものです。
学内では丸善などで取り扱っています。但し少し市場より値が高いので大量に買うのなら他のパソコンショップなどで買う方が良いでしょう。また、異なる機種間でデータのやり取りをされるのであれば 3.5’ 2DD（ニイディーディー）と呼ばれる種類の方がいい場合もあります。

付録 B

情報処理教室の利用について

学生便覧より抜粋

計算機を利用しての授業や自習のため、学内には 9 か所の情報処理教室を設置しています。この情報処理教室に設置している機器は、計算機システムの端末として利用できる他、パーソナル・コンピュータ（パソコン）としても利用できます。利用できるソフトウェアは各情報処理教室ごとに違いますので、その教室を管理する所管の事務室または計算機センター事務室で確認してください。

(1) 情報処理教室の利用機器等

建物	教室名	機器		所管
計算機科学研究所棟 2 階	C1 情報処理教室	FMV-466D	32 台	計算機科学研究所
計算機科学研究所棟 3 階	C2 情報処理教室	PanaStation	22 台	理学部
計算機科学研究所棟 3 階	C3 情報処理教室	NeXT station	15 台	計算機科学研究所
計算機科学研究所棟 4 階	C4 情報処理教室	NeXT station	6 台	理学部
1 号館 2 階	11 情報処理教室	PowerMacintosh6100	90 台	一般教育研究センター
2 号館 4 階	21 情報処理教室	DEC-3300	40 台	理学部
3 号館 2 階	31 情報処理教室	PowerMacintosh6100	44 台	外国語学部
5 号館 1 階	51 情報処理教室	FMR-280H	35 台	経済学部
5 号館 2 階	52 情報処理教室	Compaq Contura	45 台	経営学部

(2) 利用資格

本学の学生であれば自由に利用することができますが、利用機器によっては、計算機利用資格 (ID) が必要となります。計算機利用資格の申請は、計算機センターで行ってください。

(3) 利用時間

平日　　午前 8 時 45 分〜午後 8 時まで

土曜日　午前 8 時 45 分〜午後 5 時まで

情報処理教室の利用は授業を優先しますが、授業のない時間帯は研究や自習として自由に利用できます。授業に利用される時間帯は、「情報処理教室利用時間割表」として各建物の掲示板に掲示しますので、授業の有無を確認のうえ、利用してください。なお、日曜日・祝日、夏期一斉休業期間および年末年始期間には使用できません。その他、保守等によるシステム停止日、清掃等による閉室日は、その都度掲示で連絡します。

(4) 入室方法

入室は、学生証を情報処理教室出入口のカード読み取り装置に通し開錠のうえ、入室してください。なお、C2、C3、C4、11 情報処理教室の利用時間は開錠しています。ただし平日の午後 4 時 30 分以降および土曜日の正午以降に C2、C3、C4 の情報処理教室を利用する場合は、事前に計算機科学研究所事務室に届け出のうえ鍵を借用してください。また、52 情報処理教室への入室は常時鍵での開閉となりますので、経営学部事務室で鍵を借用してください。

(5) 利用心得

利用者は、次に掲げる利用心得を遵守してください。

1. 利用後は、機器の電源を切ること。
2. 最終利用者は、室内を消灯すること。
3. C2、C3、C4 および 52 情報処理教室での最終利用者は施錠し、鍵を借用した事務室の扉の郵便受けに返却すること。
4. 教室内のマニュアルおよび備品の持ち出しの禁止
5. 教室内での飲食及び喫煙の禁止
6. その他所管の学部事務室から特に指示のあった場合は、これに従うこと。

付録 C

著作権法（抜粋）

※出典　ニフティサーブ　法令データベース
※タイプミスなどの誤りが含まれている可能性があることを予めご了承ください。

（昭和四十五年五月六日法律第四十八号）
昭和四十六年一月一日
平成五年一一月一二日法律第八九号
著作権法（明治三十二年法律第三十九号）の全部を改正する。

第一章　総則
第一節　通則
（目的）
第一条　この法律は、著作物並びに実演、レコード、放送及び有線放送に関し著作者の権利及びこれに隣接する権利を定め、これらの文化的所産の公正な利用に留意しつつ、著作者等の権利の保護を図り、もつて文化の発展に寄与することを目的とする。

（定義）
第二条　この法律において、次の各号に掲げる用語の意義は、当該各号に定めるところによる。
一　著作物　思想又は感情を創作的に表現したものであつて、文芸、学術、美術又は音楽の範囲に属するものをいう。
二　著作者　著作物を創作する者をいう。
三　実演　著作物を、演劇的に演じ、舞い、演奏し、歌い、口演し、朗詠し、又はその他の方法により演ずること（これらに類する行為で、著作物を演じないが芸能的な性質を有するものを含む。）をいう。
四　実演家　俳優、舞踊家、演奏家、歌手その他実演を行なう者及び実演を指揮し、又は演出する者をいう。
五　レコード　蓄音機用音盤、録音テープその他の物に音を固定したもの（音をもつぱら映像とともに再生することを目的とするものを除く。）をいう。
六　レコード製作者　レコードに固定されている音を最初に固定した者をいう。
七　商業用レコード　市販の目的をもつて製作されるレコードの複製物をいう。
八　放送　公衆によつて直接受信されることを目的として無線通信の送信を行なうことをいう。
九　放送事業者　放送を業として行なう者をいう。
九の二　有線放送　有線送信のうち、公衆によつて同一の内容の送信が同時に受信されることを目的として行うものをいう。
九の三　有線放送事業者　有線放送を業として行う者をいう。
十　映画製作者　映画の著作物の製作に発意と責任を有する者をいう。
十の二　プログラム　電子計算機を機能させて一の結果を得ることができるようにこれに対する指令を組み合わせたものとして表現したものをいう。
十の三　データベース　論文、数値、図形その他の情報の集合物であつて、それらの情報を電子計算機を用いて検索することができるように体系的に構成したものをいう。
十一　二次的著作物　著作物を翻訳し、編曲し、若しくは変形し、又は脚色し、映画化し、その他翻案することにより創作した著作物をいう。
十二　共同著作物　二人以上の者が共同して創作した著作物であつて、その各人の寄与を分離して個別的に利用することができないものをいう。
十三　録音　音を物に固定し、又はその固定物を増製することをいう。
十四　録画　映像を連続して物に固定し、又はその固定物を増製することをいう。
十五　複製　印刷、写真、複写、録音、録画その他の方法により有形的に再製することをいい、次に掲げるものに

ついては、それぞれ次に掲げる行為を含むものとする。　イ　脚本その他これに類する演劇用の著作物　当該著作物の上演、放送又は有線放送を録音し、又は録画すること。　ロ　建築の著作物　建築に関する図面に従つて建築物を完成すること。
十六　上演　演奏（歌唱を含む。以下同じ。）以外の方法により著作物を演ずることをいう。
十七　有線送信　公衆によつて直接受信されることを目的として有線電気通信の送信（有線電気通信設備で、その一の部分の設置の場所が他の部分の設置の場所と同一の構内（その構内が二以上の者の占有に属している場合には、同一の者の占有に属する区域内）にあるものによる送信を除く。）を行うことをいう。
十八　口述　朗読その他の方法により著作物を口頭で伝達すること（実演に該当するものを除く。）をいう。
十九　上映　著作物を映写幕その他の物に映写することをいい、これに伴つて映画の著作物において固定されている音を再生することを含むものとする。
二十　頒布　有償であるか又は無償であるかを問わず、複製物を公衆に譲渡し、又は貸与することをいい、映画の著作物又は映画の著作物において複製されている著作物にあつては、これらの著作物を公衆に提示することを目的として当該映画の著作物の複製物を譲渡し、又は貸与することを含むものとする。
二十一　国内　この法律の施行地をいう。

※ 略 ※

第二章　著作者の権利
第一節　著作物
（著作物の例示）
第十条　この法律にいう著作物を例示すると、おおむね次のとおりである。
一　小説、脚本、論文、講演その他の言語の著作物
二　音楽の著作物
三　舞踊又は無言劇の著作物
四　絵画、版画、彫刻その他の美術の著作物
五　建築の著作物
六　地図又は学術的な性質を有する図面、図表、模型その他の図形の著作物
七　映画の著作物
八　写真の著作物
九　プログラムの著作物
２　事実の伝達にすぎない雑報及び時事の報道は、前項第一号に掲げる著作物に該当しない。
３　第一項第九号に掲げる著作物に対するこの法律による保護は、その著作物を作成するために用いるプログラム言語、規約及び解法に及ばない。この場合において、これらの用語の意義は、次の各号に定めるところによる。
一　プログラム言語　プログラムを表現する手段としての文字その他の記号及びその体系をいう。
二　規約　特定のプログラムにおける前号のプログラム言語の用法についての特別の約束をいう。
三　解法　プログラムにおける電子計算機に対する指令の組合せの方法をいう。

（二次的著作物）
第十一条　二次的著作物に対するこの法律による保護は、その原著作物の著作者の権利に影響を及ぼさない。

（編集著作物）
第十二条　編集物（データベースに該当するものを除く。以下同じ。）でその素材の選択又は配列によつて創作性を有するものは、著作物として保護する。
２　前項の規定は、同項の編集物の部分を構成する著作物の著作者の権利に影響を及ぼさない。

（データベースの著作物）
第十二条の二　データベースでその情報の選択又は体系的な構成によつて創作性を有するものは、著作物として保護する。
２　前項の規定は、同項のデータベースの部分を構成する著作物の著作者の権利に影響を及ぼさない。

（権利の目的とならない著作物）
第十三条　次の各号のいずれかに該当する著作物は、この章の規定による権利の目的となることができない。
一　憲法その他の法令
二　国又は地方公共団体の機関が発する告示、訓令、通達その他これらに類するもの
三　裁判所の判決、決定、命令及び審判並びに行政庁の裁決及び決定で裁判に準ずる手続により行なわれるもの
四　前三号に掲げるものの翻訳物及び編集物で、国又は地方公共団体の機関が作成するもの

第二節　著作者
（著作者の推定）

第十四条　著作物の原作品に、又は著作物の公衆への提供若しくは提示の際に、その氏名若しくは名称（以下「実名」という。）又はその雅号、筆名、略称その他実名に代えて用いられるもの（以下「変名」という。）として周知のものが著作者名として通常の方法により表示されている者は、その著作物の著作者と推定する。

（職務上作成する著作物の著作者）
第十五条　法人その他使用者（以下この条において「法人等」という。）の発意に基づきその法人等の業務に従事する者が職務上作成する著作物（プログラムの著作物を除く。）で、その法人等が自己の著作の名義の下に公表するものの著作者は、その作成の時における契約、勤務規則その他に別段の定めがない限り、その法人等とする。
２　法人等の発意に基づきその法人等の業務に従事する者が職務上作成するプログラムの著作物の著作者は、その作成の時における契約、勤務規則その他に別段の定めがない限り、その法人等とする。

（映画の著作物の著作者）
第十六条　映画の著作物の著作者は、その映画の著作物において翻案され、又は複製された小説、脚本、音楽その他の著作物の著作者を除き、制作、監督、演出、撮影、美術等を担当してその映画の著作物の全体的形成に創作的に寄与した者とする。ただし、前条の規定の適用がある場合は、この限りでない。

第三節　権利の内容
第一款　総則
（著作者の権利）
第十七条　著作者は、次条第一項、第十九条第一項及び第二十条第一項に規定する権利（以下「著作者人格権」という。）並びに第二十一条から第二十八条までに規定する権利（以下「著作権」という。）を享有する。
２　著作者人格権及び著作権の享有には、いかなる方式の履行をも要しない。
第二款　著作者人格権

（公表権）
第十八条　著作者は、その著作物でまだ公表されていないもの（その同意を得ないで公表された著作物を含む。次項において同じ。）を公衆に提供し、又は提示する権利を有する。当該著作物を原著作物とする二次的著作物についても、同様とする。
２　著作者は、次の各号に掲げる場合には、当該各号に掲げる行為について同意したものと推定する。
一　その著作物でまだ公表されていないものの著作権を譲渡した場合　当該著作物をその著作権の行使により公衆に提供し、又は提示すること。
二　その美術の著作物又は写真の著作物でまだ公表されていないものの原作品を譲渡した場合　これらの著作物をその原作品による展示の方法で公衆に提示すること。
三　第二十九条の規定によりその映画の著作物の著作権が映画製作者に帰属した場合　当該著作物をその著作権の行使により公衆に提供し、又は提示すること。

（氏名表示権）
第十九条　著作者は、その著作物の原作品に、又はその著作物の公衆への提供若しくは提示に際し、その実名若しくは変名を著作者名として表示し、又は著作者名を表示しないこととする権利を有する。その著作物を原著作物とする二次的著作物の公衆への提供又は提示に際しての原著作物の著作者名の表示についても、同様とする。
２　著作物を利用する者は、その著作者の別段の意思表示がない限り、その著作物につきすでに著作者が表示しているところに従つて著作者名を表示することができる。
３　著作者名の表示は、著作物の利用の目的及び態様に照らし著作者が創作者であることを主張する利益を害するおそれがないと認められるときは、公正な慣行に反しない限り、省略することができる。

（同一性保持権）
第二十条　著作者は、その著作物及びその題号の同一性を保持する権利を有し、その意に反してこれらの変更、切除その他の改変を受けないものとする。
２　前項の規定は、次の各号のいずれかに該当する改変については、適用しない。
一　第三十三条第一項（同条第四項において準用する場合を含む。）又は第三十四条第一項の規定により著作物を利用する場合における用字又は用語の変更その他の改変で、学校教育の目的上やむを得ないと認められるもの
二　建築物の増築、改築、修繕又は模様替えによる改変
三　特定の電子計算機においては利用し得ないプログラムの著作物を当該電子計算機において利用し得るようにするため、又はプログラムの著作物を電子計算機においてより効果的に利用し得るようにするために必要な改変
四　前三号に掲げるもののほか、著作物の性質並びにその利用の目的及び態様に照らしやむを得ないと認められる改変

第三款　著作権に含まれる権利の種類
（複製権）
第二十一条　著作者は、その著作物を複製する権利を専有する。

（上演権及び演奏権）
第二十二条　著作者は、その著作物を、公衆に直接見せ又は聞かせることを目的として（以下「公に」という。）上演し、又は演奏する権利を専有する。

（放送権、有線送信権等）
第二十三条　著作者は、その著作物を放送し、又は有線送信する権利を専有する。
２　著作者は、放送され、又は有線送信されるその著作物を受信装置を用いて公に伝達する権利を専有する。

（口述権）
第二十四条　著作者は、その言語の著作物を公に口述する権利を専有する。

（展示権）
第二十五条　著作者は、その美術の著作物又はまだ発行されていない写真の著作物をこれらの原作品により公に展示する権利を専有する。

（上映権及び頒布権）
第二十六条　著作者は、その映画の著作物を公に上映し、又はその複製物により頒布する権利を専有する。
２　著作者は、映画の著作物において複製されているその著作物を公に上映し、又は当該映画の著作物の複製物により頒布する権利を専有する。

（貸与権）
第二十六条の二　著作者は、その著作物（映画の著作物を除く。）をその複製物（映画の著作物において複製されている著作物にあつては、当該映画の著作物の複製物を除く。）の貸与により公衆に提供する権利を専有する。

（翻訳権、翻案権等）
第二十七条　著作者は、その著作物を翻訳し、編曲し、若しくは変形し、又は脚色し、映画化し、その他翻案する権利を専有する。

（二次的著作物の利用に関する原著作者の権利）
第二十八条　二次的著作物の原著作物の著作者は、当該二次的著作物の利用に関し、この款に規定する権利で当該二次的著作物の著作者が有するものと同一の種類の権利を専有する。

※ 略 ※

（私的使用のための複製）
第三十条　著作権の目的となつている著作物（以下この款において単に「著作物」という。）は、個人的に又は家庭内その他これに準ずる限られた範囲内において使用すること（以下「私的使用」という。）を目的とする場合には、公衆の使用に供することを目的として設置されている自動複製機器（複製の機能を有し、これに関する装置の全部又は主要な部分が自動化されている機器をいう。）を用いて複製するときを除き、その使用する者が複製することができる。
２　私的使用を目的として、デジタル方式の録音又は録画の機能を有する機器（放送の業務のための特別の性能その他の私的使用に通常供されない特別の性能を有するもの及び録音機能付きの電話機その他の本来の機能に附属する機能として録音又は録画の機能を有するものを除く。）であつて政令で定めるものにより、当該機器によるデジタル方式の録音又は録画の用に供される記録媒体であつて政令で定めるものに録音又は録画を行う者は、相当な額の補償金を著作権者に支払わなければならない。

（図書館等における複製）
第三十一条　図書、記録その他の資料を公衆の利用に供することを目的とする図書館その他の施設で政令で定めるもの（以下この条において「図書館等」という。）においては、次に掲げる場合には、その営利を目的としない事業として、図書館等の図書、記録その他の資料（以下この条において「図書館資料」という。）を用いて著作物を複製することができる。
　一　図書館等の利用者の求めに応じ、その調査研究の用に供するために、公表された著作物の一部分（発行後相当期間を経過した定期刊行物に掲載された個個の著作物にあつては、その全部）の複製物を一人につき一部提供する場合
　二　図書館資料の保存のため必要がある場合
　三　他の図書館等の求めに応じ、絶版その他これに準ずる理由により一般に入手することが困難な図書館資料の複製物を提供する場合

（引用）
第三十二条　公表された著作物は、引用して利用することができる。この場合において、その引用は、公正な慣行に

合致するものであり、かつ、報道、批評、研究その他の引用の目的上正当な範囲内で行なわれるものでなければならない。
２　国又は地方公共団体の機関が一般に周知させることを目的として作成し、その著作の名義の下に公表する広報資料、調査統計資料、報告書その他これらに類する著作物は、説明の材料として新聞紙、雑誌その他の刊行物に転載することができる。ただし、これを禁止する旨の表示がある場合は、この限りでない。

※ 略 ※

第四十七条　美術の著作物又は写真の著作物の原作品により、第二十五条に規定する権利を害することなく、これらの著作物を公に展示する者は、観覧者のためにこれらの著作物の解説又は紹介をすることを目的とする小冊子にこれらの著作物を掲載することができる。

（プログラムの著作物の複製物の所有者による複製等）
第四十七条の二　プログラムの著作物の複製物の所有者は、自ら当該著作物を電子計算機において利用するために必要と認められる限度において、当該著作物の複製又は翻案（これにより創作した二次的著作物の複製を含む。）をすることができる。ただし、当該利用に係る複製物の使用につき、第百十三条第二項の規定が適用される場合は、この限りでない。
２　前項の複製物の所有者が当該複製物（同項の規定により作成された複製物を含む。）のいずれかについて滅失以外の事由により所有権を有しなくなつた後には、その者は、当該著作権者の別段の意思表示がない限り、その他の複製物を保存してはならない。

（出所の明示）
第四十八条　次の各号に掲げる場合には、当該各号に規定する著作物の出所を、その複製又は利用の態様に応じ合理的と認められる方法及び程度により、明示しなければならない。

※ 略 ※

第四節　保護期間
（保護期間の原則）
第五十一条　著作権の存続期間は、著作物の創作の時に始まる。
２　著作権は、この節に別段の定めがある場合を除き、著作者の死後（共同著作物にあつては、最終に死亡した著作者の死後。次条第一項において同じ。）五十年を経過するまでの間、存続する。

※ 略 ※

第七節　権利の行使
（著作物の利用の許諾）
第六十三条　著作権者は、他人に対し、その著作物の利用を許諾することができる。
２　前項の許諾を得た者は、その許諾に係る利用方法及び条件の範囲内において、その許諾に係る著作物を利用することができる。
３　第一項の許諾に係る著作物を利用する権利は、著作権者の承諾を得ない限り、譲渡することができない。
４　著作物の放送又は有線放送についての第一項の許諾は、契約に別段の定めがない限り、当該著作物の録音又は録画の許諾を含まないものとする。

※ 略 ※

（侵害とみなす行為）
第百十三条　次に掲げる行為は、当該著作者人格権、著作権、出版権又は著作隣接権を侵害する行為とみなす。
　一　国内において頒布する目的をもつて、輸入の時において国内で作成したとしたならば著作者人格権、著作権、出版権又は著作隣接権の侵害となるべき行為によつて作成された物を輸入する行為
　二　著作者人格権、著作権、出版権又は著作隣接権を侵害する行為によつて作成された物（前号の輸入に係る物を含む。）を情を知つて頒布し、又は頒布の目的をもつて所持する行為
２　プログラムの著作物の著作権を侵害する行為によつて作成された複製物（当該複製物の所有者によつて第四十七条の二第一項の規定により作成された複製物並びに前項第一号の輸入に係るプログラムの著作物の複製物及び当該複製物の所有者によつて同条第一項の規定により作成された複製物を含む。）を業務上電子計算機において使用する行為は、これらの複製物を使用する権原を取得した時に情を知つていた場合に限り、当該著作権を侵害する行為とみなす。
３　著作者の名誉又は声望を害する方法によりその著作物を利用する行為は、その著作者人格権を侵害する行為とみなす。

※ 以下略 ※

付録 D

参考文献

ここにあげる文献にある記述が全て cc 環境で適用出来るとは限らないことに注意してください。

—— **Unix 全般、シェル、コマンドなどについて** ——

たのしい UNIX -UNIX への招待- 坂本　文著：アスキー出版局刊
月刊雑誌 UNIX Magazine の連載を集成した UNIX 初心者向けの入門書。

続 たのしい UNIX -シェルへの招待- 坂本　文著：アスキー出版局刊
上の一冊の続編。今度はシェルについて解説してくれる。

実用 UNIX ハンドブック 舟本　奨著：ナツメ社
UNIX コマンドの簡単なリファレンス。

UNIX & X コマンド辞典 Alan Southerton, Edwin C. Perkins, Jr. 著、加藤大典訳：丸善、1995 年 7 月刊、3,800 円
UNIX コマンドの非常に詳細なリファレンス。比較的高価ではあるが、400 ページ以上にわたり各種のコマンドに関して、およそありとあらゆる例が挙げてある。多くのユーザのコマンドの使い方を試す手間をかなり省いてくれるだろう。ざーっと眺めて新しいコマンドの使い道を発見するのにも良し。

DOS ユーザのための UNIX 入門 Douglas W. Topham 著、中西隆訳：技術評論社、1991 年刊、1900 円
タイトル通り、DOS ユーザの視点から見た UNIX の機能説明です。後半はむしろシステム管理社向けに書いてあります。

UNIX C SHELL フィールドガイド G・アンダーソン、P・アンダーソン著：落水　浩一郎、大木　敦雄訳：パーソナルメディア刊
csh のほぼ完全なガイド。

UNIX step++ シェルプログラミングのコツ 西沼　行博著：マグロウヒル刊
残念ながら csh ではなく、sh についての説明が主体。記述も少々古いがシェルを使いこなしたい人には便利。

The Unix Super Text 山口　和紀監修、于 旭・中村 敦司・新城 靖・西山 博泰・古瀬 一隆・石川 佳治・佐々木 重雄・林 謙一・荻原 一隆・金谷 英信・鈴木 孝幸・黒石 和宏 著：技術評論社刊
BSD と呼ばれるタイプの UNIX 主体に書いてあるため、cc 環境とは相違がある場合が散見される

が、一般的な概念からハウツーまで、広い分野に関して教えてくれる。上下巻の二冊組。上巻は一般的な UNIX の使い方、電子メイル、ニュース、X ウィンドウなどについて。下巻は LaTeX、プログラミング、システム管理などについて。高価なのが難点か。

誰にでも使える UNIX 講座 安岡孝一著 : ソフトバンク社刊、1992 年

`yasuoka` さんが `root` さんに色々教えてもらう対話形式の本。login, logout から始まって UNIX の基本的なツール sh, csh, sed, make, awk の使い方がこれ一冊でわかる。UNIX を使いこなしてみたい人の入門書。

MH & xmh Jerry Peek 著、倉骨彰 訳、砂原秀樹・鈴木麗 監訳 : アスキー出版局刊、1994 年 11 月、5,800 円

MH に関するマニュアル。

UNIX の環境設定 久野禎子、久野靖著 : アスキー出版、1993 年 9 月刊、1,800 円

cc 環境は最初の段階で殆ど全ての環境設定が行われていますが、この本は様々な UNIX での機能設定の方法について説明されています。これから自分の環境を構築したい、しなければならない人向けといえるでしょう。

—— Mule について ——

入門 Mule 大木　敦雄著 : アスキー出版刊

まさに Mule の入門書。

Mule でにゃん! だって UNIX ですもの 平山弘之著 : メロン出版刊、1,600 円

おじさんには頭がいたくなるような題名ですが、内容もそのとおり、くだけたもので、計算機を専門としない超ビギナーには、面白く読めるものと思います。mule に関する一切の難しいことが、ばっさり省略されている ところがこの本の特徴です。

—— Emacs について ——

Emacs は Mule の前のバージョンです。基本的な操作などに付いては殆どこの Emacs の文献が利用できます。参考にしてください。

GNU Emacs 入門 (株) 京都ソフトウェアリサーチメディアデザイン編 : オーム社刊、1994

Emacs を非常に分かり安く解説しています。自習にも向いていると思います。付録としてついている切り取り式のコマンド一覧も便利。

GNU Emacs Debra Cameron and Bill Rosenblatt 著 : ハイパーウェア監訳 : ソフトバンク株式会社刊

Emacs のほぼ完全なガイド。NutsShell（ナッツ（どんぐり）のカラ？）シリーズと呼ばれる非常に詳細な Unix 関係のドキュメントのシリーズの一冊。

GNU Emacs 完全ガイド M.A. シュノーバー、J.S. ボウイ、W.R. アーノルド 著・小畑喜一、磯谷正孝、山野修、林秀幸訳 : アジソン・ウェスレイ・トッパン刊（情報科学シリーズ 30,35）(上巻 : 1992 年 11 月 3,400 円、下巻:1993 年 3 月 3,900 円)

本書が取り上げている GNU Emacs は Version が、18.57 です。紐解く時に「完全ガイド」の安心感があります。

GNU Emacs マニュアル Richard Stallman 著 : 竹内郁雄・天海良治訳 : 共立出版刊、1988 年 2 月、2,930 円

GNU の御大自らの解説です。《古典》の部類なのかもしれませんが。

入門 NEmacs 大木　敦雄著 : アスキー出版刊

Emacs だけでなく、EGG, MHE, GNUS などについても説明してくれている。

—— **TEX について** ——

楽々LATEX 野寺　隆志著 : 共立出版刊、第二版、1994、2,900 円

LATEX を用いた TEX の入門書。初心者には最適。

LATEX 入門–美文書作成のポイント – 奥村　晴彦監修 : 技術評論社刊、1994

TEX についていろいろ丁寧に教えてくれる。

やさしい LATEX のはじめかた すずき ひろのぶ著 : オーム社刊、1991

表題通り LATEX のやさしい入門書です。その割に, 便利なことがチョコチョコと書いてある。

日本語 LATEX 定番スタイル集 No.1, No.2 鷺谷　好輝著 : インプレス刊

京都産業大学でキャンパスライセンスを取得している LATEX のスタイルファイルの使い方解説書。きれいなスタイルファイルの見本としてもよい。

LATEX エラーマニュアル 今井豊著 : カットシステム (Tel.0423-94-2218) 刊、1994 年 6 月、2,300 円

マイナーなところからの出版のためか、話題になりませんでしたが「エラーをなおし、エラーから学ぶ本格的解説書」ではあります。AUCTEX があれば、半減しているのでしょうが、TEX を使う楽しさの半分は、「エラーとの格闘」ゲームではないでしょうか。その意味では、ゲーム本に分類するべきなのかもしれませんが。

てくてく TEX 阿瀬はる美著 : アスキー出版局刊、(上巻 1994 年 11 月、2,000 円、下巻 1994 年 12 月、2,000 円)

語り口の楽しさが、坂本文著「たのしい UNIX」に似ているのは、「UNIX MAGAZINE」1989/04-90/07 連載だったせいでしょうね。

逆引き LATEX D.J. バーガー著、引地　信之・引地　美恵子訳 : マグロウヒル刊

「こんなことが出来るのではないか？」と思った機能から、その方法を調べる。

LATEX トータルガイド 伊藤　和人著 : 秀和システムトレーディング刊

LATEX 技能の華麗な一覧表。

文書処理システム LATEX Leslie Lamport 著、Edgar Cooke・倉沢　良一監訳、大野　俊治・小暮　博道・藤浦　はる美著 : アスキー出版局刊

他の LATEX 解説書の説明では納得出来ない時の駆け込み寺。入門書なのに理論的。

改訂新版 TEX ブック Donald E. Knuth 著、斉藤信男監修、鷺谷好輝訳 : アスキー出版局刊

教祖様直筆 TEXnician の Bible。頭から噛じるに困難でも、昼寝の枕に最適。

—— インターネットについて ——

インターネット 村井純著：岩波書店、1995 年 11 月刊、650 円

岩波の新書版です。WIDE 代表、日本のミスターインターネットとも言える村井氏がインターネットの経緯、理念などについて述べています。多分に技術的な内容を含んでおり、技術と理念が同時に語られているところが良いと思います。

ハッピー・ネットワーキング 山本和彦著：アスキー出版局、1994 年 7 月刊、1,500 円

NEmacs, mail, news, FTP だけに焦点を絞った若者による新入生向けであることに好感が持てました。cc 環境では殆ど等価のドキュメントを `ccinfo` コマンドで印刷させることも出来ますが、綺麗に製本された物が欲しければやはりこれを買うことになるでしょう。

インターネット情報生活入門 グループまたたび著：技術評論社、1994 年 10 月刊、1,700 円。 インターネットの全体的な紹介をビジュアルにしている点がよいと思います。

インターネット参加の手引き〜1994 年度版 WIDE Project 編・村井純、吉村伸 監修：共立出版、bit 別冊、1994 年 7 月刊、4,800 円

情報はタップリ詰まってますけど、高価。1995 年度版が出るそうです。

インターネットユーザーズガイド Ed Krol 著・村井純 監訳：インターナショナル・トムソン・パブリッシング・ジャパン刊 1994 年 5 月刊、4,400 円

ブームに向けてタイミング良く出版されてスタンダードなものになったようですけれど悪訳を批判されたりもしていました。1995 年の 1 月に「改訂版」が出ていましたが、訳の改善がはたしてどの程度行なわれたのか、確認していません。(これでは紹介文にならない!)

インターネット・ナビゲータ Paul Gilster 著・菱山博陸訳：丸善、1995 年 2 月刊、6,592 円

原書第 2 版が底本。「本書は常にモデムでインターネット利用する人を念頭においています。」(p.vii)って台詞に、コロっと心底、騙されました。インターネット論としても優れているものだと思います。

—— WWW ページ作成、HTML について ——

インターネット ホームページデザイン 吉村信、家永百合子、鐙聡：翔泳社、1995 年 6 月刊、2,400 円

HTML 文法のリファレンス。かなり細かく書いてあり、殆ど全ての HTML 文法をカバーしていると思われる。各文法がどのクライアント向けの拡張なのか、即ちどのクライアントソフトでならどのような効果が出るのかについて特に注意が払われているので、多くの人にページを見て欲しいページ制作には欠かせません。

HTML 入門 WWW ページの作成と公開 ローラ・リメイ著、武舎広幸、久野禎子、久野靖訳：プレンティスホール出版、1995 年 12 月刊、3,900 円

WWW ページ制作に関するノウハウと技術がぎっしり詰まっています。ページのデザインに関しての助言も数多く、バランスの取れたページデザインや構成をする上で一度は見るのがお勧めです。

続 HTML 入門 新機能、CGI、Web の進化 ローラ・リメイ著：武舎広幸、久野禎子、久野靖訳：プレンティスホール出版、1995 年 12 月刊、3,900 円

上記「HTML 入門」の続編です。前作の出版以降に追加された機能について解説しています。特に CGI (Common Gateway Interface) を使って、プログラムによる機能を WWW に組み込む方法の紹介が含まれています。

はじめての HTML3.0 Sachi 著、宇野謙吉監修 : リブロス、1995 年 11 月刊、2,700 円

非常に簡潔に多くの HTML 記述についてまとめられています。クリッカブルマップ、テーブルも押さえており、せいぜい CGI についてほとんど抜かしている程度です。

索 引

配布、改変

連絡など

訂正や誤りに対する連絡は京都産業大学の計算機センター事務室までお願いします。ここはまずい、こうした方が良いという相談は大歓迎です。我々の環境は時代に合わせてどんどん変化します。このドキュメントもまずいところはどんどん直して変えていって欲しいと思います。そうしてこのドキュメントが実際に役に立つものとして成熟して行けば良いなと考えているのです。

執筆者

谷川 正幸　竹内 茂夫　大本 英徹　安田 豊
山崎 英知　辻本 将彦　笠 克明　小坂田 浩孝
松浦 正和　坪内 伸夫　吉田 浩史　重田 裕之
土肥 順一　岡田 光博　開原 潮　尾崎 孝治

発行

京都産業大学 計算機センター
〒 603 京都市北区上賀茂本山
電話 075-705-1483

謝辞

このガイドでは計算機センターが、コンピュータを初めて使うところから電子メイルやニュースが使えるようになるまでの一連のチュートリアルを書いています。残りの部分については cc 環境利用者の有志から原稿を頂きました。

まず最近話題の WWW では、HTML おいしいホームページの作り方を外国語学部の竹内茂夫先生に書いていただきました。Mule の章とそのリファレンス、レポートシステムと FAQ の部分を当時理学部計算機科学科の学生だった尾崎君が書いてくれています。そして彼は現在計算機センター職員となりこのガイドの編集に携わっています。また、FAQ は計算機センターの MiCS 補助員（計算機運用補助員）がまとめてくれたものを利用しています。LaTeX の章を理学研究科物理学専攻の山崎君が書いてくれました。LaTeX 前半の一部分は同じく物理学科の辻本君が書き起こしてくれたものが元になっています。AUCTeX と TGIF の章については理学研究科物理学専攻の松浦君が、Mathematica の章については理学研究科数学専攻の笠君が書いてくれました。MODEM から telnet の章は経済学部の小坂田君[1]が書いてくれてものを元に計算機センターの開原君が書いてくれました。GNUPLOT の章は理学部の谷川正幸先生が書いてくださいました。NeXT の章は計算機センターの開原君が書いてくれました。彼はまた著作権法の部分をまとめてくれています。NQS、*xv*、xpaint の章と UNIX コマンドリファレンスは元計算機センターの安田君[2]が cc 環境の一ユーザとして書いています。ダイアルアップ IP は安田君、尾崎君と一般教育研究センターの吉田君の合作となっています。今回も突然の依頼となりましたが工学部の黒住先生には巻頭を引き受けていただきました。ガイド執筆では直接現れませんが、cc 環境の Mule の設定には理学部の立木先生や他の方々から多くの御協力を得て何とかなりました。DEC-3000/300 の特製 xdm は当時工学研究科情報通信工学専攻の為永君[3]の作品です。その他にも多くの方の協力を戴きました。

それら全ての人に感謝いたします。

経緯

cc 環境は 1993 年の夏に計算機センターが導入した UNIX マシン群を核としたコンピュータ利用環境です。それまで大学の中に共用 UNIX 環境が存在しなかったという事もあって、当時大学の中では UNIX の利用者は少なかったのですが、1994 年の春から一般学生にも授業と関わりなくアカウントを出すということになりました。そこで授業などとは関係なく UNIX 環境を利用することになる学生のために、ガイドが必要となったのです。そうして「UNIX ガイド」初版が cc 環境の設計者でもあった当時の計算機センター職員安田君によって書かれました。

1994 年の夏には多くの人の執筆協力を得て「UNIX ガイド追補版」を出版する事が出来ました。これは学内で UNIX 利用者がそれだけ育った事を象徴する出来事でした。1995 年この二冊を合本して更に内容を加えた第二版が出版されました。

そしてこの度「UNIX ガイド」はさらに内容を加えて「コンピュータガイド・インターネット編」と「コンピュータガイド・アプリケーション編」に分冊されることになりました。第二版がすでに 400 ページ近い厚さになっており、常に携帯してもらうために、もう少し見やすく軽くならないかとの配慮からです。

Netscape に代表されるブラウザは簡単な操作で WWW の利用を可能にし、利用者層を拡大しました。いわばインターネットの「大衆化」を急激に進めたといえます。1996 年からは 4 回生が就職における求人情報収集のためにインターネットを使うでしょう。1993 年の cc 環境設計当時には想像できなかったほど、

[1] 小坂田君は 1995 年 3 月卒業

[2] 安田 豊君は 1995 年 11 月に京都産業大学を退職し神戸大学経済経営研究所に移られました。その後も cc 環境のために協力してくれています。

[3] 為永君は 1995 年 3 月修了

大衆化は進んでいます。「コンピュータガイド・インターネット編」はUNIXをあまり意識せずにインターネットを使う人達のために書かれています。一方、古くからのcc環境利用者、つまりUNIXをより使いたい人達のために「コンピュータガイド・アプリケーション編」が書かれています。

cc環境の設計者である安田君は、多くの利用者によって支えられている現在のcc環境をみて「1993年に苦労してcc環境の設計を行なっていたのが嘘のように昔の事に思える。最初の半年の利用者には何のドキュメントも用意してやれず苦労を掛けたと今更ながらに思う。」と述懐しています。

これからcc環境がどのように育っていくのか予想がつかない部分もありますが、このドキュメントがその助けになってくれればと願っています。

最後に。論文やレポート時期の貴重な時間をさいて、この「コンピュータガイド」を書いてくださった有志の方々、そしてその成果物のとりまとめにまるまる二週間を費やした編集者たち、こうしてcc環境を支えてくれているすべての人々に感謝します。

1996.3.25 計算機センター