はじめての BSD rev.

小久保 温 著

青森大学出版局 発行

はじめての BSD rev.

小久保 温 著

青森大学 出版局 発行

はじめに

この本は、はじめて UNIX システムを使う人を対象に、その使い方を間単に紹介するために書いたものです。Windows などの OS は既に使ったことがあるという前提で書いていて、それらと比較して OS とは一体何だろうかということも考えてもらえればと思っています

この本では、 FreeBSD を使って、 UNIX 系の OS とアプリケーションの使い方を紹介しています。システムは青森大学の B 演習室のものに合わせて書いてあるので、自分でインストールしたりすると、多少異なった点があるかもしれません。ログイン、ログアウトから、ファイル操作、 Mule と Canna、プロセス、インターネットなどを取り上げています。UNIX には、この本に書ききれないようなさまざまなツールや機能があり、それらを知れば知るほど簡単な操作で高度なことができるようになります。興味のある方は、是非、自分で調べてみてください。

FreeBSD は年に何回かバージョンアップがあり、 B 演習室の環境も変更されることがあります。この本のサポートページは `http://www.aomori-u.ac.jp/staff/inform/kokubo/BSD/` で、ここでいろいろと本に書ききれなかった情報をはじめとして変更点なども紹介していこうと思っています。

なお、この本は、元々、青森大学情報工学科 1 年生の情報工学演習の後期で使用するテキストとして書いたものでした。この授業は、現在では 2 年生後期の UNIX 演習という名前に変わっています。当時は FreeBSD も 2.x だったのですが、現在では 4.x にバージョンが上がってしまっていて、随分変わってしまいました。

2002 年 秋 小久保 温

謝辞

この本は、青森大学出版局から出版されました。出版の上でお世話になりました角田のり子先生はじめ、青森山田学園法人本部の方々に感謝いたします。

また、テキストは、授業を受ける学生のみなさんがあってはじめて意味があるものです。これまでのわたしの UNIX の授業を受講した学生のみなさんに感謝いたします。

それから、この本を書くのに使用した FreeBSD、 Cygwin、日本語 pLaTeX、その他のさまざまなソフトウェアは、基本的にフリーで配布されています。これらのすばらしいソフトウェアを書かれ、しかもフリーで配布してくださっているプログラマの方々に感謝いたします。

奇しくも先日、ちょうどこの本の執筆作業が佳境に入った頃、永遠に終わらない学園祭前日を描いた作品が長い年月を経て DVD 化されました。プログラムを書いたり、サーバを構築したり、イベントを開催したり、本を作ったり …、それらは終わって振り返ってみれば、学園祭前日の夢のようなものなのかもしれません。

そんな夢をひとしずくそっとすくいあげるのに、一体、何を費やしたのでしょうか。そんな試みにやさしく関わってくださった方に深く謹んで感謝いたします。

2002 年 9 月 小久保 温

目 次

第1章　UNIX システムとは何か

UNIX(ユニックス) システムとは、 OS(オーエス) の一種である。そして、この本では、 FreeBSD(フリー・ビーエスディ) という OS を使って、UNIX 系のシステムの使い方を紹介していく。と言っても、よくわからないと思うので、まずこの章では OS、それから UNIX システムなどについて簡単に説明することにしよう。

1.1　OS とは何か

短く説明すると、 OS とは、 Operating System(オペレーティング・システム)(操作するためのシステム) の略で、コンピュータを快適に使いこなすための、基本システムのプログラムである。

それでは、具体的に紹介していこう。

1.1.1　どんな OS があるのか

いわゆるパソコンと呼ばれているもので動く OS には、Microsoft 社の Windows(ウィンドウズ) シリーズ、Apple 社の Mac OS(マックオーエス) シリーズ、東京大学の坂村教授の TRON(トロン)、DOS(ドス) と呼ばれるいくつかの OS、そしてこの本で取り上げる UNIX 系のシステムなどがある。

パソコンで動く UNIX 系のシステムは、 PC-UNIX とも呼ばれていて、Linux(リナックス) や FreeBSD などがよく使われている。また、 MacOS も X(テン) (version 10) 以降は、UNIX をベースとした OS になっている。

図 1.1: Windows XP のデスクトップ

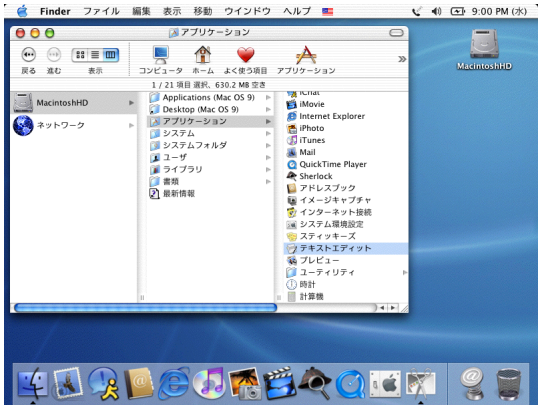


図 1.2: Mac OS X のデスクトップ

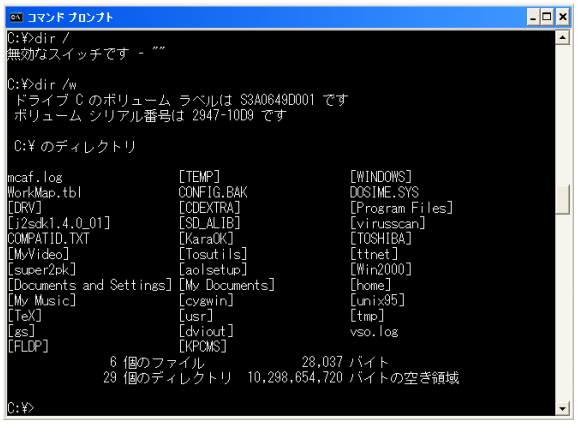


図 1.3: MS-DOS コマンド・プロンプト

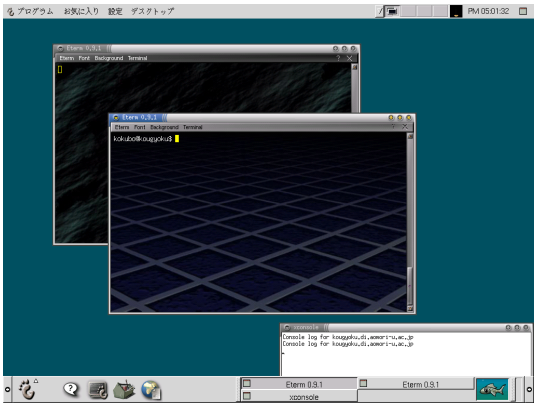


図 1.4: FreeBSD のデスクトップ (GNOME 使用)

1.1.2 OS の役割とは

コンピュータはいろいろなハードがつなぎ合わされて作られている。これらをうまく適切にコントロールするには、実際にはとても複雑な処理が必要となる。その複雑な部分を包み込んで活用して、ユーザから見たときには簡単な操作でコントロールすることができるようにするのが OS の役割である。

例えば、コンピュータでフロッピー・ディスクを使いたいとしよう。Windows の場合を思い出してみよう。デスクトップにある「マイコンピュータ」というアイコンをクリックするとウィンドウが開いて、その中に「3.5 インチ FD」というアイコンが見える。このアイコンがフロッピーを表してして、更にクリックすると中に入っているファイルのリストを見ることができる。

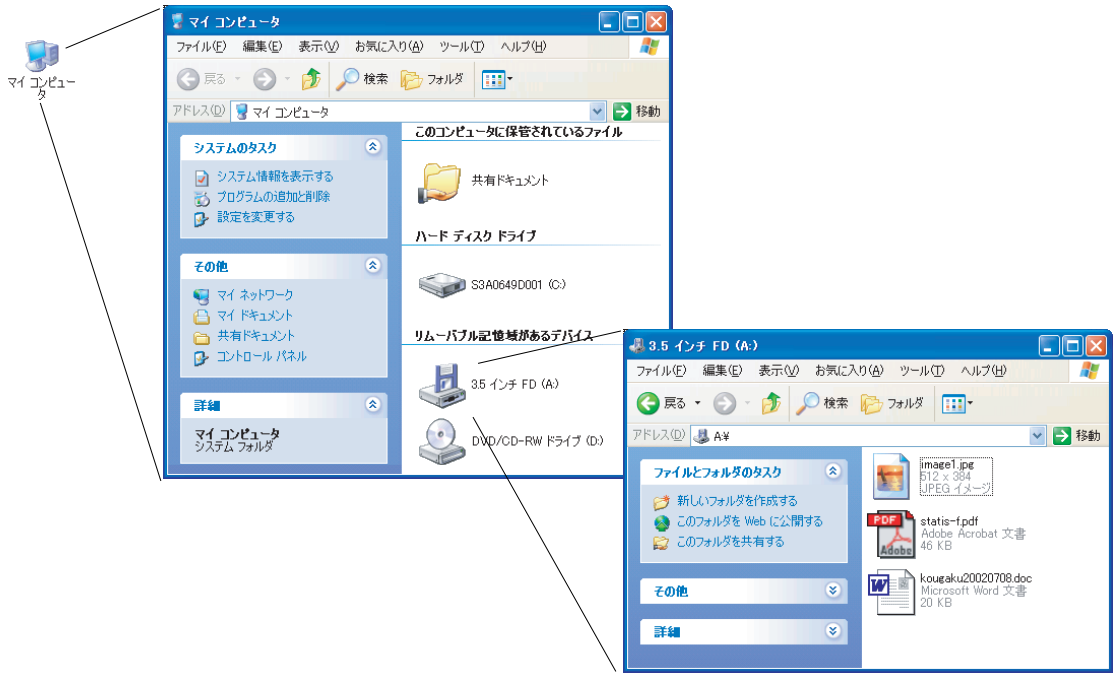


図 1.5: Windows のファイル一覧

また、ファイルをフロッピー・ディスクにコピーするには、ファイルのアイコンを持って、このフロッピーのアイコンに重ねてやればいい。

このようなアイコンの操作は、アイコンというシンボルを使って簡略化、抽象化された「記号操作」の一種であると言える。

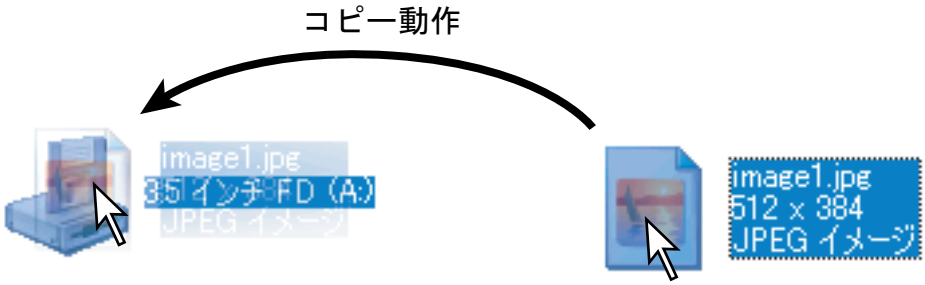


図 1.6: Windows でのファイルのコピー動作

ところで、このとき実際にコンピュータは何をしているのだろうか?

まず、フロッピーを回転させ、ドライブのヘッドをフロッピーに近づけて、フロッピー上のどこにどんなデータが書かれているかというインデックスを探して、それを読む。次にファイルの中身を読むには、またヘッドを移動して … 。

などという、気の遠くなるような複雑な操作が行なわれているのである。

こんなに複雑な操作を全部自分で一々やっていては、とてもコンピュータを気軽には使えないだろう。

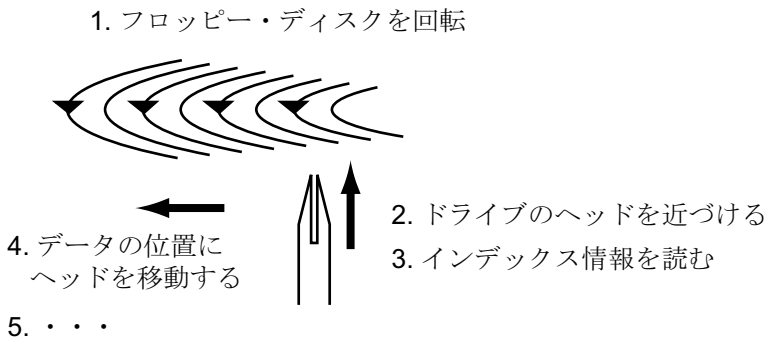


図 1.7: フロッピーを読むときのハードウェアの動作

Windows という OS では、このような複雑なハードに関する操作を包み込んで見えないようにしている。ユーザは単にアイコンをクリックするような「記号の操作」で、簡単にコンピュータのシステムを使えるようになっているのである。これは別に Windows だけでなく、他の OS でも同様である。

1.2　UNIX システムとはどういう OS か

1.2.1　UNIX システムの特徴

UNIX システムは 1970 年代のはじめに基本的な部分が開発され、それ以降も発展し続けてきた歴史の長い OS である。ちょっと難しい話になるが、最初に UNIX システムの特徴を短くまとめておこう。

1. **マルチユーザ**

   一台のマシンを何人もの人が同時に使うことができる。

2. **マルチタスク**

   一台のマシンで同時にいくつもの命令 (タスク) を実行することができる。

3. **階層型ファイル・システム**

   ファイル・システムがツリー状に構成されている。

4. **デバイスなどの取り扱いの統一**

   周辺機器などのデバイスにいたるまですべてがファイルとして統一的に取り扱われる。

5. **ネットワーク機能**

   システムが開発された初期の頃からネットワークを利用するためのシステムが用意されていた。

6. **主に C 言語で書かれたソース・コード**

   システムの多くの部分が C 言語で書かれていて、いろいろなマシンに移植しやすい。

これらのいくつかの特徴は、現在では多くの OS に備えられ、当たり前のこととなっているが、当時としてはシンプルかつ先進的なものであった。

1.2.2　UNIX システムと Windows

具体的な話に入ろう。UNIX システムでは、 Windows とやり方は異なるものの、文章を書いたり、プログラムを作ったり、電子メールやニュースを使ったり、WWW にアクセスしたり、絵を描いたりすることができる[1]。

では、この 2 つの OS ではどこが違うのかを見てみよう。

1. **コマンド・ライン・ベースと GUI**

   UNIX では、基本的にコンピュータに何かをさせるときには、コマンドをキーボードから打ち込む。コマンドは簡単な英語を短縮したものが多い。このようなシステムを、コマンド・ライン・ベースのシステムと呼ぶ。

[1] ちなみに、このテキストも、多くの部分は UNIX で作成している。

一方、 Windows では、アイコンと呼ばれる「絵」をクリックしたり、持って移動したりして、コンピュータを使う。このようなシステムを GUI (Graphical User Interface) のシステムと呼ぶ。

ただし、UNIX 系のシステムでも、X Window System という GUI を使うことができ、Windows のようにアイコンを用いて操作することができる。X Window System はユーザの好みによって、大幅に環境を変えることができるという特徴を持っている。特にウィンドウ・マネー

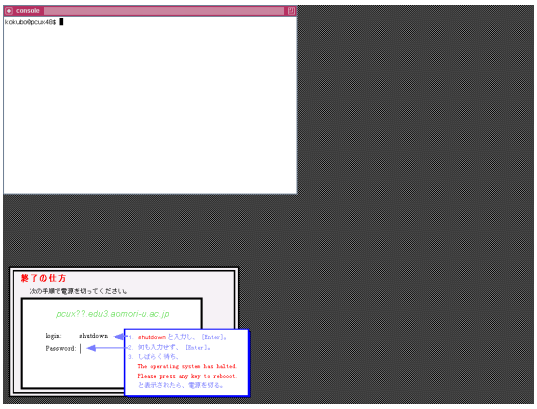


図 1.8: Twm

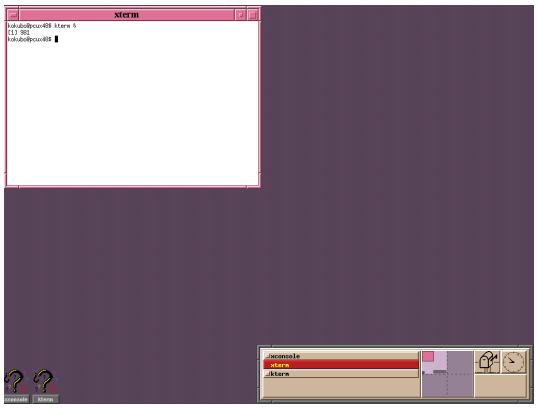


図 1.9: Fvwm

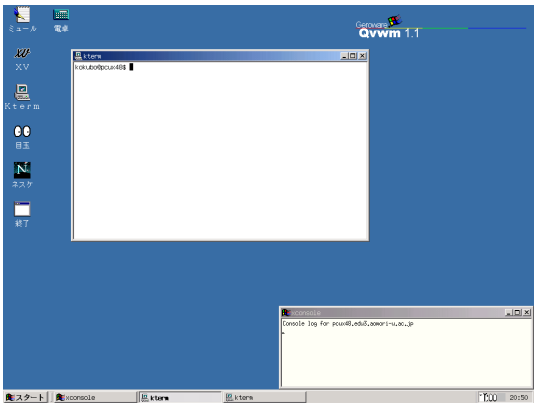


図 1.10: Qvwm

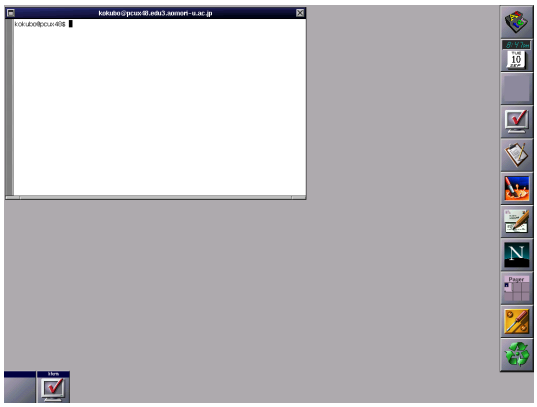


図 1.11: AfterStep

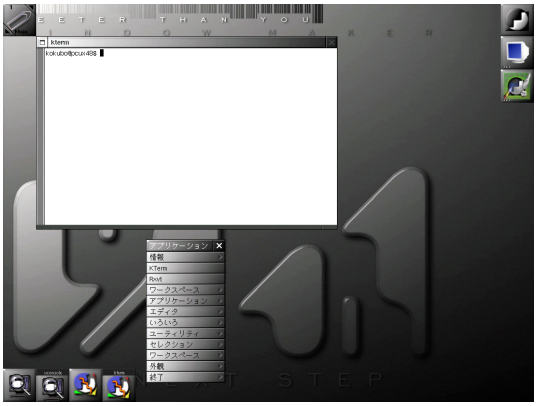


図 1.12: WindowMaker

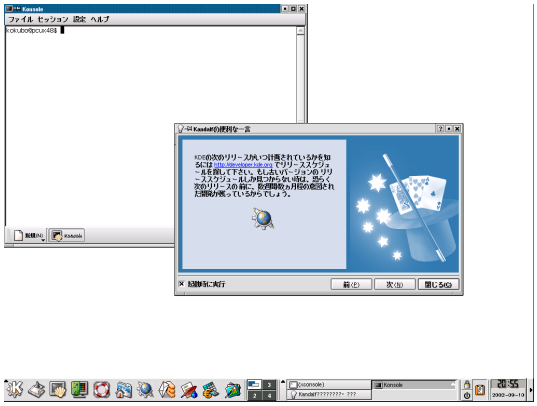


図 1.13: KDE

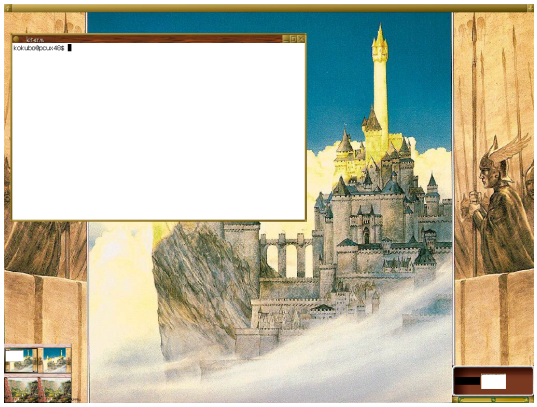


図 1.14: Enlightenment

ジャーというソフトを変更すると、全く異なった外見と操作方法になる。ウィンドウ・マネージャーには、図 1.4 の GNOME の他、Twm、Fvwm、Qvwm、AfterStep、WindowMaker、KDE、Enlightenment などがある。

2. **マルチユーザとシングルユーザ**

   UNIX 系のシステムでは、ネットワークごしにログインすることができて、一台のマシンを同時に何人もで使うことができる。

   例えば、何人もで同時にログインしてファイルを置いたりするようなことができる必要のあるインターネット・サービス・プロバイダの WWW サーバなどには、UNIX 系のシステムが使われていることが多い。

   一方、 Windows は基本的にはコンピュータの前に座って、一人で使うシステムである。他のマシンからハードディスクの中身を見たりすることはできるが、それ以上のことをするには特殊なソフトが必要になることが多い。

3. **オープンとクローズド**

   UNIX 系のシステムのうち、特に Linux や FreeBSD は、C 言語などで書かれたソース・コードは、完全にオープンになっている。

   たとえば、 FreeBSD は `http://www.jp.freebsd.org/` が日本の Web サイトである。ここからすべて無料で自由にダウンロードすることができる。ダウンロードしたソース・コードは中を読むこともできるし、自分で自由にいじることもできる。

   また、 Linux や FreeBSD の場合、 C コンパイラをはじめとした数多くのプログラミング言語や便利なソフトウェアが無料で最初から付いていて、わざわざお金をかけずに自由にプログラムを作ることができる。

   一方、 Windows は商品であり、ソース・コードは一般に公開されていない。Windows 用のさまざまなソフトウェアも商品である場合が多い。

第2章　UNIX システムを使ってみよう

2.1　システムの起動、終了、ログイン、ログアウト

2.1.1　システムの起動

B 演習室の PC/AT 互換機には、 Windows XP と Windows NT と FreeBSD が入っている。FreeBSD を起動するには次のようにする。

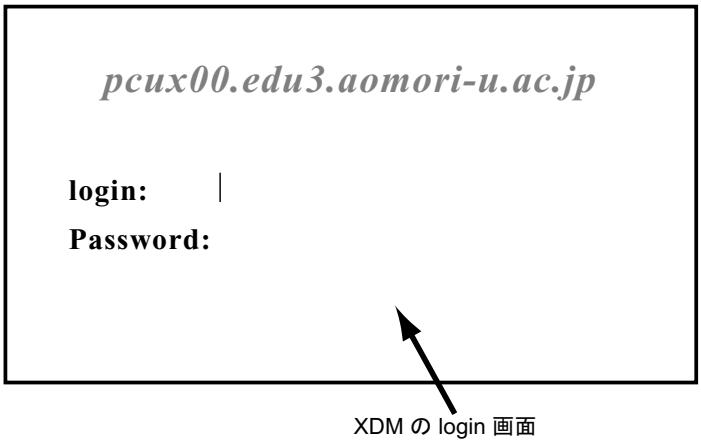
1. 電源スイッチをいれる。
2. OS Loader が起動し、次の図のようになるので、 FreeBSD を選択し、Enter を押す。

3. 「Booting [kernel] in ○ seconds...」と表示されるので、 Enter を押す (下の図では Enter を ↵ と書いている)。これは放っておいても大丈夫である。

```
BTX loader 1.00 BTX version is 1.01
Console: internal video/keyboard
BIOS drive A: is disk0
BIOS drive C: is disk1
BIOS 639kB/392064KB available memory
                                  ⋮
                                中略
                                  ⋮
Hit [Enter] to boot immediately, or any other key for command prompt.
Booting [kernel] in 6 seconds... _↵
```

4. FreeBSD が起動し、 X Window System が立ち上がる。そして、次の図のように XDM[1] の login 画面になる。

[1] X Display Manager



以上で、起動は OK である。

2.1.2　ログイン

ID とパスワードがまず必要

UNIX システムを使うためには、そのシステムの管理者が用意してくれた ID とパスワードが必要になる。

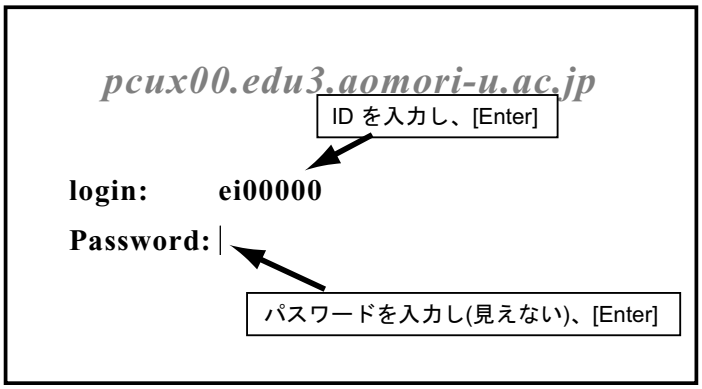
ID は UNIX システムを使うときのユーザの名前である。これは、そのシステムを使うときにずっと使うもので、ユーザが自分で勝手に変えることはできない。ID には、例えば kokubo、 judy、 aichan、 rms のように、名前やニックネーム、頭文字を使ったりすることもある。また、 ei00001、 0198011、 tz0la92 のように、学校の学籍番号や、会社の社員番号や、ランダムな文字列を使うこともある。

パスワードは、たぶんシステムの管理者が用意してくれたものを最初に渡されるだろう。これをユーザは、自分で好きなものに変更して使う。パスワードの変更の仕方は後の方で説明する。

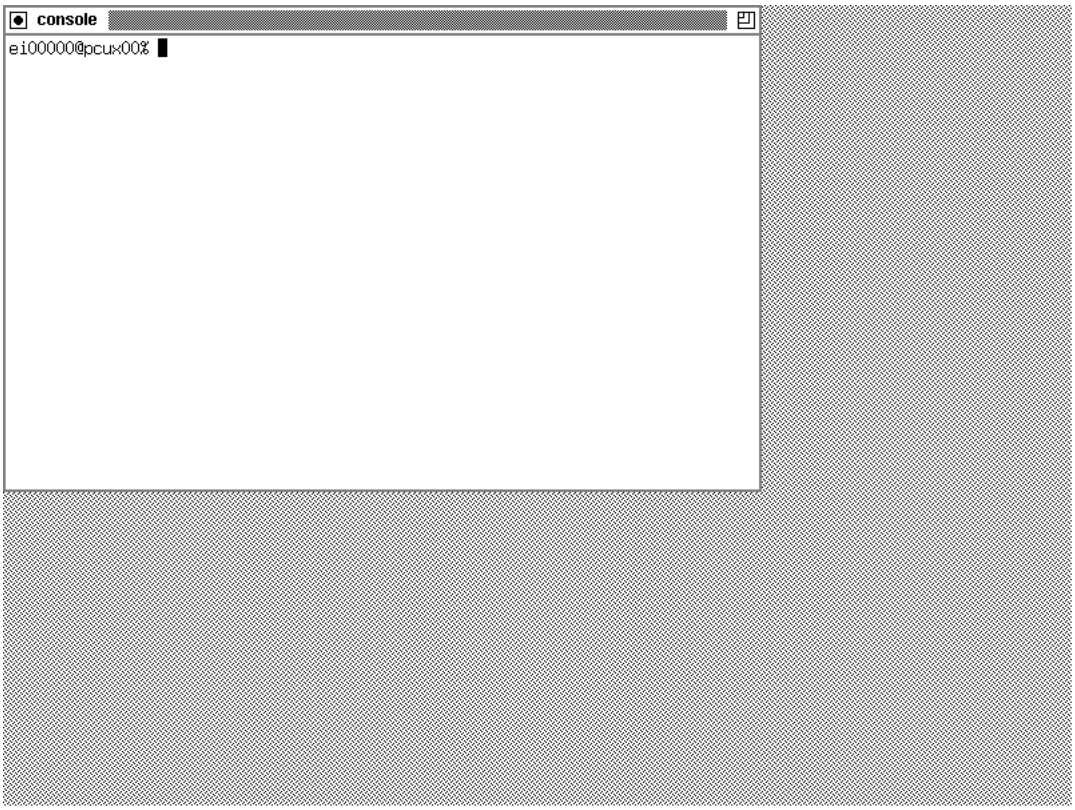
XDM からのログイン

B 演習室のマシンは、 UNIX システムが起動すると、 XDM のログイン画面がでる。この画面からログインするには、次のようにする。

1. 「login:」のところに ID を入力して、 Enter。
2. 「Password:」のところにパスワードを入力して (打った文字は見えない)、 Enter。



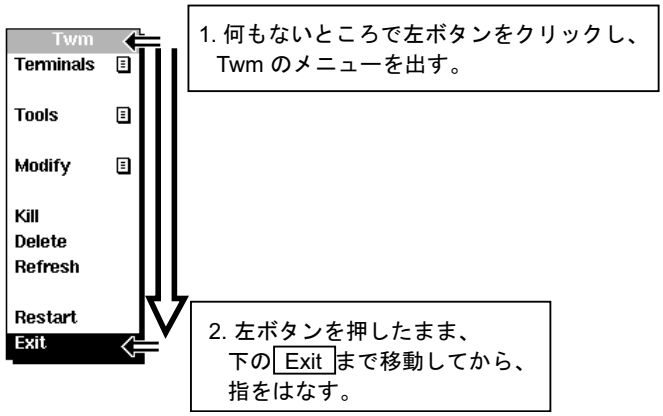
3. login に成功すると、次のような画面になる。失敗した場合は、赤字で「Login incorrect」と表示されるので、ID を入力するところからやりなおす。



2.1.3 ログアウト

UNIX システムからログアウトする方法を説明しよう。

1. 起動しているプログラムを一通り終了する (終了の仕方は後で説明する)。
2. 画面の何もない部分で、左ボタンをクリックする。
3. 「Twm」と書かれたメニューが出るので、左ボタンを押したまま、1 番下の「Exit」までマウスを移動してから、指をはなす。



4. XDM の login 画面になって、ログアウトが完了する。

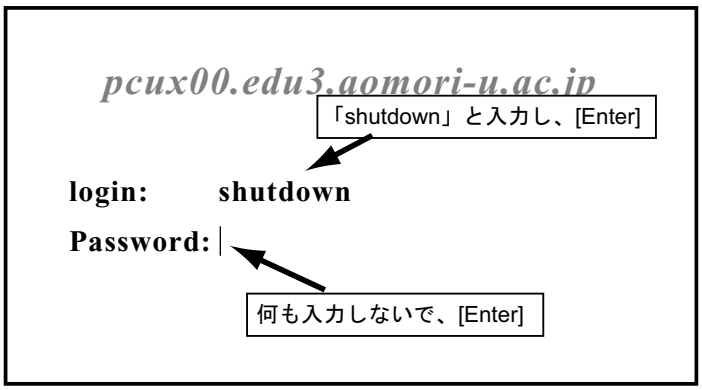
2.1.4 システムの終了

通常、 UNIX システムは、電源を切らないで 24 時間動かしっぱなしにすることが多い。また、普通のユーザは電源を切ることができず、システム管理者だけがこれを行うことができる。

しかし、B 演習室のマシンは、 Windows に切替えて使ったりするために、普通のユーザでも電源を切ることができるように設定されている。ちなみに、ここに書かれているように実行しても、 B 演習室以外のマシンでは終了できないので、注意すること。

では、電源の切り方を説明しよう。この手順通りに実行しないと、システムがクラッシュしたりする可能性があるので注意すること。

1. UNIX システムを使っている場合は、ログアウトして、 XDM の login 画面にする。
2. login: に対して、「shutdown」と入力して Enter する。
3. Password: に対して、何も入力しないで Enter する。



4. 一瞬、間をおいて shutdown が実行される。
5. 次のようなメッセージがでて、システムが終了する。

```
The operating system halted. (オペレーティング・システムは停止しました。)
Please press any key to reboot. (再起動するには何かキーを押してください。)
```

6. 今回は 再起動しない ので、 何もキーを押さず に、電源スイッチを押して、電源を切る。

2.2　パスワードの変更

最初、システム管理者からパスワードが配られると思うが、 UNIX システムでは自分で好きなパスワードに変更できる。ここでは、その方法を紹介しよう。

2.2.1　パスワードの条件

まず、 FreeBSD の場合、パスワードには次のような条件がある。

1. 文字数は 6 〜 128 文字

2. 使える文字は、英、数、記号

3. ただし、大文字だけ、小文字だけのパスワードはダメ。

2.2.2　どんなパスワードがいいか

では、どんなパスワードがいいかというと次のようなものだ。

1. 万一、一瞬他人から見られても、何だかよくわからないもの。
2. 自分では忘れにくい。
3. 大、小文字、数字、記号などが適当に混じっている。

逆に危険なパスワードは、次のようなものだ。

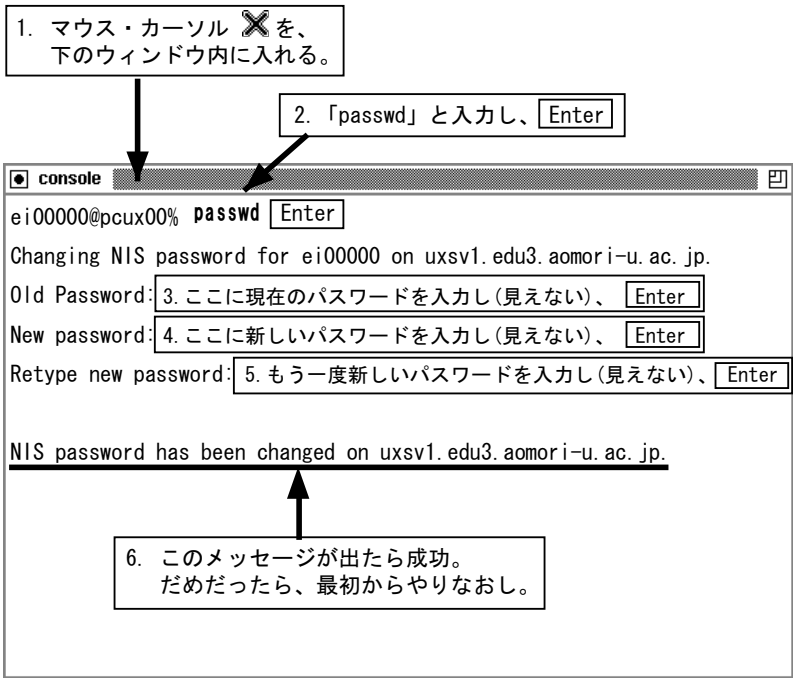
1. 辞書にそのまま載っているような単語。
2. 自分の名前、電話番号、住所など、他人から簡単に想像できるものを そのまま 使う。
3. あまりに長いなど、自分でも覚えられないようなもの。

2.2.3　パスワードの変更

パスワードを決めたら、変更することにする。変更するには、次のようにする。

1. マウスをウィンドウに入れる。
2. 「`passwd`」と入力し、 Enter 。
3. 「Old Password:」に対して、現在のパスワードを入力し、 Enter
4. 「New password:」に対して、新しいパスワードを入力し、 Enter
5. 「Retype new password:」に対して、もう一度新しいパスワードを入力し、 Enter

6. 「NIS password has been changed on uxsv1.edu3.aomori-u.ac.jp.」と出れば、変更に成功。だめなら、 2 番からやり直す。



パスワードの変更に失敗したときのエラー・メッセージ

パスワードの変更に失敗すると、次のようなエラー・メッセージが出る。その意味を説明しよう。

1. 「**password: Sorry.**」

   → 現在のパスワードが間違っていた。

2. 「**Please enter a password at least 6 characters in length**」

   → 新しいパスワードが短すぎた (最低でも 6 文字以上)。

3. 「**Please don't use an all-lower case password. Unusual capitalization, control charactors or digits are suggested.**」

   → パスワードが全部小文字だった。

4. 「**Mismatch: try again, EOF to quit.**」

   → 新しいパスワードとして入力したものが 1 回目と 2 回目で異なっていた。

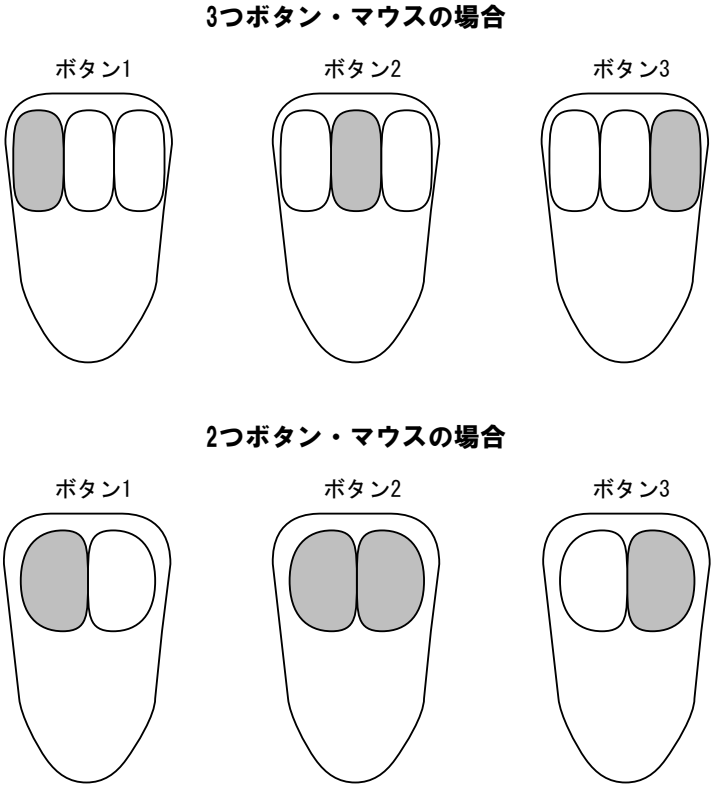
2.3 X Window System 入門

ここでは、 X Window System について、簡単に紹介する。X Windows System の操作は実際にはウィンドウ・マネージャの種類によって異なってくるが、ここでは従来から標準的に用いられている Twm を取り上げることにする。

2.3.1 マウスのボタンの呼び方

まず、最初にマウスのボタンの呼び方を紹介しよう。X Window System を使う UNIX では、普通、3 つボタン・マウスを使う。それぞれのボタンは、左から順番に、ボタン 1、 2、 3 と呼ぶ。

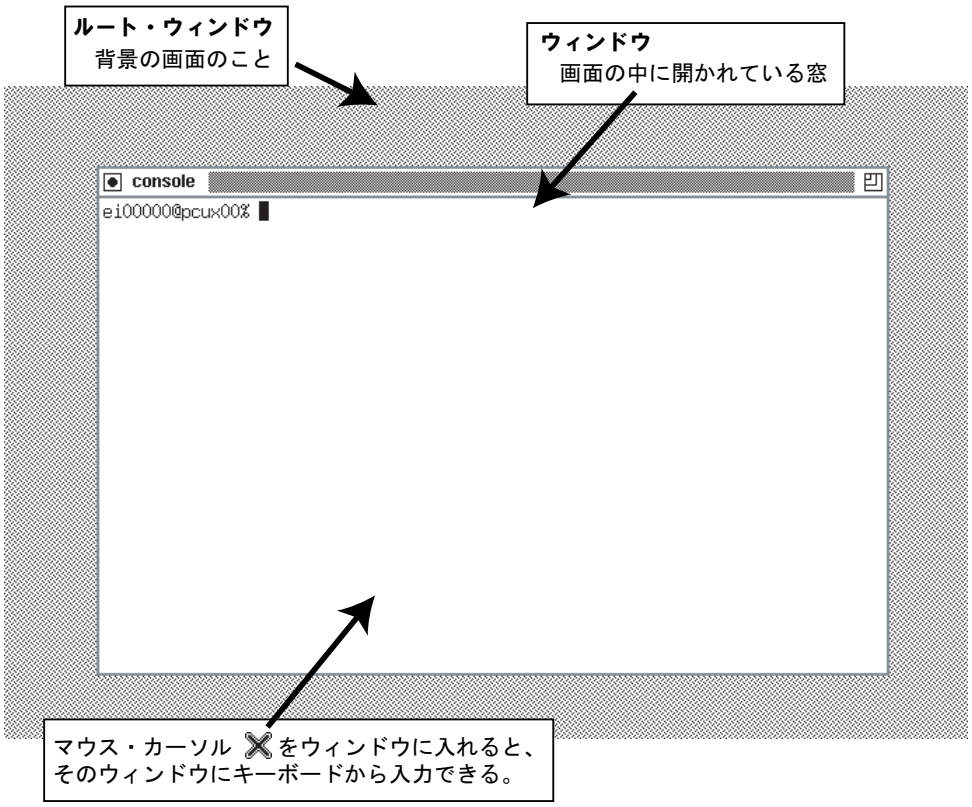
なお、 2 つボタン・マウスを使っているときは、左右のボタンを同時に押すと、ボタン 2 の代わりになる[2]。



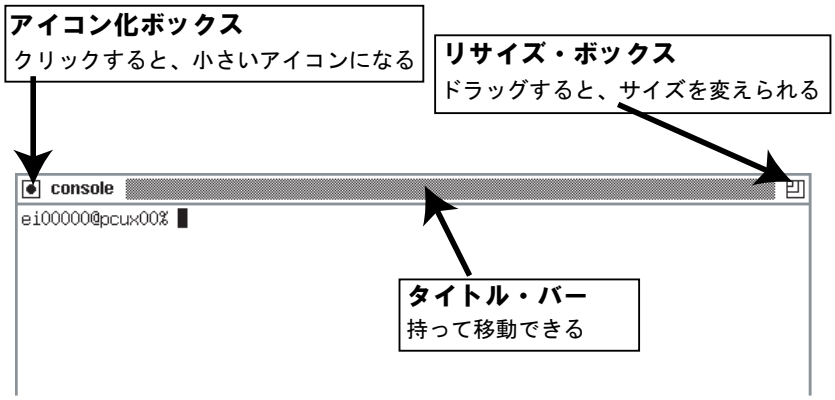
[2]システム管理者がそのように設定しておく必要がある。

2.3.2 画面内の各部の名称

X Window System のデスクトップの名称を紹介しよう。次の図のように、画面の背景を「ルート・ウィンドウ」、画面の中に開かれている様々なプログラムを「ウィンドウ」と言う。いくつかの「ウィンドウ」は、マウス・カーソルを入れると、文字を入力できるようになったりする。



また、各ウィンドウの上部には、「バー」が付いている。「バー」には、持つとウィンドウを移動できる「タイトル・バー」、クリックするとウィンドウがアイコン化される「アイコン化ボックス」、ドラッグするとウィンドウのサイズを変更できる「リサイズ・ボックス」などが付いている。



2.3.3 ウィンドウを開く

ウィンドウを開く方法はいくつかある。その代表的な方法を紹介しよう。

ウィンドウを開くコマンドを入力

まず、文字の入力できるウィンドウにマウス・カーソルを移動する。そして、「**コマンド名** &↵」と入力する。

では実際に、次のコマンドを、いくつか実行してみよう。

- kterm &↵
- xeyes &↵
- xcalc &↵
- xclock &↵
- asclock &↵
- xengine &↵
- tksol &↵
- xmine &↵
- xbill &↵
- oneko &↵
- xearth &↵
- xsoldier &↵

&を最後に付け忘れると

コマンドの最後に「&」を付けずに実行すると、元のウィンドウにタイプしても何も効かなくなる。

```
% kterm↵
(ここにコマンドを入力しても何も効かない)
```

このような場合は、まず「Ctrl+z」と入力し (入力すると「^Z」と表示される)、それ続けて「bg↵」と入力する。すると、再び入力を受け付けるようになる。

```
% kterm↵
^Z
[1]+  Stopped                 kterm
% bg↵
[1]+ kterm &
%(普通にコマンドが打てるようになる)
```

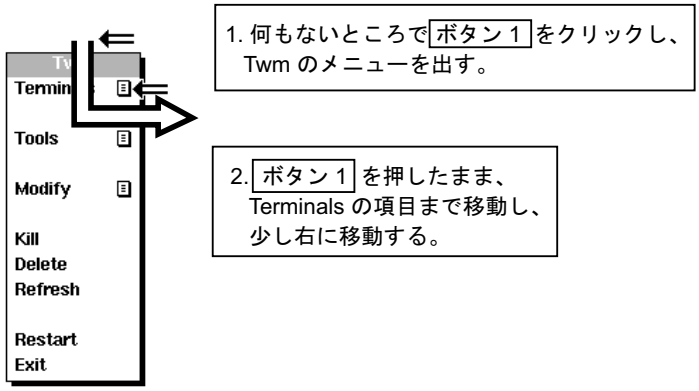
ウィンドウ・マネージャのメニューから選択

ルート・ウィンドウでマウスのボタン 1をクリックするとメニューが出る。このメニューからウィンドウを開くこともできる。

では実際に、次のようにしてみよう。

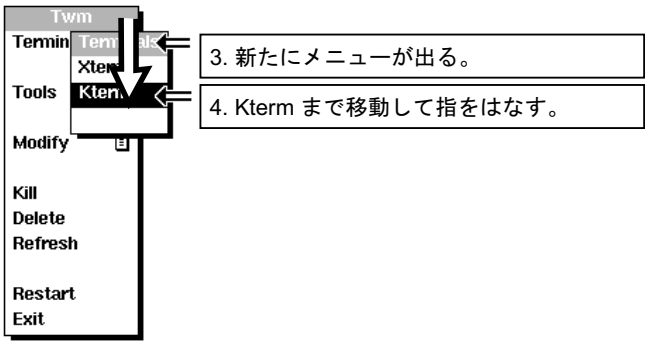
1. ボタン 1 を何もないところでクリックし、Twm のメニューを出す。

2. ボタン 1 を押したまま、「Terminals」まで移動し、少し右に移動する。

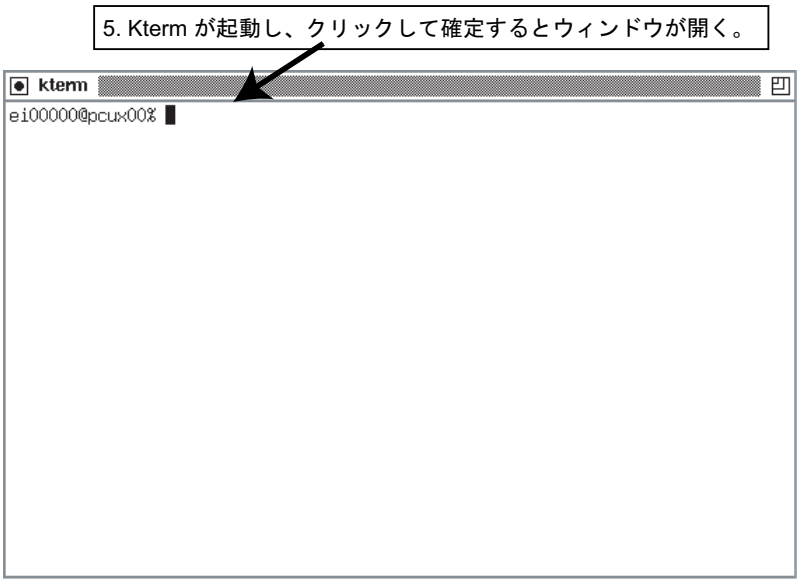


3. 新たにメニューが出る。

4. 「Kterm」まで移動して、指をはなす。



5. マウスで確定してやると、Kterm が起動してウィンドウが開く。



2.3.4 ウィンドウを閉じる

ウィンドウを閉じる方法も何種類かある。

`exit` コマンドを入力

Kterm のように、コマンドが入力できるウィンドウを閉じる場合は、「`exit`」と入力する。

```
ei00000@pcux00% exit↵
```

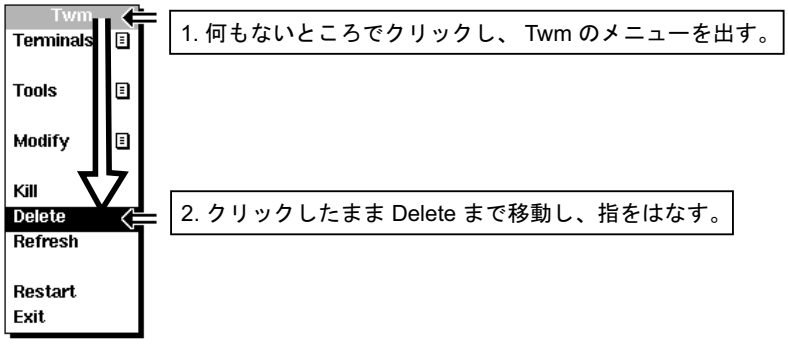
ウィンドウのメニューから exit などを選ぶ

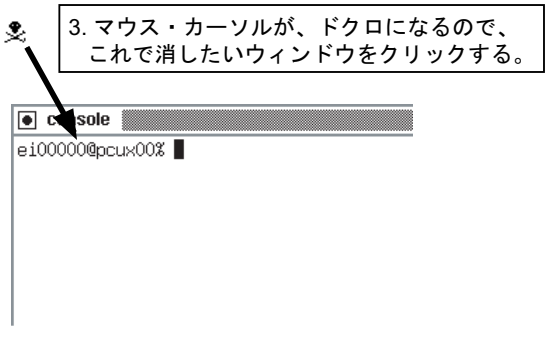
いくつかのウィンドウには、メニューが付いている。このメニューの「file」などの項目から、「exit」などを選ぶと終了できる。

ウィンドウ・マネージャのメニューから「Delete」を選ぶ

上の 2 つの方法で終了できない場合は、Twm のメニューから「Delete」を選ぶと「終了」させられる。なお、「Kill」を選んだ場合は、「強制終了」になる。

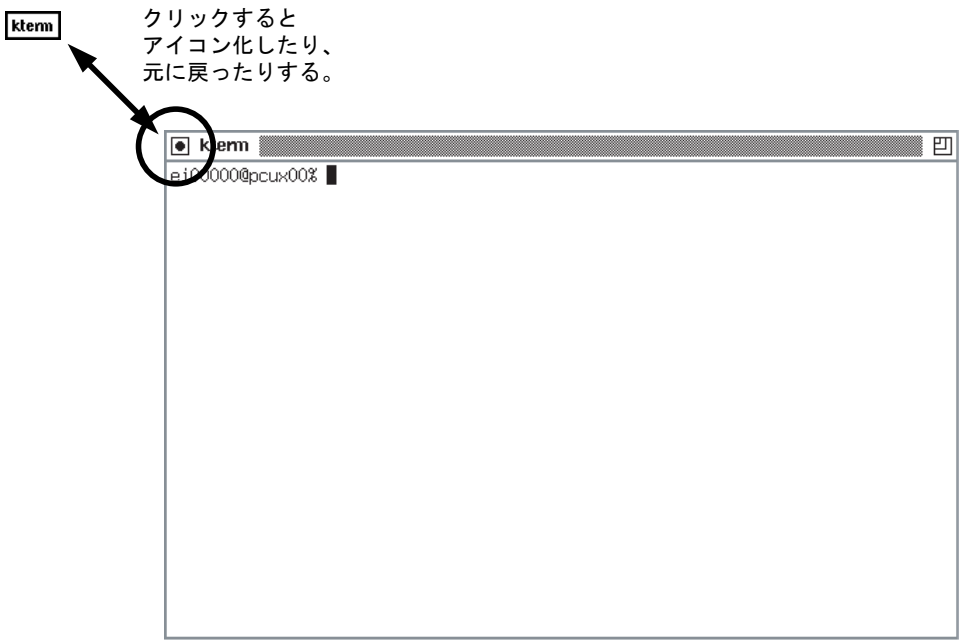
1. 何もないところでクリックし、「Twm」のメニューを出す。
2. クリックしたまま「Delete」まで移動し、指をはなす。
3. ドクロに変わったマウス・カーソルで、閉じたいウィンドウをクリックする。





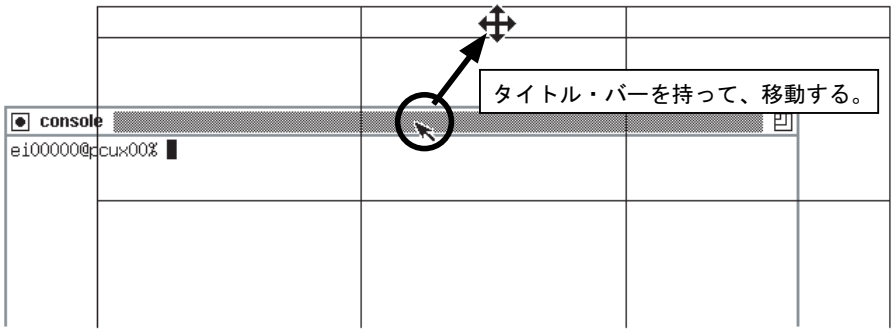
2.3.5 ウィンドウのアイコン化

アイコン化ボックスをクリックすると、ウィンドウがアイコン (小さいマーク) 化される。元に戻すには、アイコンをクリックすればよい。



2.3.6 ウィンドウの移動

ウィンドウのタイトル・バーを持てば、ウィンドウを移動することができる。



2.3.7 ウィンドウの大きさの変更

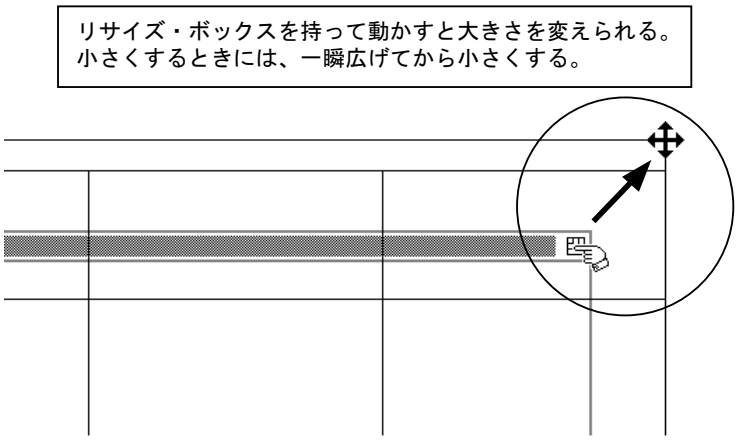
ウィンドウの大きさは、「リサイズ・ボックス」を持って、ドラッグすることで変更できる。

1. 大きくするとき

   単にリサイズ・ボックスをクリックし、そのままドラッグして広げる。

2. 小さくするとき

   大きくするときと同様だが、一旦、少しだけ広げてからでないと、小さくできない。

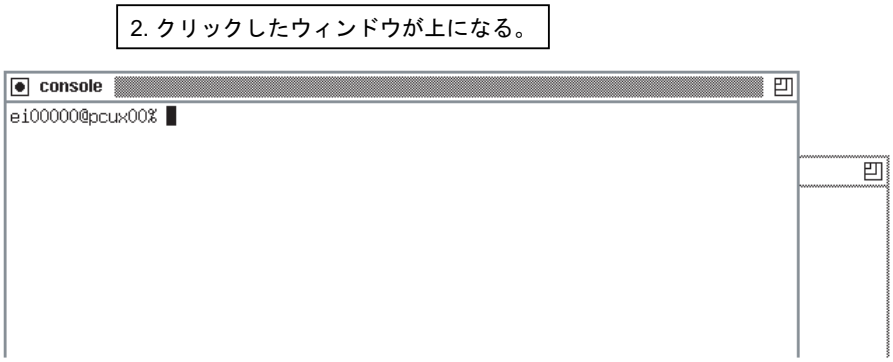


2.3.8 重なったウィンドウを一番上にする

重なったウィンドウを上にする方法は何通りかある。ここでは、いくつか紹介しよう。

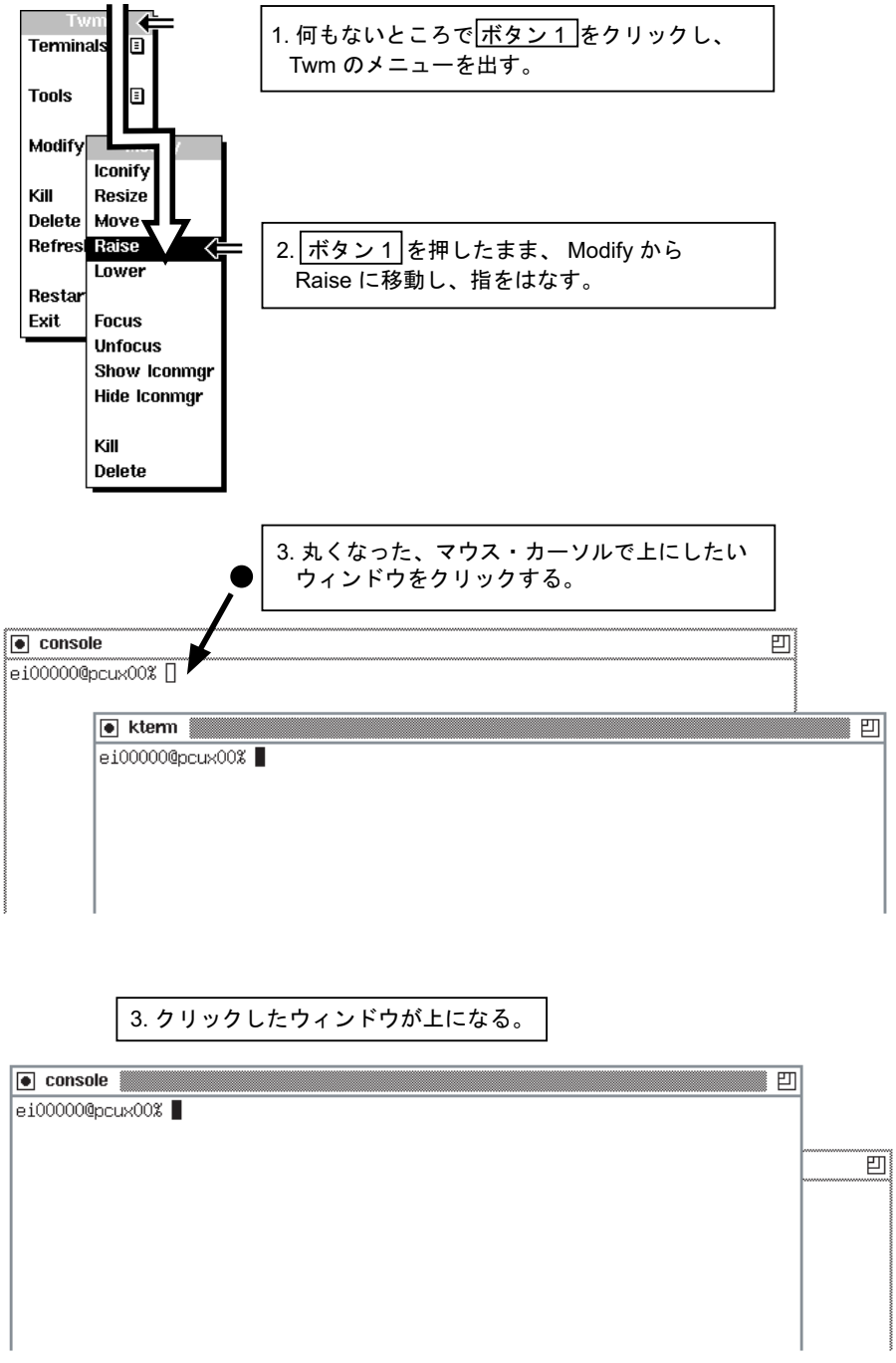
上にしたいウィンドウのタイトル・バーをクリック

下になっているウィンドウのタイトル・バーをクリックすると、一番上に出すことができる。



ウィンドウ・マネージャのメニューを使う

何もないところで ボタン 1 を押して、Twm のメニューを出し、その中の「Modify」から「Raize」を選ぶ。マウス・カーソルが「●」になるので、これで上にしたいウィンドウをクリックする。



第3章　UNIX コマンド入門

UNIX のコマンドを使ってみよう

UNIX システムでは、キーボードからコマンドを打ち込んで、プログラムを実行することが多い。コマンドに慣れるために、この章では UNIX の簡単なコマンドをいくつか紹介しよう。

コマンドは、英語を省略したものが多い。英語と言っても、日本語の会話にもよく出てくるような、簡単なものがほとんどである。UNIX システムの上で C を使ってプログラミングをするくらいであれば、極端な話 10 個くらいのコマンドを知っていればなんとかなる。

とは言え、コマンドは一度覚えてしまえば、多くの UNIX システムで、ほとんど同じように使うことができる。また、たくさんのコマンドやテクニックを知っていると、簡単な手順で実にいろいろなことができるようになる。この本に出てくるコマンドは、全部おぼえてしまって損はないだろう。

3.1　カレンダーの表示: `cal` コマンド

3.1.1　`cal` コマンドを使ってみよう

UNIX にログインすると、 B 演習室では「`ei00000@pcux00%`」のような文字が表示されている。これを「プロンプト – prompt(入力促進記号)」と言う。プロンプトが表示されているのは、これに続けてコマンドが入力できるというお知らせである。

なお、この本では、プロンプトは「`%`」と書いておくので注意して欲しい。プロンプトをどんなものにするかは、自分で自由に設定することができる。

では、プロンプト「`%`」に対して、次のように「`cal`↵」と打ち込んでみよう。UNIX システムでは、大文字と小文字は区別 されるので、このコマンドは全部小文字で打つこと。

なお、この本では、ユーザが入力する文字は「**太字**」で、そうでない文字は「普通の書体」で表現している。

```
% cal↵
```

すると、次のように、今月[1]のカレンダーが表示される。さっき、 UNIX システムのコマンドは英語を省略したものが多いと言ったが、「`cal`」は「**cal**endar (カレンダー)」を省略したものだ。

[1] この本を書いたのは 2002 年の 9 月でした。

```
% cal↵
      9月 2002
日 月 火 水 木 金 土
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30

%
```

3.1.2 コマンドを打ったけどうまくいかない??

コマンドを入力するときにありがちなトラブルとその解決方法を紹介しよう。

- **打ち間違えた文字を消すには?**

  文字を打ち間違えたときは、 ↵を押す前であれば、Delete や Back Space で消すことができる。

- **「Command not found.」って言われたけど?**

  「cal」を、「cak」のように**ミス・タイプ**したりすると、次のように「Command not found.」(コマンドが見つからないよ。) というメッセージがでる。

```
% cak↵
cak: Command not found. (コマンドが見つからないよ。)
%
```

- **コマンドを打ったら暴走した?**

  何かのキーを間違って押したり、ミス・タイプすると、運が悪い場合、プログラムが暴走してプロンプトが出て来なくことがある。そんなときは、 Ctrl+c (Ctrl を押しながら c) と打てば、多くの場合止めることができる。

3.1.3 cal コマンドの引き数

もう少し高度な cal コマンドの使い方を紹介しよう。

cal コマンドでは、好きな月のカレンダーを表示させることができる。「cal 4 2000↵」というふうに月と年を指定して、実行してみよう。

```
% cal 4 2000↵
      4月 2000
日 月 火 水 木 金 土
                   1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30
%
```

このように、 2000 年の 4 月のカレンダーが表示される。

今度は、「月」を省略して、「cal 2000↵」と打ってみよう。

```
% cal 2000↵
                              2000

       1月                   2月                   3月
日 月 火 水 木 金 土  日 月 火 水 木 金 土  日 月 火 水 木 金 土
                   1            1  2  3  4  5            1  2  3  4
 2  3  4  5  6  7  8   6  7  8  9 10 11 12   5  6  7  8  9 10 11
 9 10 11 12 13 14 15  13 14 15 16 17 18 19  12 13 14 15 16 17 18
16 17 18 19 20 21 22  20 21 22 23 24 25 26  19 20 21 22 23 24 25
23 24 25 26 27 28 29  27 28 29              26 27 28 29 30 31
30 31

                                  ⋮
                               [中略]
                                  ⋮

       10月                  11月                  12月
日 月 火 水 木 金 土  日 月 火 水 木 金 土  日 月 火 水 木 金 土
 1  2  3  4  5  6  7            1  2  3  4                  1  2
 8  9 10 11 12 13 14   5  6  7  8  9 10 11   3  4  5  6  7  8  9
15 16 17 18 19 20 21  12 13 14 15 16 17 18  10 11 12 13 14 15 16
22 23 24 25 26 27 28  19 20 21 22 23 24 25  17 18 19 20 21 22 23
29 30 31              26 27 28 29 30        24 25 26 27 28 29 30
                                            31
%
```

このように 2000 年のカレンダーが表示されるはずだ。ただし、画面が小さい場合、最初の部分は流れてしまって、後ろの方の部分だけが表示される[2]。

ひとまず、ここまでのところをまとめよう。cal コマンドは次のように使う。

cal 「月」 「年」

[2]長い出力を 1 画面ずつ表示させるには、ページャーというプログラムを使う。ページャーについては、後ろの章で紹介する。

ここで、「年」は西暦で指定する。「月」を省略すると、「年」で指定した年の 1 年分のカレンダーが出る。「月」も「年」も省略すると、今月のカレンダーが出る。

この cal コマンドの「月」や「年」のように、コマンドの後ろに付けて細かい指定をするのに使うものを、**「引き数 – argument」**と言う。これはプログラミングの本や、マニュアルなどでもよく使う用語なので、おぼえておこう。

☆練習☆

1. 自分の生まれた日が何曜日だったか調べてみよう。
2. 友だちの生まれた日が何曜日だったか調べてみよう。

ちなみに、昭和 60 年が西暦 1985 年といったように、昭和の年数に 25 を足すと西暦の下 2 ケタになる。また、平成の場合は 88 を足す。

3.2 その他の簡単なコマンド

3.2.1 日時の表示: date コマンド

date は、現在の時刻を表示するコマンドだ。「date↵」と打ってみよう。なお、今後は ↵ を一々書かないで省略する。

```
% date
2000 年 10 月 01 日 日曜日 03 時 28 分 49 秒 JST
%
```

3.2.2 login している人のリスト: who コマンド

who は、現在、そのマシンにログインしている人のリストを表示するコマンドだ。UNIX システムは、同時に何人もの人が使うことができるので、このようなコマンドがある。「who」と打ってみよう。

```
% who
kokubo   ttyp0     9/01 01:06   (:0.0)
kokubo   ttyp1     9/01 01:05   (:0.0)
tomoda   ttyp2     9/01 02:31   (172.31.1.1)
%
```

これが現在ログインしている人のリストである。B 演習室の場合は、「ei00000」のような自分の ID が並ぶことだろう。

それから、 who の仲間に、 whoami というコマンドがある。これは、次のように「whoami」と打つと現在使用中のユーザの ID を表示する。

```
% whoami
kokubo
%
```

3.2.3 今日は何の日?: today コマンド

today[3]は、今日がどんな日かを表示するコマンドだ。「today」と打ってみよう。

```
% today
こんばんは。
きょうは、平成 12 年 10 月 1 日（日曜日）です。
旧暦では、平成 12 年 9 月 4 日（赤口）[中潮] です。
干支では、庚辰（かのえたつ）の年、壬辰（みずのえたつ）の日です。
九星は、二黒 です。
              ⋮
          [中略]
              ⋮
AD1948/10/01  110 番設置
AD1949/10/01  中華人民共和国誕生
AD1964/10/01  東海道新幹線開業
%
```

3.2.4 おみくじ: fortune コマンド

fortune[4]は、ランダムにおみくじ、ことわざ、格言、小説からの引用などを表示するコマンドだ。「fortune」と打ってみよう。

```
% fortune
Mother is the invention of necessity. (必要は発明の母。)
%
```

[3] このコマンドはオプションなので、入っていないこともある。

[4] フォーチュン・クッキーというおみくじ入りのお菓子がある。それから来たコマンド。なお、このコマンドもオプションなので、入っていないこともある。

3.3　エラーからの回復方法

人間は誰でも失敗することがある。だから、コマンドを入力するときにも、いろいろ失敗するだろう。それらの対処方法を改めて詳しく紹介しておこう。

1. **打ち間違えた文字を消したい**

   打ち間違えた文字は、 ↵を押す前なら、Delete や Back Space で消すことができる。

   また、コマンドを途中まで打ったが、その行を全部キャンセルしたくなったら、Ctrl+c と入力すればよい。

2. **「Command not found.」(コマンドが見つからないよ。) と言われた**

   これは、コマンドをタイプミスしたときにでるメッセージである。よく見直してみよう。

3. **コマンドを打ったら暴走した**

   プログラムが暴走した場合には、 Ctrl+c と打てば、たいてい止められる。

4. **コマンドを打っても効かない**

   たとえば X Window System でウィンドウを開くコマンドの後ろに「&」を付けないで実行すると、コマンドを実行したウィンドウには文字が入力できなくなる。このときは、 Ctrl+z と入力して、続けて「bg」というコマンドを打てばよい。

5. **コマンドを実行したが止め方がわからない**

   ユーザの入力を対話的に処理するプログラムの多くは、Ctrl+d で、終了できる。また、ものによっては、「quit」、「exit」、「bye」などと入力すると、終了できることがある。

   例を挙げよう。bc という電卓のプログラムがあって、「bc」と打つと起動できる。

```
% bc
bc 1.03 (Nov 2, 1994)
Copyright (C) 1991, 1992, 1993, 1994 Free Software Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type ‘warranty’.
```

   後は、ここに数式を打つと、計算結果を表示してくれる。例えば、「1 + 2 + 3」なら、

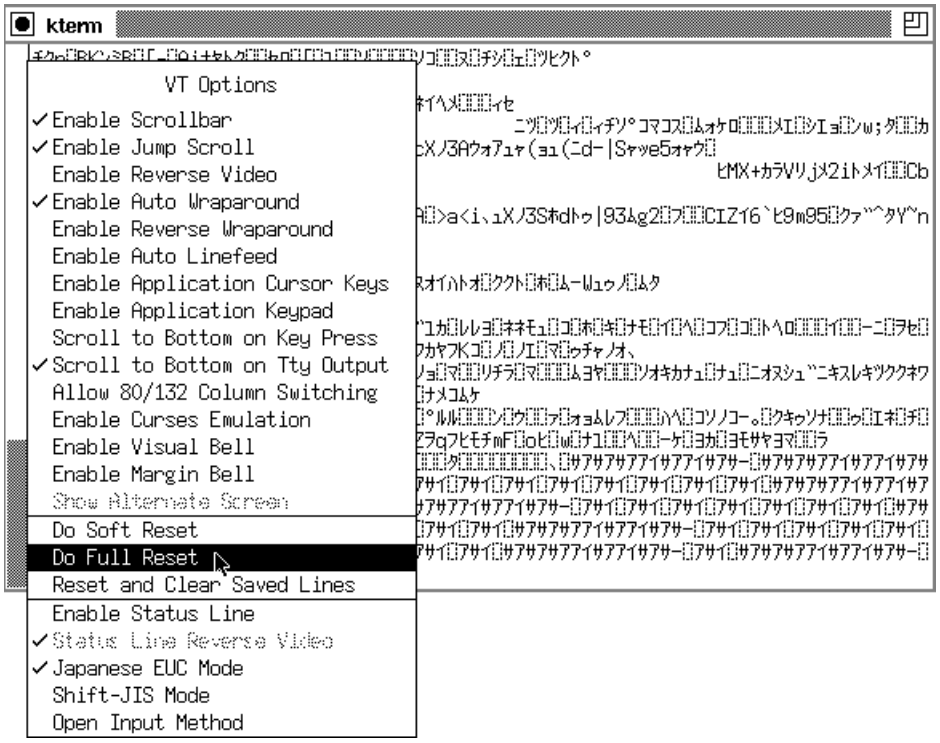
```
% bc
bc 1.03 (Nov 2, 1994)
Copyright (C) 1991, 1992, 1993, 1994 Free Software Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type ‘warranty’.
1+2+3
6
```

このコマンドの実行を終了させるには、 Ctrl +d、または「quit」と入力する。

6. **文字化けが起こった**

X Window System で Kterm や xterm を使っていると、文字化けが起こることがある。

このときは、 Kterm などのウィンドウで、 Ctrl + ボタン 2 で「VT Options」というメニューを出すことができる。このメニューから、「Do Full Reset」を選ぶと、画面の表示にリセットをかけて直すことができる。



また、 Windows などから、 telnet などを使って入っている場合には、その telnet などのソフト自身で端末のリセットを行なう。

この章で紹介したコマンド

いろいろなコマンド

cal : カレンダーの表示

使い方: cal 「月」 「年」

date : 日時の表示

使い方: date

who : ユーザのリストの表示

使い方: who、whoami

today : 今日がどういう日かを表示

使い方: today

fortune : おみくじの表示

使い方: fortune

第4章　ファイル操作コマンド

ファイルの操作はすべての基本

データを保存したり、コピーしたりといったファイルの操作は、コンピュータの操作の基本中の基本である。ファイル操作の腕が、コンピュータ・ライフの明暗を分けると言ってもいい。

この章では、コマンドの出力をファイルに入れたり、どんなファイルがあるのかリストを出したり、ファイルをコピーしたり、消したりする方法を紹介しよう。

4.1　コマンドの実行結果をファイルに入れる: リダイレクト

前の章で紹介したように、 cal コマンドを実行すると、次のように今月のカレンダーを表示する。

```
% cal
      10月 2000
日 月 火 水 木 金 土
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31

%
```

UNIX システムでは、コマンドの実行結果を、簡単にファイルに入れることができる。次のようにして、 cal の実行結果を、「kongetsu」というファイルに入れてみよう。

```
% cal > kongetsu
%
```

すると、今回は何も表示されなかったはずだ。

単に cal と打ったときに画面に表示されていたものは、今回は「kongetsu」という名前のファイルに入ったのだ。

つまり以下のようにすると、ファイルにコマンドの実行結果を入れることができるのである。

「コマンド」 > 「ファイル名」

このような操作を UNIX システムでは、リダイレクト[1]という。

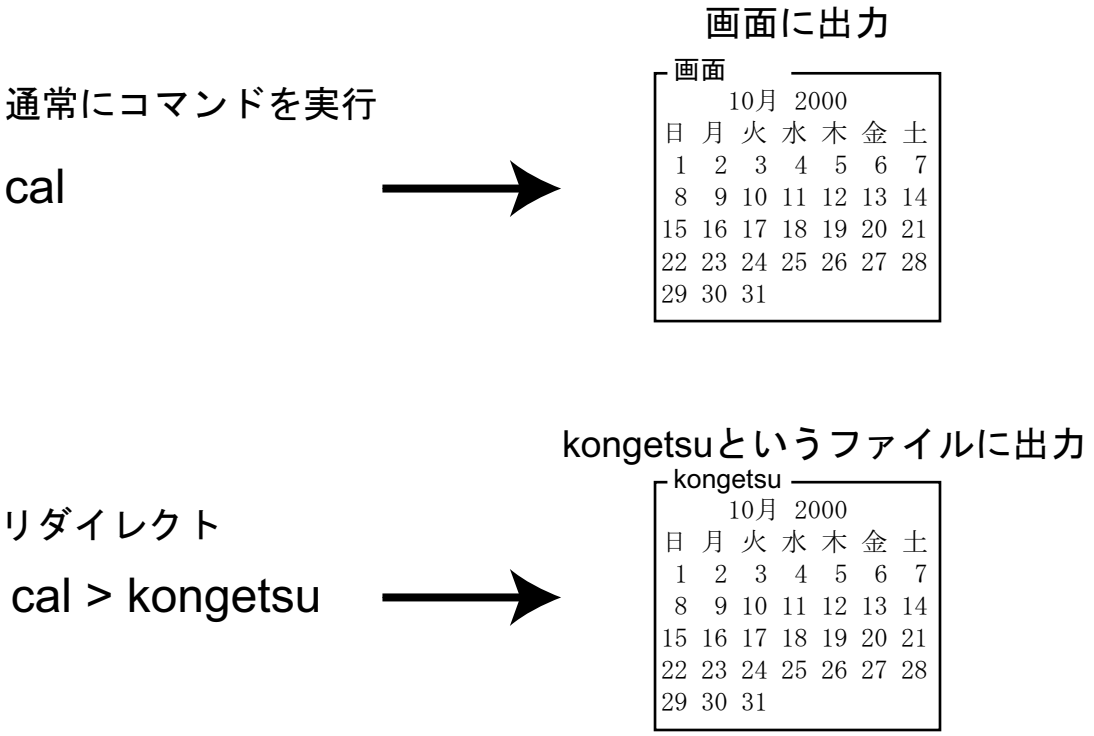


図 4.1: リダイレクト

4.2　ファイル名について

ところで、この例では、ファイル名を「kongetsu」にしたが、別にこれは何でもいい。たとえば、「thismonth」(今月という意味) などでも OK である。。

UNIX システムでよく使われるファイルの名前は、「Address」、「exam1.c」、「gcc-2.8.1」、「gzip-1.2.4.tar.gz」などのように、英数の文字と「-」や、「.」などを組み合わせたものだ。

なお、ファイル名には、使えないわけではないが、使わない方がいい文字がいくつかある。たとえば、「*」や、「?」などの特殊な記号である[2]。また、ファイル名の最初の文字を「-」にするもやめた方がいいだろう[3]。それから、ファイル名の最初の文字を「.」にすると、隠しファイル[4]になるので注意すること。

---

[1] redirect – 「出力する向きを変える」。画面に向かって出力していたものを、ファイルに向かって出力するように、向きを変えたという意味。

[2] ワイルド・カードで使う文字は、使わない方が無難である。ワイルド・カードについては、後ろの章で説明する。

[3] コマンドのオプションと混同される危険がある。コマンドのオプションについては、後ろの章で説明する。

[4] これも後ろの章で説明するが、「.」で始まるファイルは、各種の設定ファイルである。これを書き換えると、ユーザの環境をカスタマイズできる。

4.3 ファイルのリストを表示する: ls

どのようなファイルがあるのかリストを表示するには、 ls コマンドを使う。ls は、「リスト – **lis**t」という意味だ。では、実行してみよう。

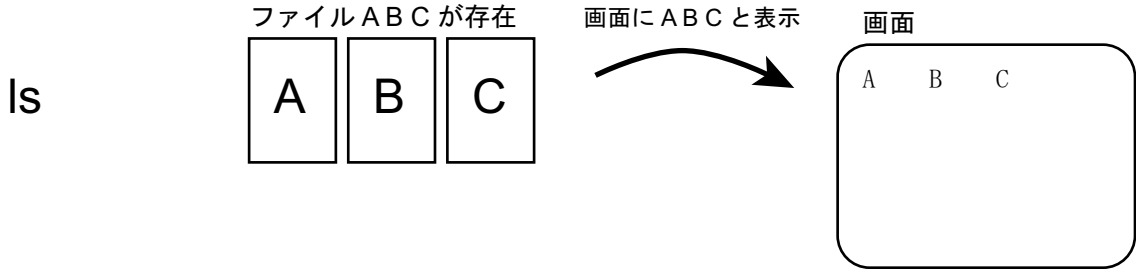
```
% ls
kongetsu
%
```

さっき作った「kongetsu」というファイルがあることがわかる。

つまり以下のように実行すると、ファイルのリストを表示することができるのである。

```
ls
```

ls はファイルのリストを表示する



4.4　ファイルの中身を表示: cat

cat コマンドを使うとファイルの中身を表示することができる。次のようにして実行してみよう。

```
% cat kongetsu
      10月 2000
日 月 火 水 木 金 土
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31

%
```

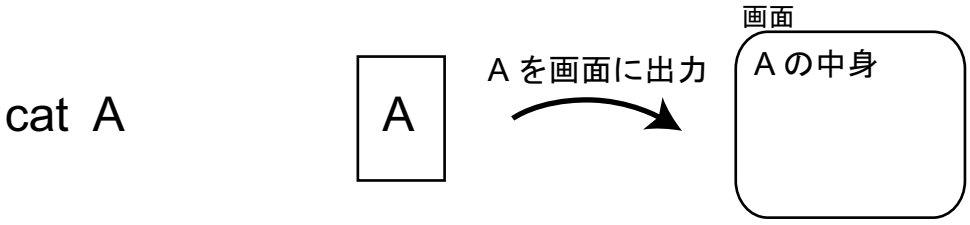
cat は、あまり使わない英語だが、「(ファイルの) 連結表示 – con**cat**nete」から来ている。

つまり、以下のようにしてやるとファイルの中身を画面に表示することができる。

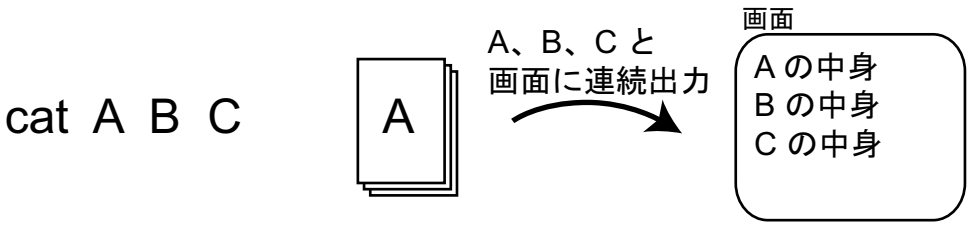
cat 「ファイル名」

ちなみに、どの辺が「**連結**表示」かを説明しよう。cat コマンドの引数には、ファイル名は 1 つだけではなく、「cat 「ファイル名 1」 「ファイル名 2」 「ファイル名 3」 … 」という具合に、複数のファイル名を指定できる。このようにすると、「ファイル名 1」、「ファイル名 2」、「ファイル名 3」、… という順番にファイルの中身が (連結して) 表示される。

cat は画面にファイルの中身を表示する



ファイルを複数指定すると、ファイルの中身を順番に連続表示する



ファイルをたくさん作ってみよう

前の章で紹介したように、たとえば 2000 年のカレンダーを表示するには、「cal 2000」である。

今年のカレンダーを「kotoshi」というファイルに入れてみよう。以下の「2000」の部分には、現在の年号を入れる。

```
% cal 2000 > kotoshi
%
```

ls コマンドで、「kotoshi」ができたか、確かめてみよう。

```
% ls
kongetsu        kotoshi
%
```

確かにできている。次は、今年の 4 月のカレンダーも、同じように「4gatsu」というファイルに入れてみよう。同様に「2000」の部分を現在の年号にして欲しい。

```
% cal 4 2000 > 4gatsu
%
```

「ls」と打つと、確かに「4gatsu」もできていることがわかる。

```
% ls
4gatsu          kongetsu        kotoshi
%
```

☆練習☆

1. 自分が生まれた月のカレンダーを「tanjoubi」というファイルに入れてみよう。
2. ls でファイルができていることを確認しよう。
3. cat でファイルの中身を表示してみよう。

4.5　ファイルのコピー: cp

ファイルをコピーするには、 cp コマンドを使う。cp は「**c**o**p**y – コピー」という意味だ。

「kotoshi」というファイルをコピーして、「thismonth」(今月という意味) というファイルを作るには、次のように打つ。

```
% cp kongetsu thismonth
%
```

ls と cat で確かめてみよう。

```
% ls
4gatsu          kongetsu        kotoshi         tanjoubi        thismonth
% cat thismonth
      10月 2000
日 月 火 水 木 金 土
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31

%
```

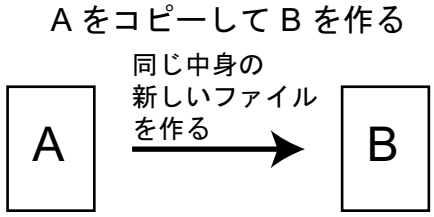
確かに、「thismonth」というファイルができていて、「kongetsu」の中身がコピーされていることがわかる。

まとめると、ファイルをコピーするには、次のようにすればいい。

cp 「コピー元のファイル名」 「コピー先のファイル名」

cp はファイルをコピーする

cp A B



4.6 ファイルの消去: rm

ファイルを消すには、 rm を使う。rm は、「**rem**ove – 消去」を短くしたものだ。
次のようにして、「thismonth」を消してみよう。

```
% rm thismonth
%
```

ls で消えたか確認してみよう。

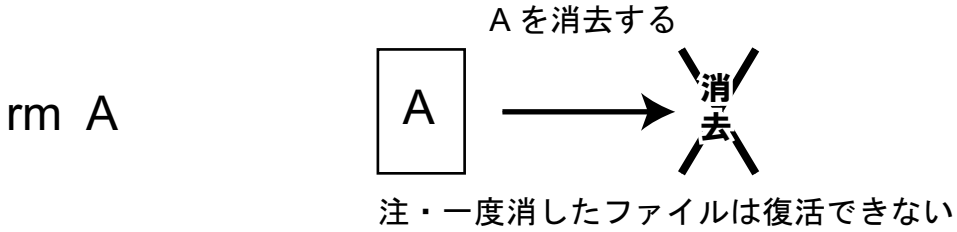
```
% ls
4gatsu          kongetsu        kotoshi         tanjoubi
%
```

確かに消えていることがわかる。

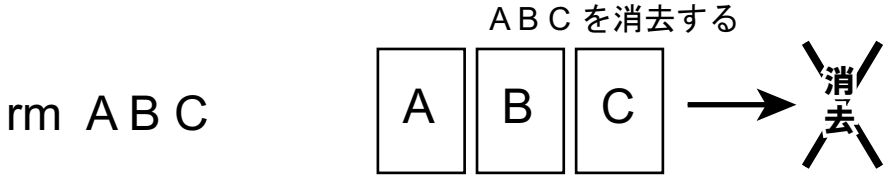
なお、 UNIX システムでは、一般的に消してしまったファイルを、元に戻すことはできないので、rm を実行するときには注意すること。
まとめると、ファイルを消去するには、次のようにすればいい。

rm 「消したいファイル名」

rm はファイルを消去する



ファイル名は複数個指定できる



4.7　ファイルの名前変更: mv

ファイルの名前を変更するには、 mv を使う。mv は、「**m**o**v**e – 移動」を短くしたものだ。

こんな名前が付いているのは、 UNIX システムでは、ファイルの名前を変更することは、ファイルを別の名前に移動することと同じ だと考えるからだ。

では、次のようにして、「kongetsu」を「thismonth」というファイル名に変えてみよう。

```
% mv kongetsu thismonth
%
```

ls で名前が変わっているか、確認してみよう。

```
% ls
4gatsu          kotoshi         tanjoubi        thismonth
%
```

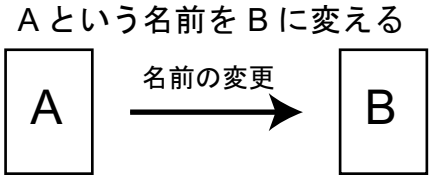
「kongetsu」がなくなって、「thismonth」ができていることがわかる。

まとめると、ファイルの名前を変えるには、次のようにすればいい。

mv 「変更前のファイル名」 「変更後のファイル名」

mv はファイル名を変更する

mv A B



なお、ここでは紹介しないが、 mv でファイルを移動することもできる。

☆練習☆

1. 「kotoshi」をコピーして、「thisyear」というファイルを作ってみよう。
2. 「thisyear」ができているか、 ls で確かめてみよう。
3. 「thisyear」を消してみよう。
4. 「thisyear」が消えたか、 ls で確かめてみよう。
5. 「kotoshi」の名前を、たとえば「2000nen」のように実際の年号に変えてみよう。
6. 名前が変わっているか、 ls で確かめてみよう。

4.8 ファイル操作上の注意

UNIX システムでは、ファイルのコピーや移動、削除、それからリダイレクトなどを行う場合、慣れないうちは誤って上書きしたり削除してしまう ことがある。

というのも、ファイルを誤ってコピーして上書きしそうな場合、Windows では本当に上書きしてよいかという確認のメッセージが出るが、UNIX 系のシステムでは出ないからである。

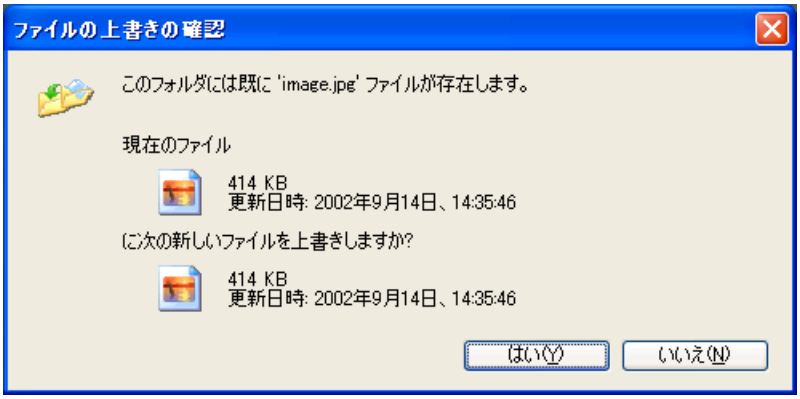
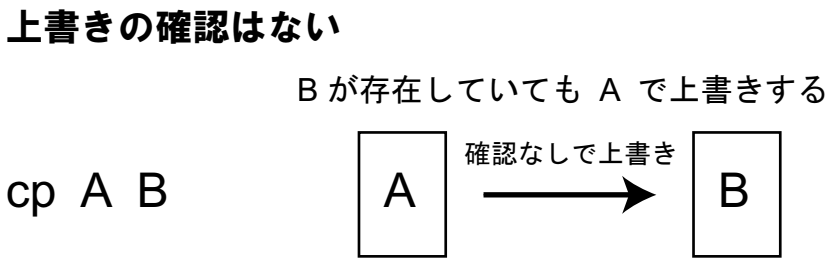


図 4.2: Windows XP の「ファイルの上書きの確認」メッセージ

たとえば、「A」と「B」というファイルが存在していたとしよう。このとき、「cp A B」を実行すると、何もメッセージが出ないままに、平気で「A」によって「B」は上書きされてしまうのである。これは、 cp に限らず mv などでも同様である。



上書きしそうな場合に確認をするようにするには、「cp」の代わりに「cp -i」を使う。

では、試してみよう。まず準備として、「thismonth」を「kongetsu」にコピーする。

```
% cp thismonth kongetsu
%
```

では上書きコピーを実行しようとしてみよう。ただし、今回は「cp -i」を使う。すると、下のように、「overwrite kongetsu? (kongetsu を上書きしますか？)」と聞かれる。「y」と答えると上書きを行い、「n」なら上書きを行わない。

```
% cp -i thismonth kongetsu
overwrite kongetsu? (y/n [n]) n
not overwritten
%
```

なお、「mv -i」などでも同様である。

この章で紹介したコマンド

ファイル操作コマンド

ls : ファイルのリストの表示
　　使い方: ls

cat : ファイルの中身を表示
　　使い方: cat 「ファイル名」

cp : ファイルのコピー
　　使い方: cp 「コピー元のファイル名」 「コピー先のファイル名」

rm : ファイルの消去
　　使い方: rm 「ファイル名」

mv : ファイル名の変更
　　使い方: mv 「変更前のファイル名」 「変更後のファイル名」

第5章　ページャーとマニュアル

1 画面ずつ表示させるには

大きなファイルを cat コマンドなどで表示させようとすると、画面が流れてしまい、ファイルの先頭の方を見ることができない。1 ページ分ずつ画面に表示させるには、「Pager – ページャー」と呼ばれるコマンドを使う。

また、 UNIX システムには、オンライン・マニュアルが付いていて、コマンドや、 C 言語の関数などの詳しい使い方を調べることができる。このオンライン・マニュアルは、ページャーを使って表示される。

この章では、ページャーの使い方と、マニュアルの読み方を紹介しよう。

5.1　1 画面ずつ表示する: ページャー jless

5.1.1　ページャーとは

前の章で説明したように「cal 2000 > 2000nen」とすると、2000 年のカレンダーを「2000nen」というファイルに入れることができる。

小さな画面で、 cat を使って、このファイルの中身を見ようとすると、最初の方が流れてしまって見えないことがある。こんな時は、ページャーを使うと、 1 画面分ずつ表示することができる。「ページャー – pager」という名前は、長いファイルや実行結果を、1 ページずつ区切って表示するところからきている。

主なページャーには、 more と less がある。ここでは、日本語化された less である、 jless を紹介する。

5.1.2　jless を使ってファイルを表示しよう

最初に大きなファイルを用意しよう。まず、次のようにして、 2000 年から 2002 年までの 3 年分のカレンダーを、「2000nen」〜「2002nen」というファイルにそれぞれいれてみよう。

```
% cal 2000 > 2000nen
% cal 2001 > 2001nen
% cal 2002 > 2002nen
%
```

それから、次のように cat の連結機能を使って、この 3 つのファイルを「3years」という一つのファイルに入れよう。

```
% cat 2000nen 2001nen 2002nen > 3years
%
```

こうして作った「3years」は、102 行くらいある大きなファイルになっている。これを、 jless というページャーを使って表示してみよう。

ファイルを表示するとき、 jless は次のように使う。

jless 「ファイル名」

「jless 3years」と入力してみよう。

```
% jless 3ears
```

すると、下のように「3years」の最初の 1 ページ分が表示される。

次のページを見るには スペース を押す。また、 jless を終了するには「q」である。

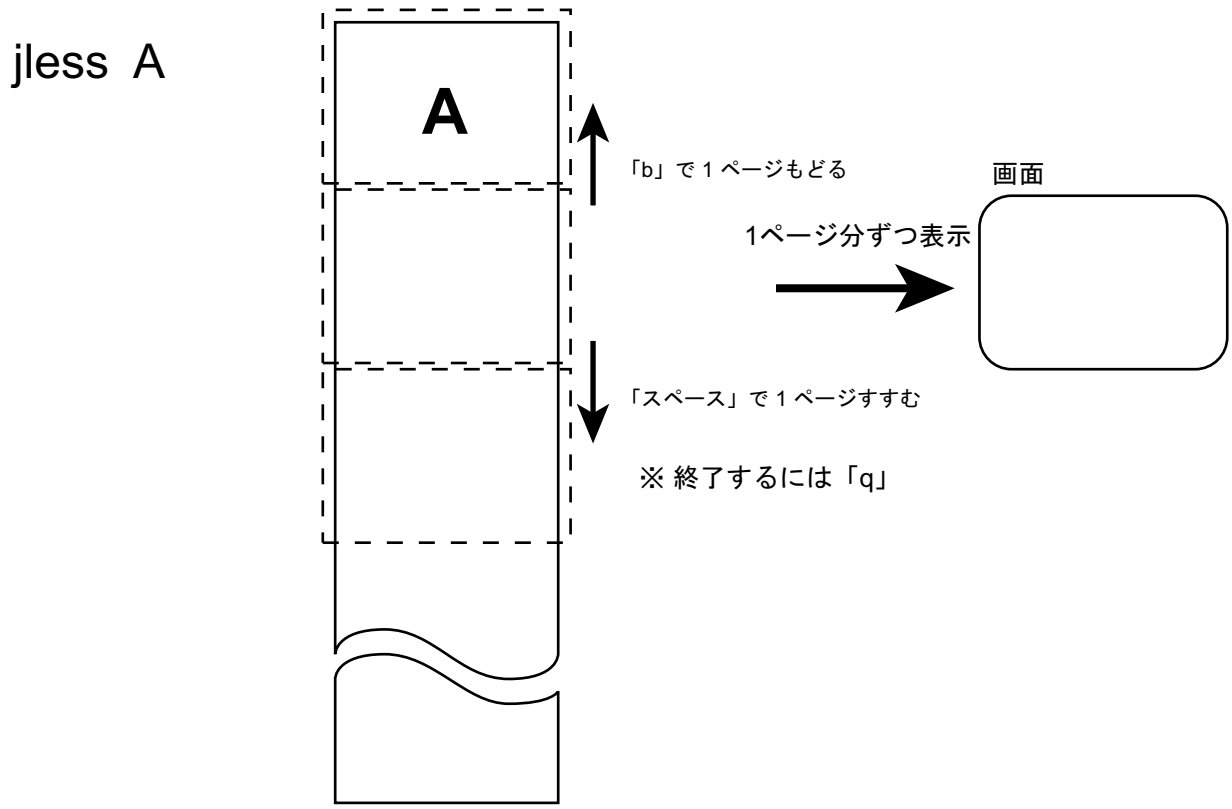
```
                              2000

      1月                  2月                  3月
日 月 火 水 木 金 土  日 月 火 水 木 金 土  日 月 火 水 木 金 土
                1     1  2  3  4  5        1  2  3  4
 2  3  4  5  6  7  8   6  7  8  9 10 11 12   5  6  7  8  9 10 11
 9 10 11 12 13 14 15  13 14 15 16 17 18 19  12 13 14 15 16 17 18
16 17 18 19 20 21 22  20 21 22 23 24 25 26  19 20 21 22 23 24 25
23 24 25 26 27 28 29  27 28 29              26 27 28 29 30 31
30 31
      4月                  5月                  6月
日 月 火 水 木 金 土  日 月 火 水 木 金 土  日 月 火 水 木 金 土
                1     1  2  3  4  5  6           1  2  3
 2  3  4  5  6  7  8   7  8  9 10 11 12 13   4  5  6  7  8  9 10
 9 10 11 12 13 14 15  14 15 16 17 18 19 20  11 12 13 14 15 16 17
16 17 18 19 20 21 22  21 22 23 24 25 26 27  18 19 20 21 22 23 24
23 24 25 26 27 28 29  28 29 30 31           25 26 27 28 29 30
30
      7月                  8月                  9月
日 月 火 水 木 金 土  日 月 火 水 木 金 土  日 月 火 水 木 金 土
                1     1  2  3  4  5              1  2
 2  3  4  5  6  7  8   6  7  8  9 10 11 12   3  4  5  6  7  8  9
 9 10 11 12 13 14 15  13 14 15 16 17 18 19  10 11 12 13 14 15 16
3years
```

jless の操作

jless の基本操作は次のようになっている。

表 5.1: jless の基本操作

動作	コマンド
一ページ進む	スペース または「f」 (フォーワード forward)
一ページ戻る	「b」 (バック backward)
jless を終了する	「q」 (クイット quit)



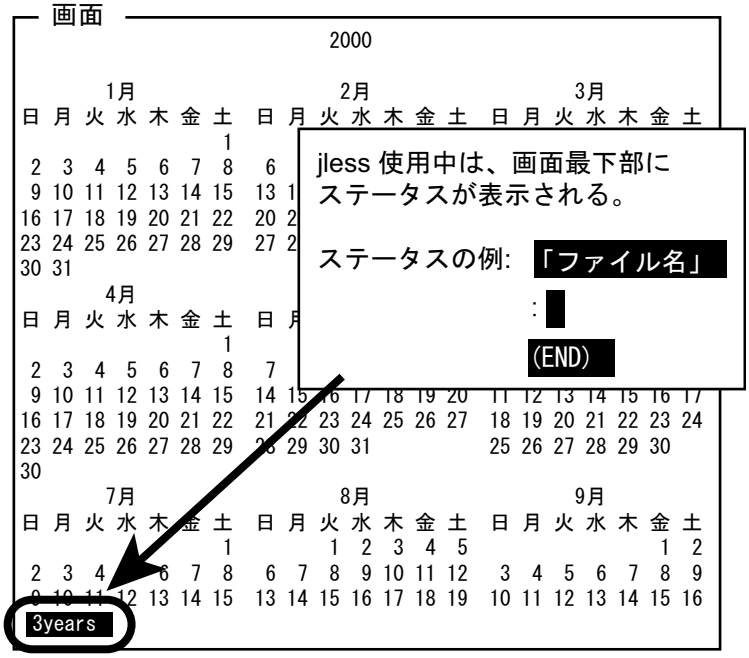
また、次の操作を知っていると便利である。

表 5.2: jless の応用操作

動作	コマンド
ファイルの最初に飛ぶ	Esc を一瞬押して「<」
ファイルの最後に飛ぶ	Esc を一瞬押して「>」
1 行進む	「j」
1 行戻る	「k」
前方の文字をサーチ	「/」に続けてサーチしたい文字列を打って ↵
後方の文字をサーチ	「?」に続けてサーチしたい文字列を打って ↵

jless 使用中のステータス表示

jless を使っているのか、いないのかが、わからなくなることがある。そんなときは、画面の一番下を見よう。jless 使用中には、「3years」のようにファイル名や、「:」や、「(END)」などが画面の一番下に表示されている。このうちの「(END)」は、ファイルの最後にたどり着いたときに表示されるメッセージである。



☆**練習**☆

1. jless の基本操作と応用操作を一通りためしてみよう。

5.1.3 コマンドの実行結果も 1 ページずつ: jless とパイプ

jless は、ファイルの中身だけでなく、次のようにすると、コマンドの実行結果も 1 画面ずつ区切って表示することができる。

「コマンド」 | jless

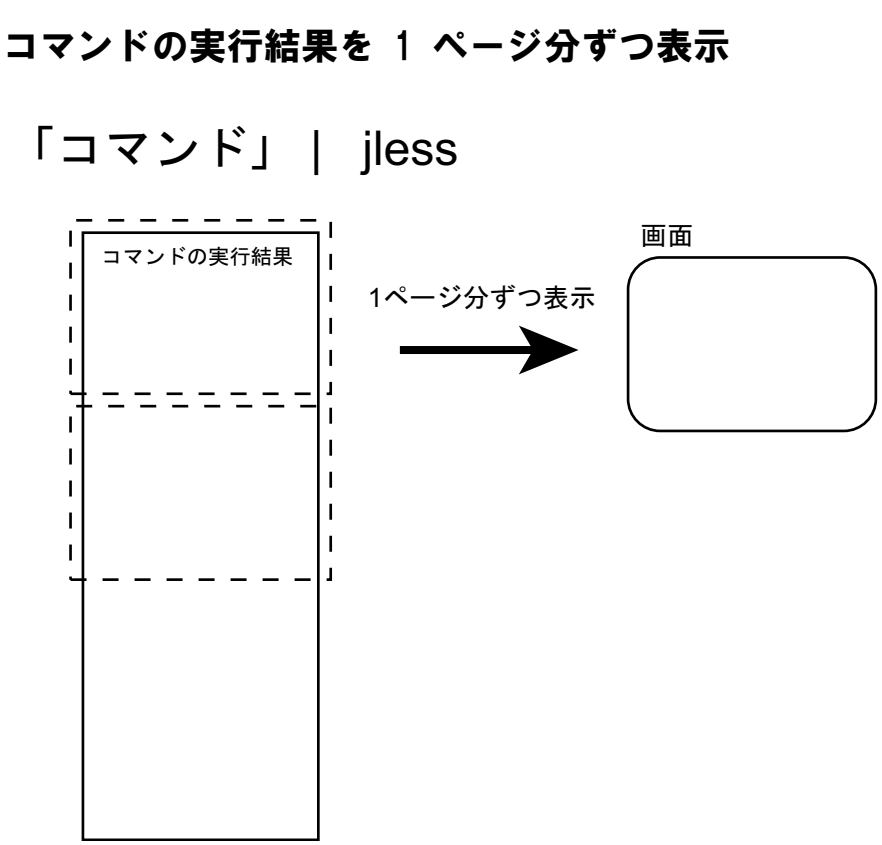
では、 today コマンドの実行結果を、jless で 1 画面ずつ表示してみよう。

```
% today | jless
```

すると、次のようになり、 1 画面ずつ表示できる。jless の使い方は、ファイルの中を見るときと全く同じである。

```
こんにちは。
きょうは、2000 年 10 月 1 日（日曜日）です。
旧暦では、2000 年 9 月 4 日（赤口）[中潮] です。
干支では、庚辰（かのえたつ）の年、壬辰（みずのえたつ）の日です。
九星は、二黒 です。
十二直は、危 です。
二十八宿は、虚 です。

****** きょうは何の日かな? ******
十方暮です。
母倉日です。
大明日です。
凶会日です。
コーヒーの日です。
ネクタイの日です。
衣替えです。
印章の日です。
音楽の日です。
家具の日です。
都民の日 (東京都) です。
日本酒の日です。
法の日です。
川崎市民の日 (神奈川県川崎市) です。
:
```

UNIX システムでは、「|」を使って、コマンドの実行結果を、別のコマンドの入力に渡すことができる。この方法を、**「パイプ – pipe」**と言う。ここでは、 today の実行結果を、パイプを使って、jless コマンドの入力に使ったというわけである。



5.2　Kterm のスクロール

ページャーとは異なるが、Kterm などでもより広い画面を表示したりすることができる。Kterm のメニューなども含めて、簡単に紹介しておこう。

5.2.1　Kterm のメニュー

Kterm のウィンドウ内で、Ctrl を押しながら ボタン 1 ～ ボタン 3 を押すことで、3 種類のメニューを出すことができる。

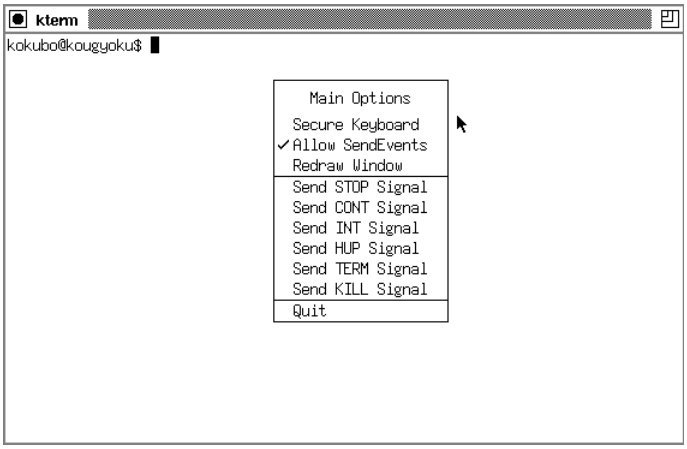
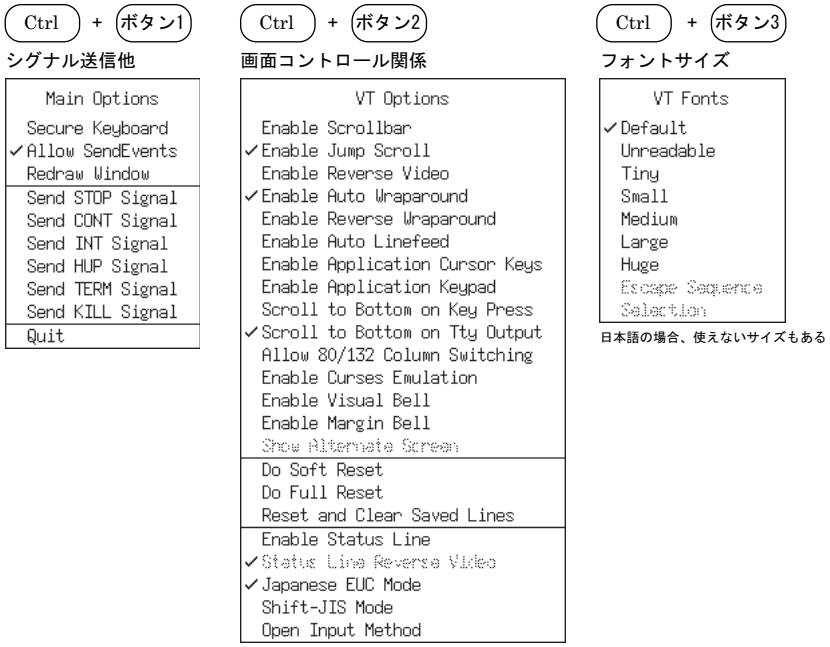


図 5.1: Kterm の画面内でメニューを出したところ

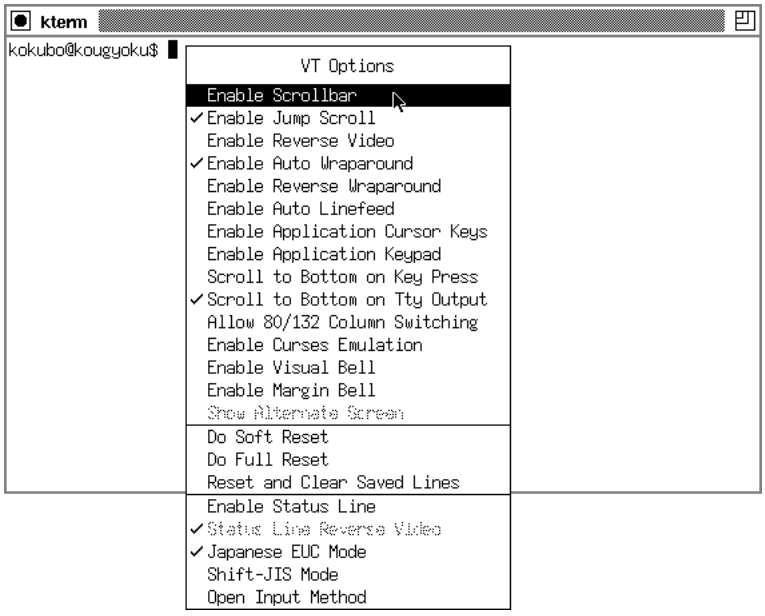


これらの内、 Ctrl+ボタン 3 では、フォントのサイズを変えることができる。もしも、文字が小さく感じられるようであれば、大きめに変更してみるといいだろう。デフォルトは Tiny サイズである。なお一部、日本語には対応していないサイズもある。

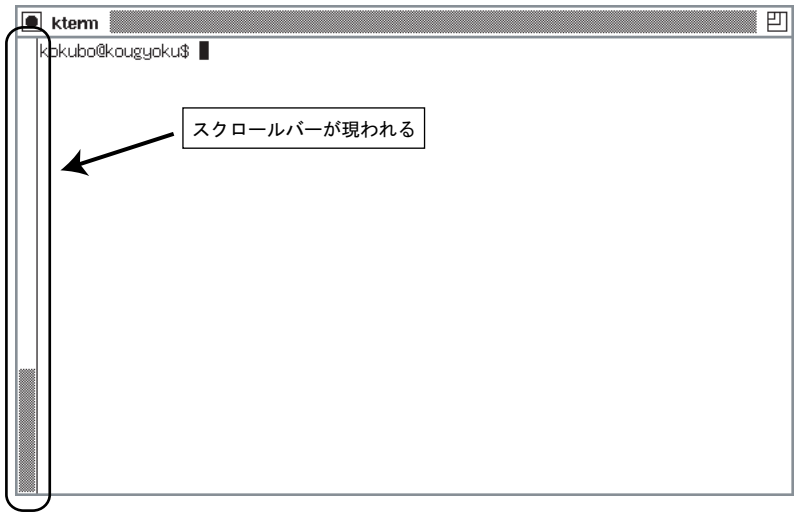
5.2.2 Kterm のスクロールバー

Kterm を使っているときに使用頻度の高いスクロールバーの使い方を紹介しよう。

まず、 Kterm のウィンドウ内で Ctrl+ボタン 2 で、メニューを出し、その一番上の「Enable Scrollbar」を選ぶ。

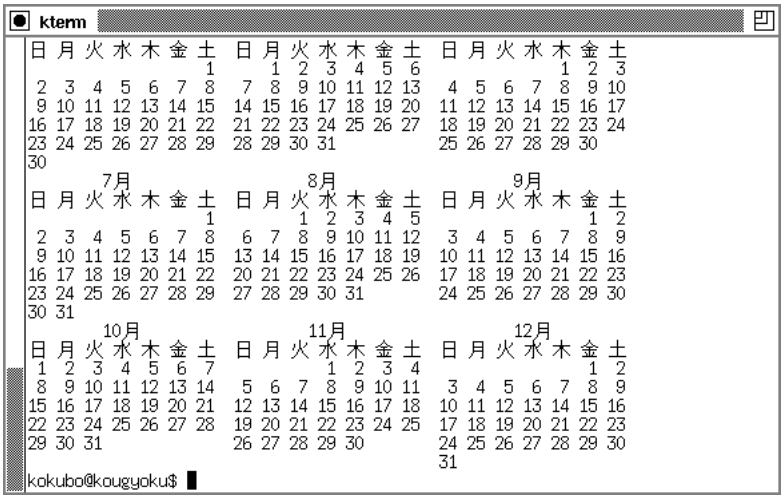


すると、ウィンドウの左側にスクロールバーが現われる。

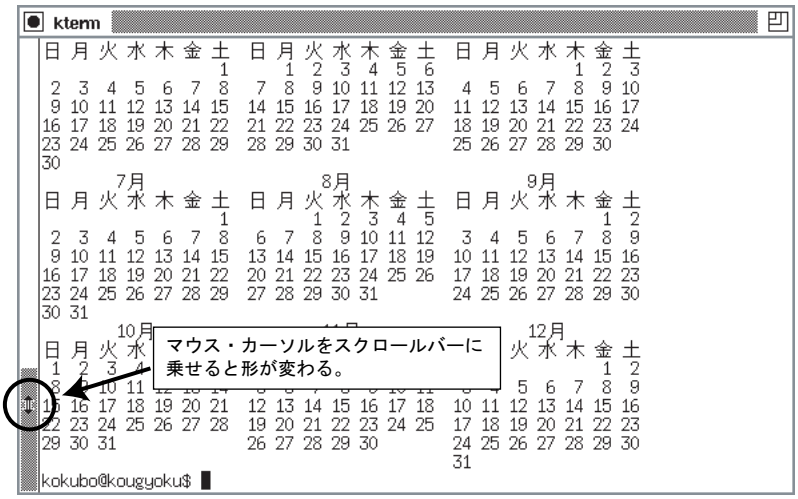


2000 年のカレンダーを画面に表示するコマンド「`cal 2000`」を打ってみる。

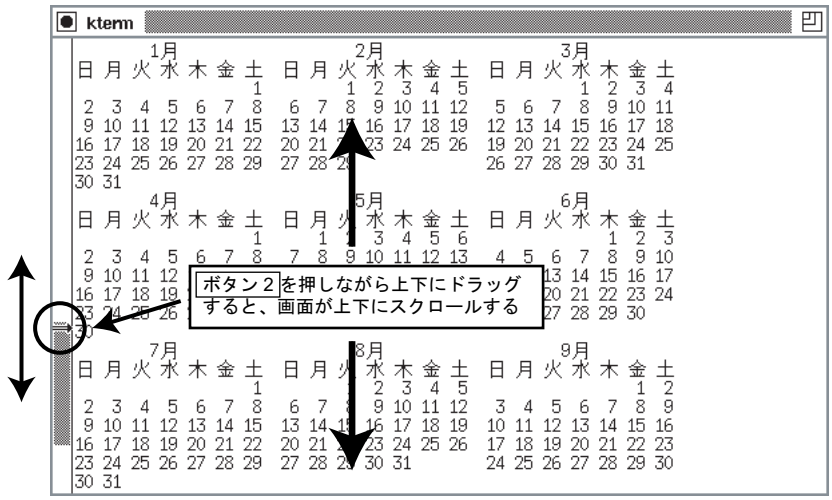
コマンドの出力結果が長すぎて、先頭の方が流れて読めなくなってしまう。このようなときには、紹介したようにページャーを使えばいいのだが、Kterm のスクロールバーを使うことでもある程度対応できる。



マウス・カーソルをスクロールバーにのせると形が変わる。



この状態で ボタン 2 を押したままドラッグすると、画面が上下にスクロールして、前の部分も読むことができる。



5.3 マニュアルを読む: man と jman

UNIX システムには、充実したオンライン・マニュアルが用意されている。マニュアルには、コマンドの使い方、設定ファイルの書き方、C 言語などの関数の仕様を書いたものなどがある。

FreeBSD では、英語のマニュアルを表示する man コマンドと、日本語のマニュアルを表示する jman がある。ただし、すべてのコマンドについて日本語のマニュアルが付いているわけではなく、jman を使っても英語のマニュアルが表示されることもある。

5.3.1 マニュアルを読んでみよう

jman コマンドで、コマンドの使い方を調べるには、次のようにする。

jman 「調べたいコマンド名」

では、次のようにして、 cal コマンドのマニュアルを表示してみよう。

```
% jman cal
```

すると、次のように表示される。このマニュアルの表示には、 jless が使われていて、操作方法はファイルを表示しているときと全く同じである。

```
CAL(1)                  FreeBSD General Commands Manual                  CAL(1)

名称
     cal, ncal - カレンダおよびイースターの日付を表示する

書式
     cal [-jy] [[month] year]
     ncal [-jJpwy] [-s country_code] [[month] year]
     ncal [-Jeo] [year]

解説
     cal は簡単なカレンダを表示します。また ncal は別のフォーマット、追加のオ
     プション、イースターの日付も提供します。新しいフォーマットは込み入ってい
     ますが、 25x80 文字の端末で一年が表示できます。引数が指定されなかった場合
     は今月のものを表示します。

     オプションには以下のものがあります:

     -J      ユリウス暦でカレンダーを表示します。 -e オプションと共に使用する
             と、ユリウス暦でのイースターを表示します。

     -e      イースターの日付を表示します (西方教会)。

:
```

☆**練習**☆

1. jman で、 who のマニュアルを読んでみよう。
2. ls のマニュアルを読んでみよう。

5.3.2 オンライン・マニュアルはどんなときに読むか?

UNIX システムを使い始めたばかりのとき、オンライン・マニュアルを調べても、難しく感じることが多い。たとえば ls のように複雑なこともできる強力なコマンドの場合、非常に長くて初心者には難解なマニュアルが表示されたりして、特にそう感じることだろう。

ここで、オンライン・マニュアルの活用の仕方を紹介しておこう。

オンライン・マニュアルは、 C 言語や UNIX システムの入門書の代わりにするのには、あまり向いていない。コマンドの使い方が、詳しく網羅して書いてあるからだ。

では、どういうときに役に立つのかと言うと、主に次のような場合だ。

1. コマンドや関数のちょっとした使い方を知りたい。

   昔使ったことがあるコマンドの使い方をド忘れしたとき、普段使っているコマンドの特殊な使い方を知りたいときに便利である。また、初めて見るコマンドの使い方をためしに見てみようかというときにもいいだろう。

   また、 C でプログラムを書いていて、関数の使い方を忘れたときにも意外と便利である。

2. キーワード検索したい。

   マニュアルは、画面で読むよりも紙に印刷されているものの方がずっと読みやすい。では、オンライン・マニュアルの利点と言うと、キーワード検索ができることだ。ちょっとしたことを調べたいときに、マニュアルを最初から読むのは大変なので、前の節で紹介した jless の検索機能を使って、キーワードをサーチして読むと便利である。

なお、 man と jman 以外に、info コマンドというのもある。ちなみに、こちらはほとんどのマニュアルが英語で書かれている。

5.3.3 オンライン・マニュアルの構成

オンライン・マニュアルは、いくつかのセクションに別れている。そのセクションの構成を紹介しよう。なお、これは少し難しい話題なので、はじめて UNIX システムを学ぶときには、あまり役に立たないかもしれない。その場合には、読み飛ばしてもかまわない。

表 5.3: オンライン・マニュアルの構成

セクション	内容
1　ユーザ・コマンド	ユーザが通常使うアプリケーションや、ユーティリティなど
2　システム・コール	UNIX システムをコントロールするのに使う関数など
3　ライブラリ	C の関数など
4　デバイス	UNIX システムの周辺機器に関するプログラミング情報など
5　ファイル・フォーマット	ファイルのフォーマットについて
6　ゲーム	ゲームの使い方など
7　その他	その他の情報
8　システム管理	システム管理用のコマンド
9　カーネル	カーネルに関するマニュアル
n　Tcl/Tk	Tcl/Tk の関数など

複数のセクションに同じ名前で存在している項目を調べる場合、マニュアルを表示するのにセクションを指定する必要がある。セクションを指定してマニュアルを表示するには、次のようにする。

jman 「セクション番号」 「調べたい項目」

例えば printf のように、ユーザ・コマンドと C 言語の関数の両方のセクションに同じ名前で存在している項目がある。このとき、ユーザ・コマンドの方の printf のマニュアルを表示したければ、「jman 1 printf」と入力する。また、 C 言語の関数の方の printf のマニュアルを表示するには、「jman 3 printf」と入力する。

この章で紹介したコマンド

ページャーとマニュアル

jless : ファイルの中身やコマンドの出力を 1 画面ずつ表示

使い方: jless 「ファイル名」　　ファイルを表示するとき

「コマンド名」 | jless　コマンドの出力を表示するとき

jman : 日本語マニュアルの表示

使い方: jman 「コマンド名」

jman 「章」 「コマンド名」

第6章　エディタ Mule と Canna による日本語入力

ここまでのところでは、コマンドの実行結果をリダイレクトしてファイルを作っていた。自分でファイルを作ったり、書き換えたりするにはエディタを使う。この章の内容をマスターすれば、自由にプログラムを書いたり、文書を作成したりすることができるようになる。

6.1　エディタ

文章やプログラムなどを書いたり編集したりするのに使うソフトを、エディタという。たとえば、Windows では「メモ帳」、 MacOS X では「TextEdit」などがエディタである。

UNIX システムには、様々なエディタがある。代表的なのは GNU Emacs（グニュー イーマックス）系と vi（ヴィジュアル）系のエディタである。GNU Emacs 系のエディタには、 GNU Emacs、Mule、XEmacs などがある。

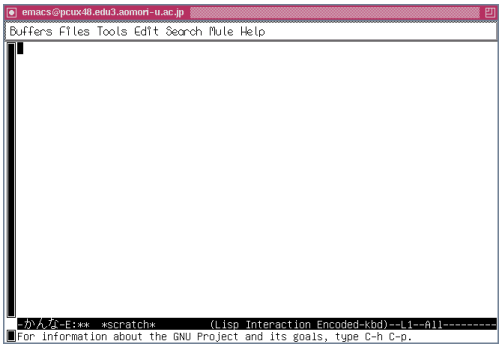


図 6.1: GNU Emacs

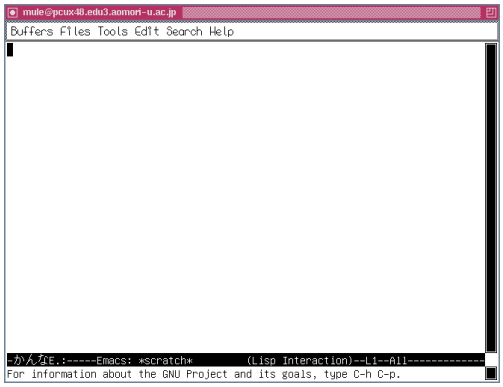


図 6.2: Mule

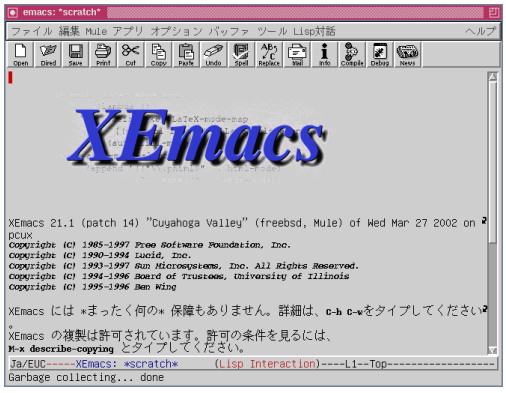


図 6.3: XEmacs

Emacs とは、(Edit MACroS: 編集マクロ) が語源であり、実に様々な機能を持っていて、自分でカスタマイズすることもできる。今回は、その中でも Mule (ミュール) を中心に、基本的なことだけを紹介していこう。

なお、 Mule とは「Multilingual Enhancement to GNU Emacs」の略で、日本の電総研を中心に開発された多言語拡張版 Emacs である。日本語、英語だけでなく、韓国語、中国語、ロシア語、ヘブライ語・・・など、実にいろいろな言語を編集することもできる。もっとも、最近の GNU Emacs や XEmacs では Mule の成果を取り入れて、同様に多言語表示が行えるようになっている。

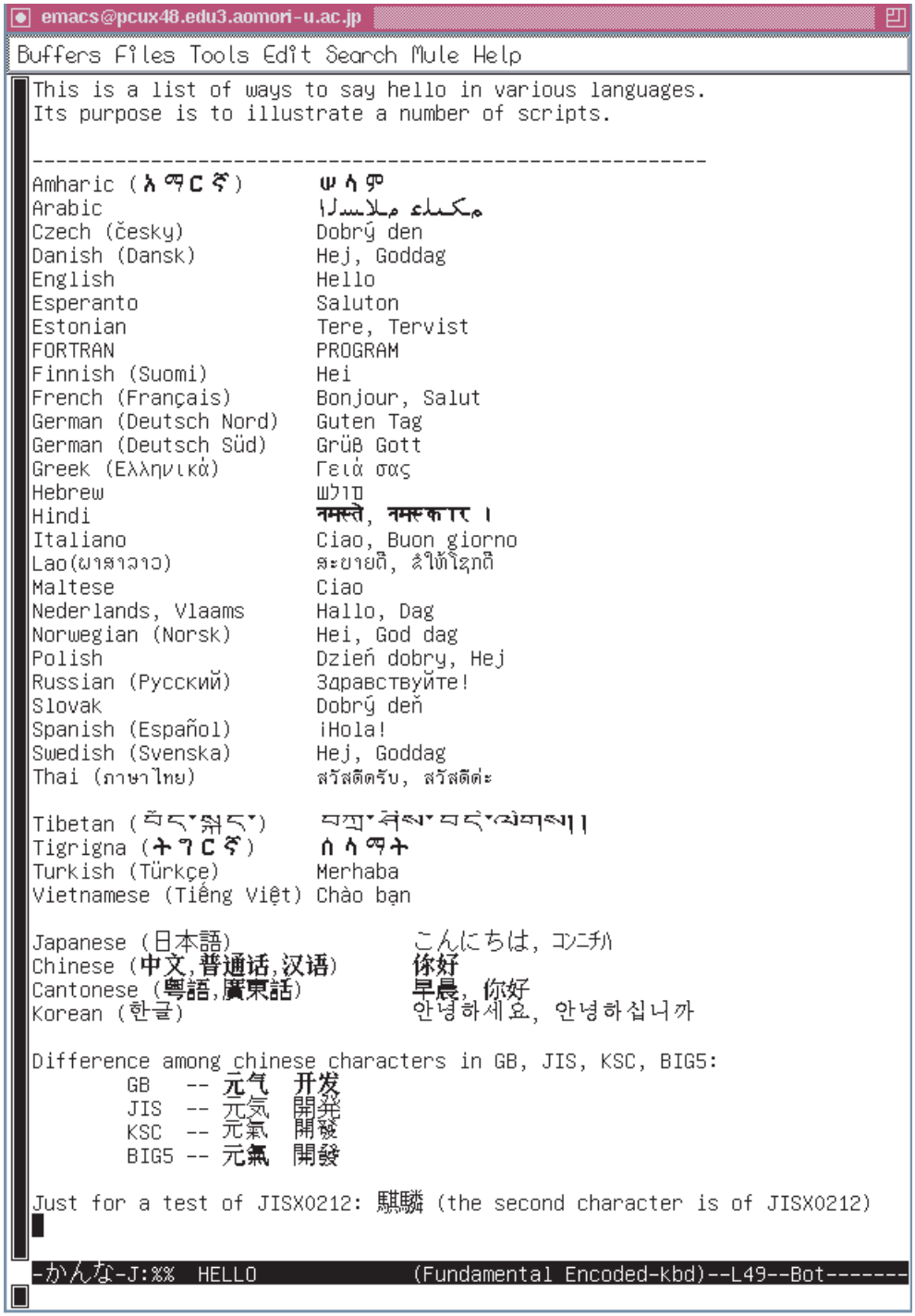


図 6.4: GNU Emacs を使用してさまざまな言語で「こんにちわ」を表示

6.2 Mule の使い方: 基本編その 1 – 起動と終了、文字の入力

6.2.1 Mule の起動

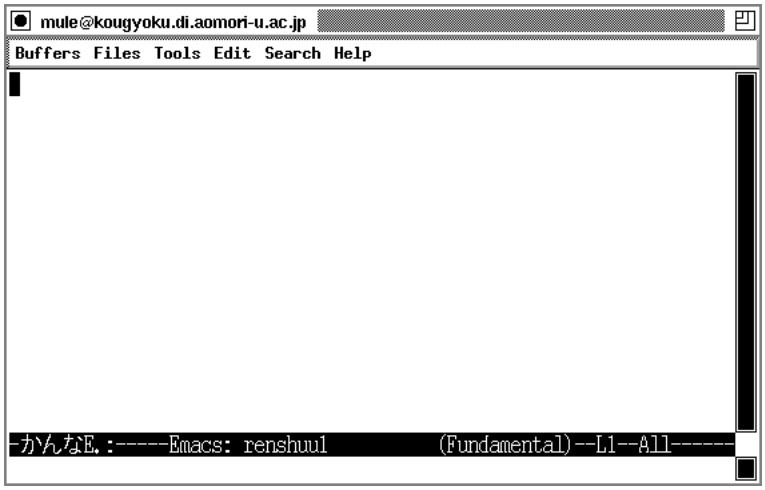
X Window System を使っているとき、 Mule は次のようにして起動する。

```
mule 「ファイル名」 &
```

では「mule renshuu1 &」と入力して、「renshuu1」というファイルを編集してみよう。

```
% mule renshuu1 &
%
```

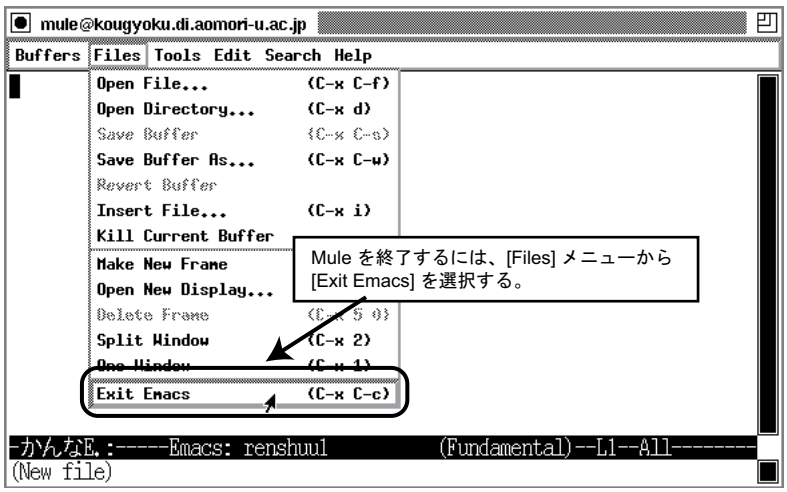
すると、下のようなウィンドウが開く。



6.2.2 Mule の終了

最初に簡単に Mule の終了の仕方を紹介しておこう。実際に使う上では、いくつか注意した方がよい点があるが、それは後の方で紹介することにする。

Mule を終了するには、 [Files] メニューから、[Exit Emacs] を選択する。

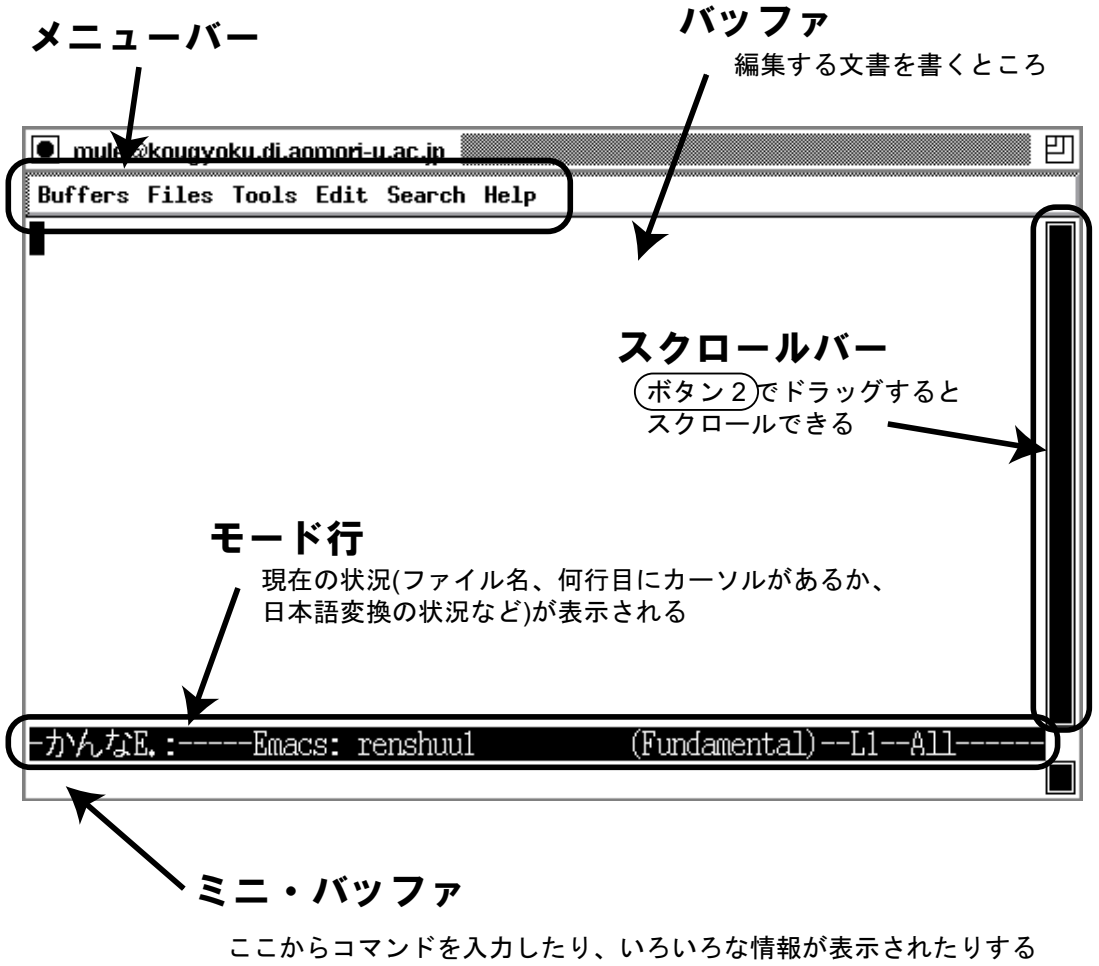


6.2.3 Mule の各部名称

では、再び「mule renshuu1 &」と入力して、Mule を起動しよう。

```
% mule renshuu1 &
%
```

ここで、各部の名前を紹介しておこう。



- **メニューバー**

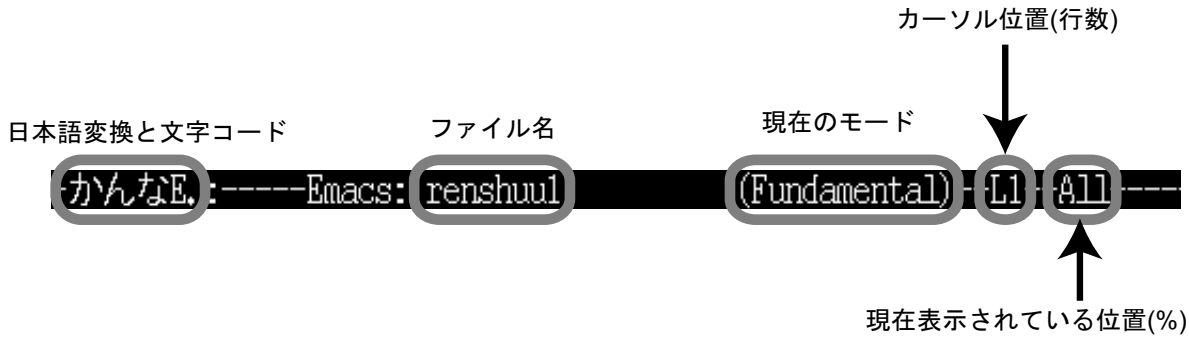
  一番上に表示されるメニュー。これを使うと、簡単に Mule を操作できる。

- **バッファ**

  真ん中の広い部分。ここに文章やプログラムなどを入力する。

- **モード行**

  黒いラインの部分。ファイル名、ファイルの漢字コード、ファイルのどの辺が見えているかなどの情報が表示される。



- **ミニ・バッファ**

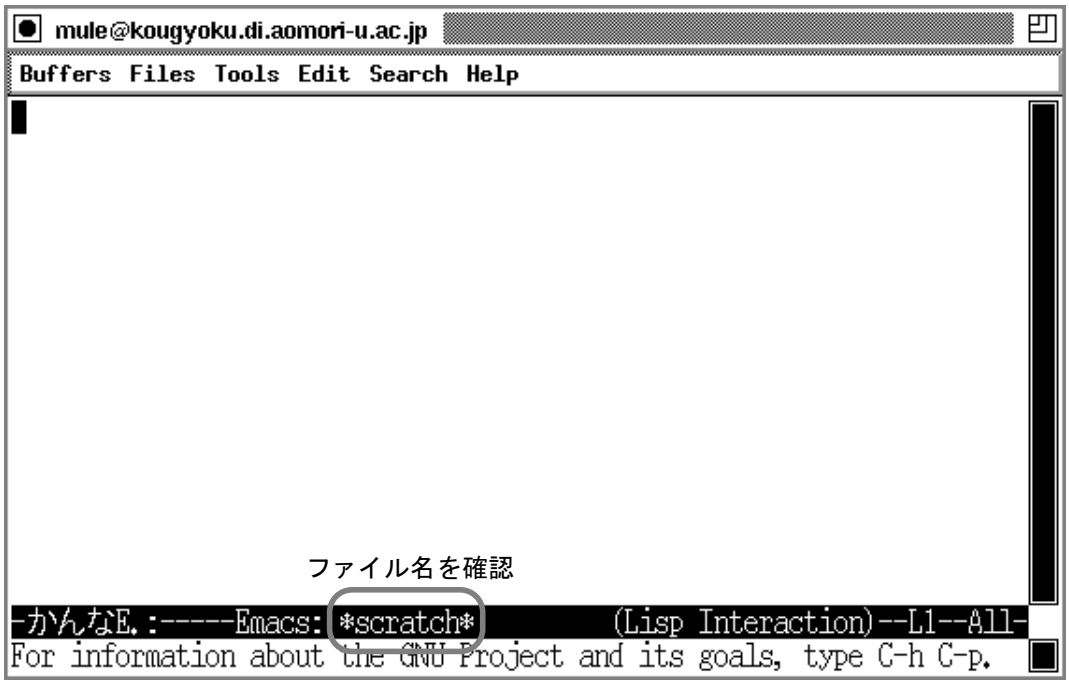
  画面の一番下。コマンドを打ち込むと、ここに表示される。ファイル名を指定したりするときにも使う。

- **スクロールバー**

  画面右側の黒い部分。マウスの ボタン 2 でドラッグすると、スクロールさせることができる。

6.2.4 文字を打つ前に

文字を打つ前に、最初にモード行のところで「ファイル名」を確認しよう。



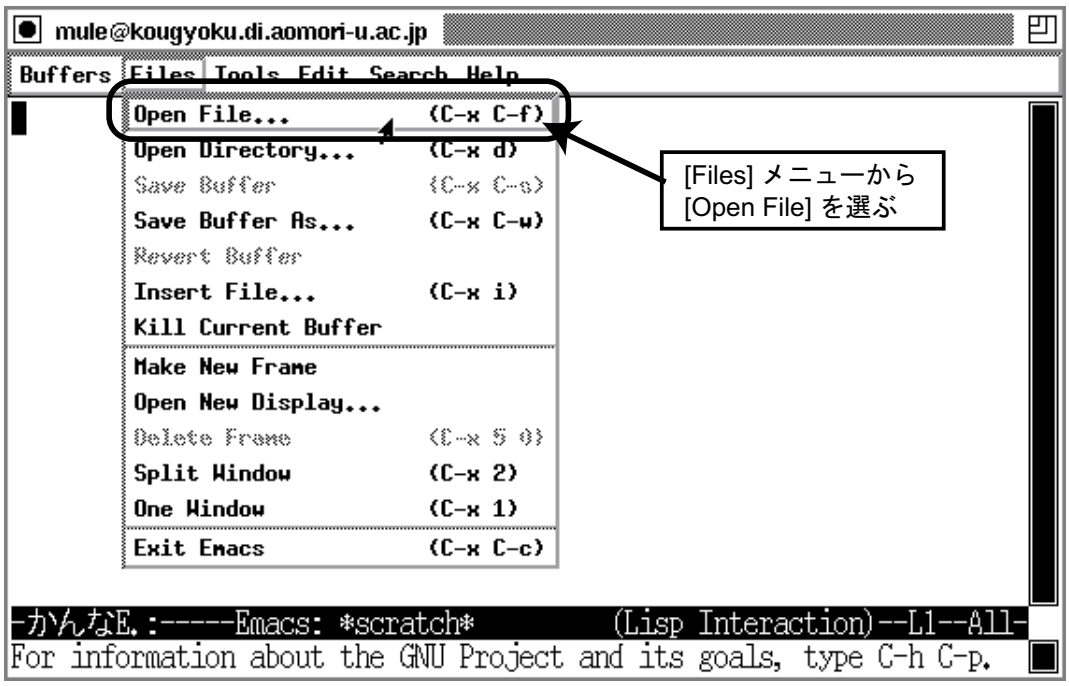
ここに自分で指定したファイル名が入っていれば、文字を入力しても OK だ。

もしも「`*scratch*`」と書かれていたら、それは Mule の起動時にファイル名を指定し忘れた[1]ということだろう。このまま編集を続けると、書いたものを保存し忘れる危険性がある。そこで、今のう

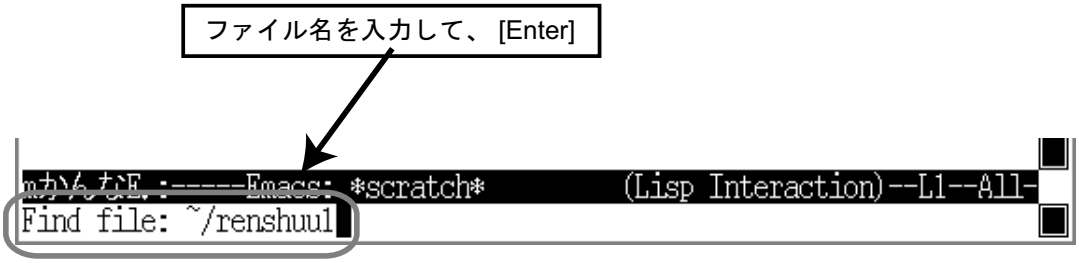
[1]おそらく「`mule` 「ファイル名」 `&`」ではなく、「`mule &`」と起動したのだろう。

ちにファイル名を指定しておこう。

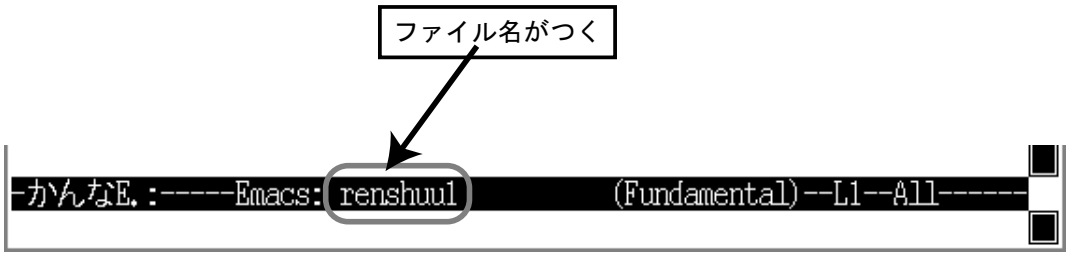
ファイル名を指定するには、[Files] メニューから、[Open File] を選ぶ。



すると、下のミニ・バッファにファイル名を入力して指定できるようになるので、入力して Enter する。

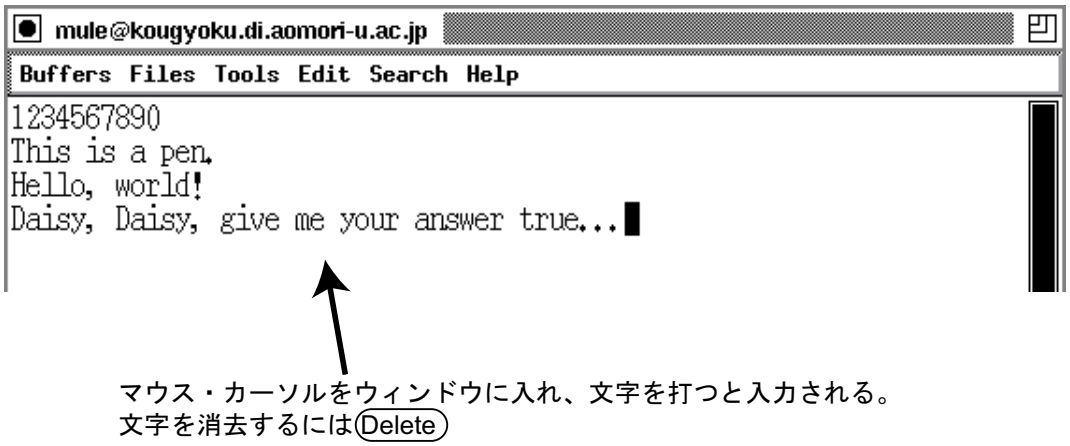


モード行を見ると、ファイル名が正しくついたことがわかる。これで OK である。



6.2.5 文字を打ってみよう

カーソルを Mule のウィンドウに入れて、文字を打って見よう。文字を消すには Delete を使う。



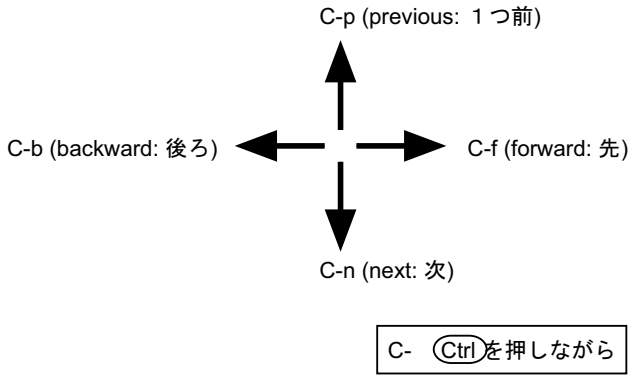
6.2.6 カーソルの移動

Mule でカーソルを移動する方法は幾つかある。X Window System を使っている場合、一番簡単なのは、マウスと矢印キーを使うことだ。

慣れてくると、いちいちマウスを使ったり、矢印キーを押したりすると、すばやく打てなくて面倒になってくる。その日の来ることを信じて、次の方法も憶えておこう

表 6.1: カーソルの移動

移動方向	コマンド	意味	
↑	C-p	ひとつ前の行へ	previous
↓	C-n	ひとつ次の行へ	next
→	C-f	一文字前に	forward
←	C-b	一文字後ろに	backward



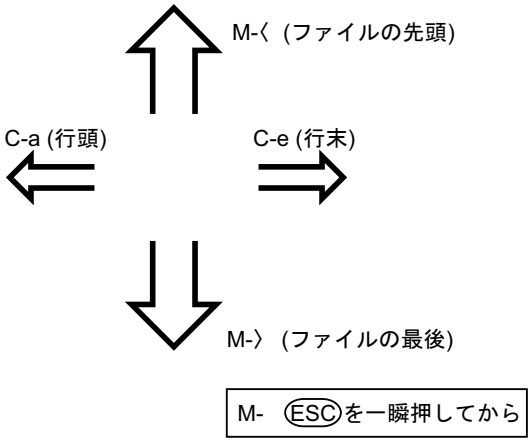
なお Mule では、Ctrl+p のことを「C-p」と書く。

また、これ以外にも「M-x」というのもある。「M」は「Esc を一瞬押すこと」を意味している。つまり「M-x」と書いたら、「一瞬 Esc を押して、x を押す」という意味になる。このとき Esc は、 Ctrl のように押しっぱなしにしてはいけない。

移動に関しては、他にも知っていると便利なコマンドが幾つかある。

表 6.2: カーソルの特殊移動

動作	コマンド
ファイルの先頭	M-<
ファイルの最後	M->
行の先頭	C-a
行の後ろ	C-e
N 行目に行く	M-x goto-line↵行番号↵



6.2.7　キャンセルとアンドゥ

Mule の中では、コマンドのキャンセルは「C-g」である。変になったり、わけがわからなくなったら、とりあえず「C-g」と入力してみよう。

また、アンドゥ(実行したコマンドを元に戻す) は「C-x u」である。

表 6.3: キャンセルとアンドゥ

動作	コマンド
キャンセル	C-g
アンドゥ	C-x u

6.3 Canna による日本語入力

Mule で日本語入力をするソフトは Wnn、 Canna、 Sj3、 SKK など幾つかあるが、演習室の Mule では「Canna(かんな)」を使う。では、実際に使ってみよう。

日本語モードと英語モードの切り替え

Canna で、日本語モードと英語モードの切り替えは「C-o」である。「C-o」と打つと、モード行に [あ] と表示されて、日本語モードになったことがわかる。

また、ここでもう一度「C-o」すると、 [あ] が消えて、英語モードにもどる。

C-o で 「日本語入力モード」←→「英語モード」が切り替わる

日本語入力モードでは、[あ] と表示される。

日本語の入力

ローマ字入力

日本語はローマ字入力になっていて、 Windows などとおおよそ同じである。小さい「ぁ」などの入力は「xi」などになっている。また、「ん」の入力は、「nn」、「n'」、「mn」などになる。

表 6.4: ローマ字表: 直音

ア行	あ/a	い/i	う/u	え/e	お/o
カ行	か/ka	き/ki	く/ku	け/ke	こ/ko
サ行	さ/sa	し/si(shi)	す/su	せ/se	そ/so
タ行	た/ta	ち/ti(chi)	つ/tu(tsu)	て/te	と/to
ナ行	な/na	に/ni	ぬ/nu	ね/ne	の/no
ハ行	は/ha	ひ/hi	ふ/hu(fu)	へ/he	ほ/ho
マ行	ま/ma	み/mi	む/mu	め/me	も/mo
ヤ行	や/ya		ゆ/yu		よ/yo
ラ行	ら/ra	り/ri	る/ru	れ/re	ろ/ro
ワ行	わ/wa	ゐ/wi		ゑ/we	を/wo
ガ行	が/ga	ぎ/gi	ぐ/gu	げ/ge	ご/go
ザ行	ざ/za	じ/zi(ji)	ず/zu	ぜ/ze	ぞ/zo
ダ行	だ/da	ぢ/di	づ/du	で/de	ど/do
バ行	ば/ba	び/bi	ぶ/bu	べ/be	ぼ/bo
パ行	ぱ/pa	ぴ/pi	ぷ/pu	ぺ/pe	ぽ/po

表 6.5: ローマ字表: 拗音

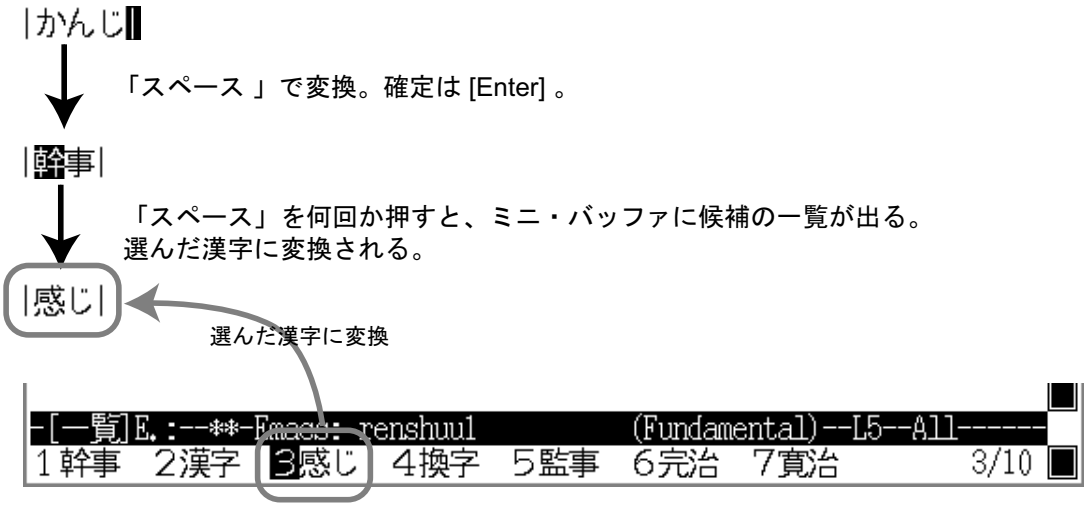
カ行	きゃ/kya	きゅ/kyu	きょ/kyo
サ行	しゃ/sya(sha)	しゅ/syu(shu)	しょ/syo(sho)
タ行	ちゃ/tya(cha)	ちゅ/tyu(chu)	ちょ/tyo(cho)
ナ行	にゃ/nya	にゅ/nyu	にょ/nyo
ハ行	ひゃ/hya	ひゅ/hyu	ひょ/hyo
マ行	みゃ/mya	みゅ/my	みょ/myo
ラ行	りゃ/rya	りゅ/ryu	りょ/ryo
ガ行	ぎゃ/gya	ぎゅ/gyu	ぎょ/gyo
ザ行	じゃ/zya(ja)	じゅ/zyu(ju)	じょ/zyo(jo)
ダ行	ぢゃ/dya	ぢゅ/dyu	ぢょ/dyo
バ行	びゃ/bya	びゅ/byu	びょ/byo
パ行	ぴゃ/pya	ぴゅ/pyu	ぴょ/pyo

単語一個を変換

単語を変換してみよう。まず最初に日本語モードする。そして、ためしに「かんじ (実際の入力は kanji)」と打ってみよう。画面には「|かんじ|」と表示される。

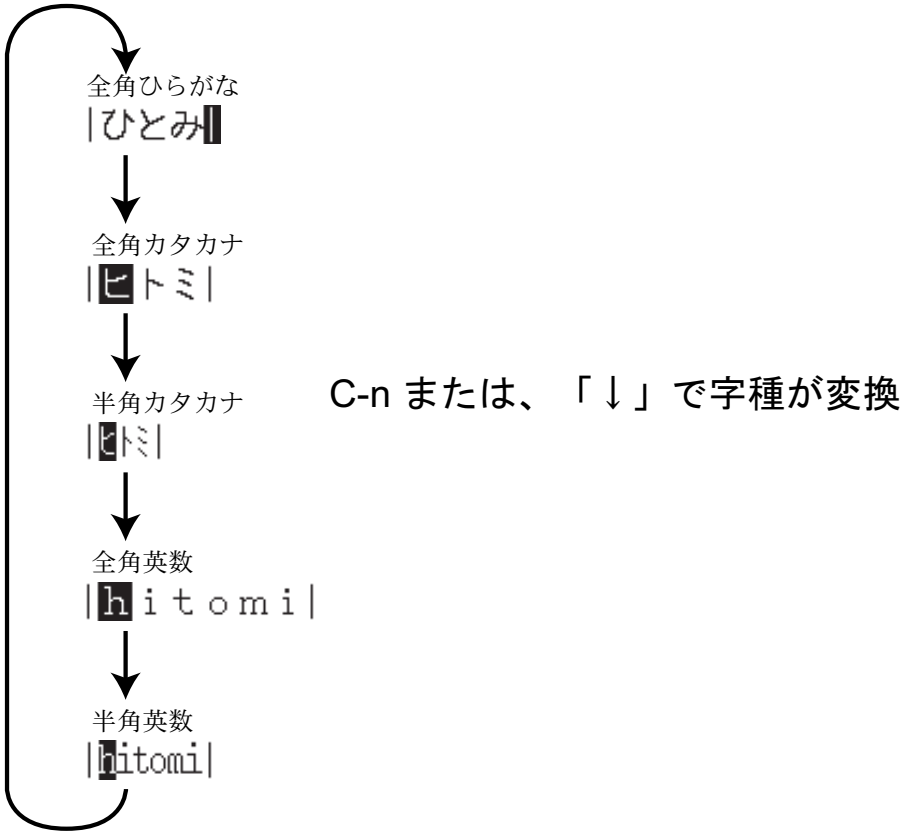
ここでこのまま [↵] すると、ひらがなが入力される。

漢字に変換するには [スペース] を押す。[スペース] を押していくと、今度は候補が下のミニ・バッファに表示される。気に入った候補が選ばれたら、[↵] すると確定する。また、変換を途中でやめる場合には、「C-g」である。



文字種の変換

カタカナや (全角の) 英数など、文字種を変えるには、 ↑ や ↓、または「C-n」を何回か入力する。なお半角のカタカナは UNIX では使用しない方が無難である。



まとめると次のようになる。

表 6.6: 変換のコマンド

動作	コマンド
日本語 / 英語入力モードの切り替え	C-o
漢字へ変換	スペース
カタカナや英数に変換	↓ や ↑、または C-p
変換のキャンセル	C-g
変換の確定	↵

文節を変換

では、文節を日本語変換してみよう。

ためしに「あくまのにんぎょう」と打って変換してみよう。

すると、「|悪魔の 人形|」と表示されて、「あくまの」と「にんぎょう」のところで文が区切られて変換された。

ところが、実は この人は「あ」「熊の」「人形」と区切って変換して欲しかった としよう。

ここで文節の切り替えが必要になってくる。先に文節関係のコマンドを全部書いておくと、次のようになる。

表 6.7: 文節操作

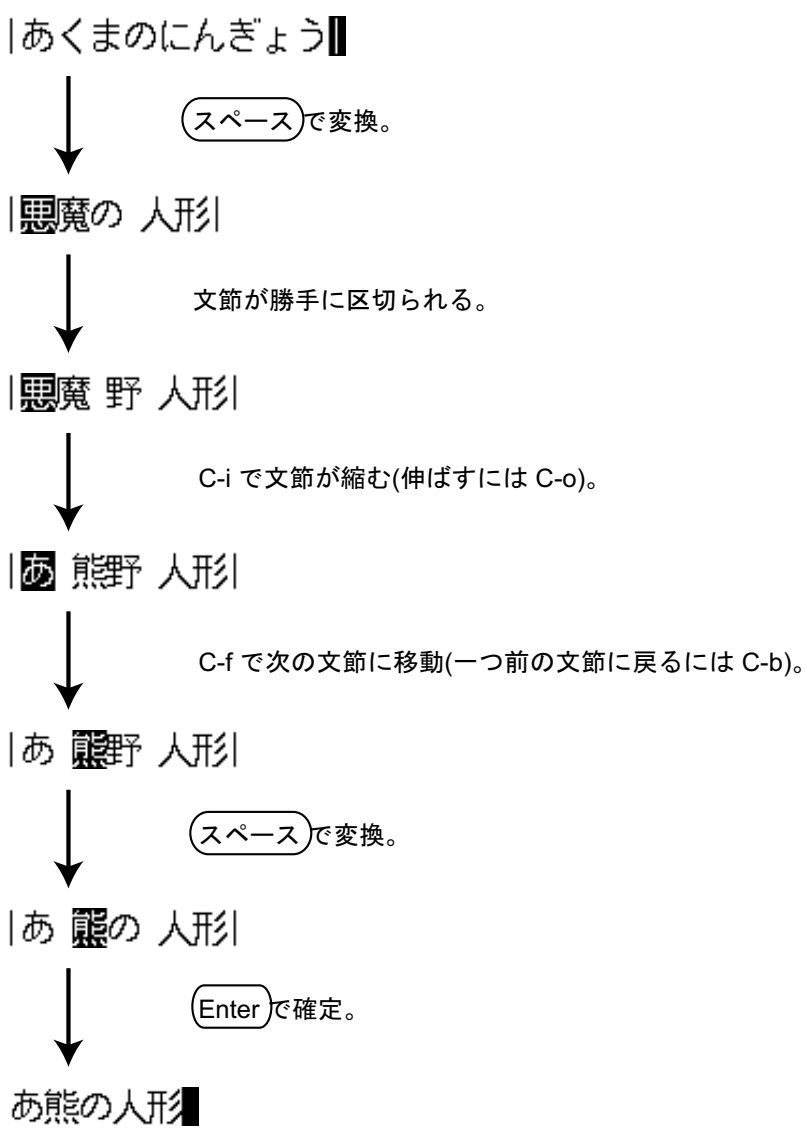
動作	コマンド
文節を縮める	C-i
文節を伸ばす	C-o
一つ前の文節にカーソルを移動する	C-b
次の文節にカーソルを移動する	C-f

つまり、まず「あくまの」という文節を縮めて、「あ」と「くまの」に分ける必要があるので、「C-i」を何回か打つ。

すると、「|あ 熊野 人形|」と、文節はうまくわかれた。

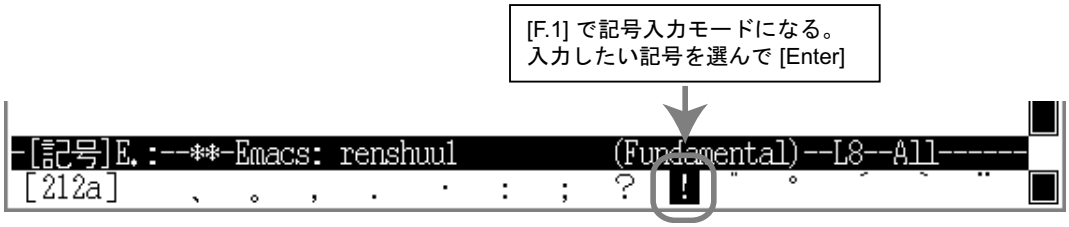
次に、 2 番目の文節を正しく変換しなおす必要があるので、「C-f」と打ってカーソルを 2 番目の文節に移す。

そして スペース で変換して、「熊の」になったところで、軽く ↵ して、変換を確定する。これで「|あ 熊の 人形|」と、希望する変換になったので、再び ↵ して全体を確定する。

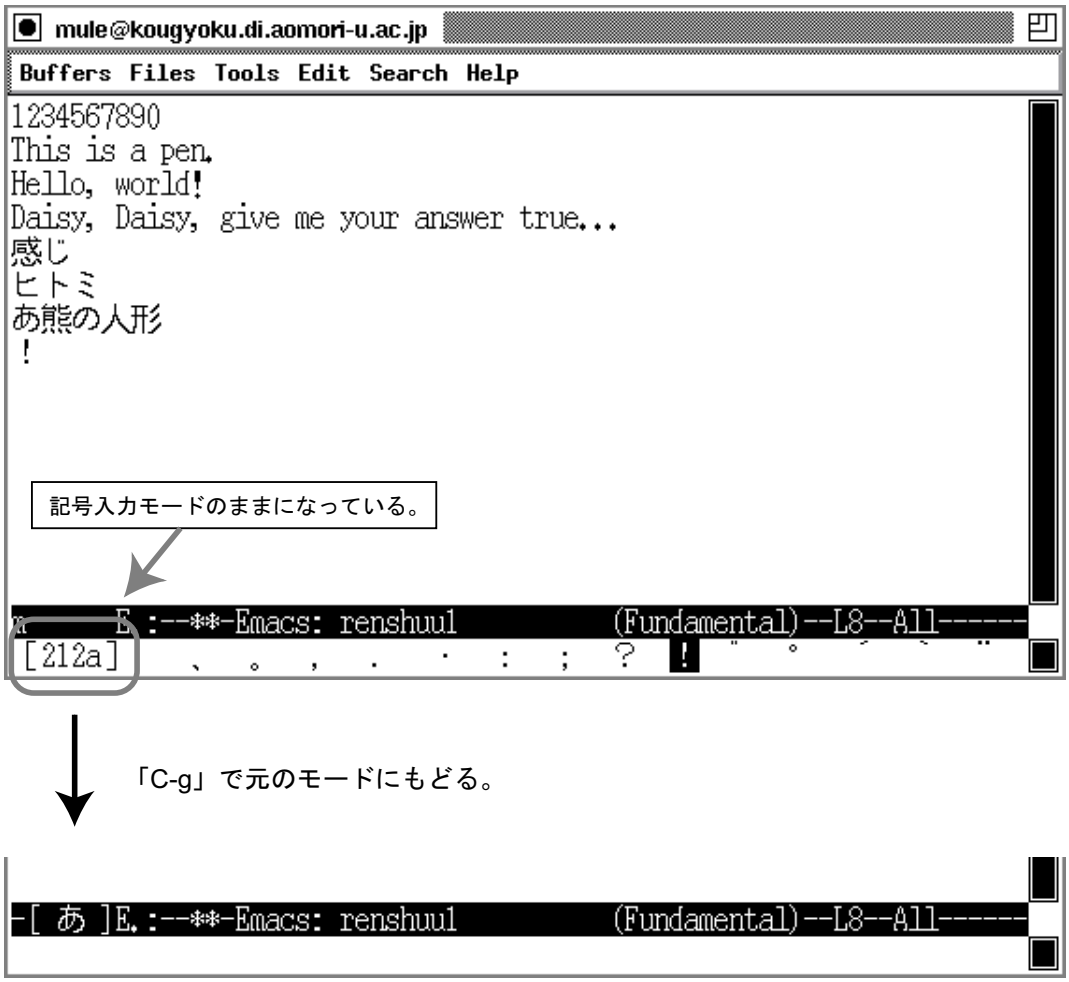

|あくまのにんぎょう
スペースで変換。
|悪魔の 人形|
文節が勝手に区切られる。
|悪魔 野 人形|
C-i で文節が縮む(伸ばすには C-o)。
|あ 熊野 人形|
C-f で次の文節に移動(一つ前の文節に戻るには C-b)。
|あ 熊野 人形|
スペースで変換。
|あ 熊の 人形|
Enter で確定。
あ熊の人形

記号入力

記号を入力するには Insert か F.1 を押して、ミニ・バッファにリストを表示させて選ぶ。



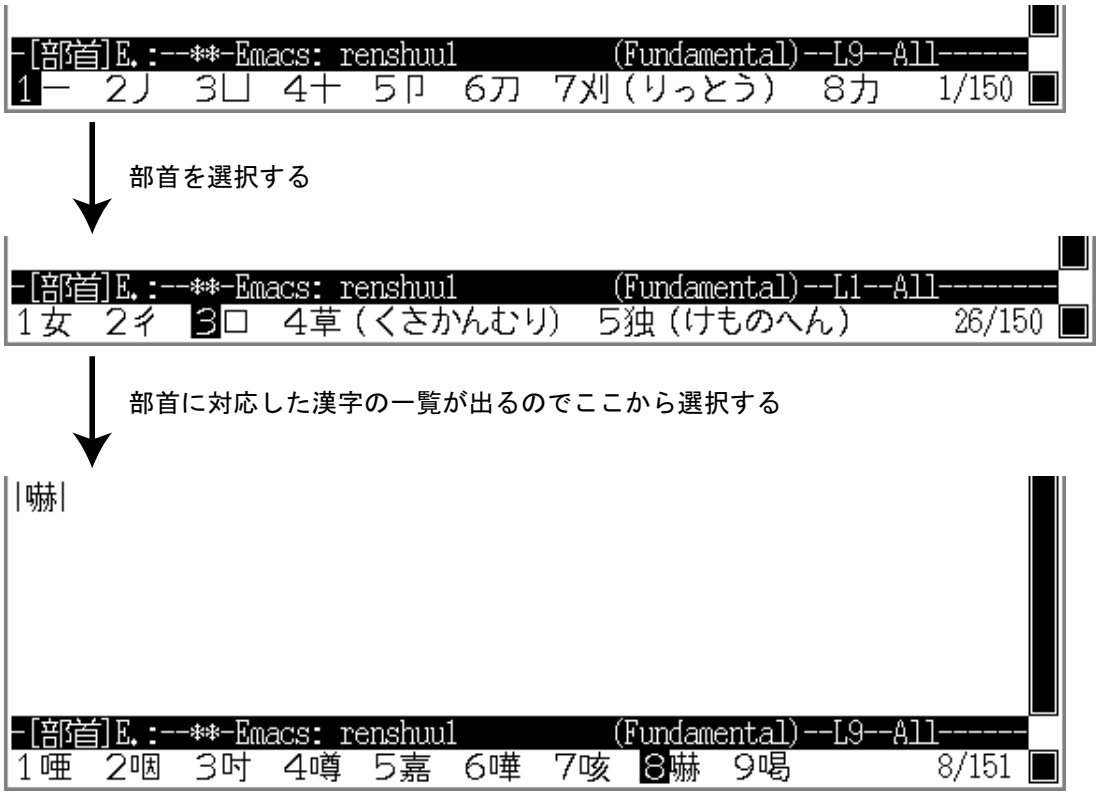
記号入力を終えても、そのままでは記号入力モードのままなので、「C-g」を入力して元のモードに戻る。



部首入力

部首入力は F.3。部首を選ぶと漢字のリストが出るので、ここから選択する。

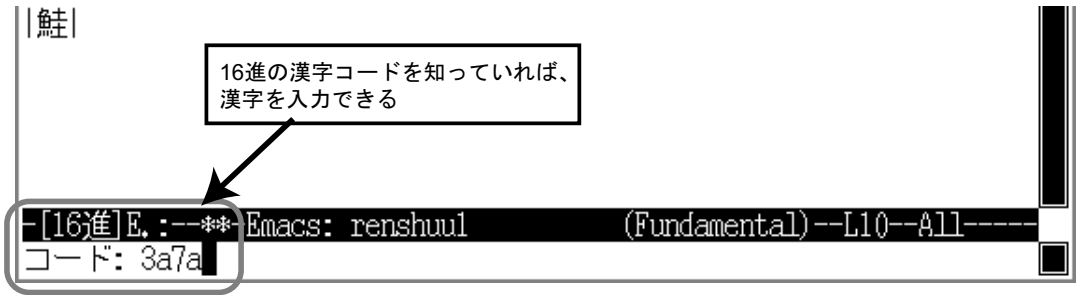
[F.3] で部首入力モードになる



コード入力

コード入力は F.2。16 進の漢字コードを知っていれば自由に入力することができる。

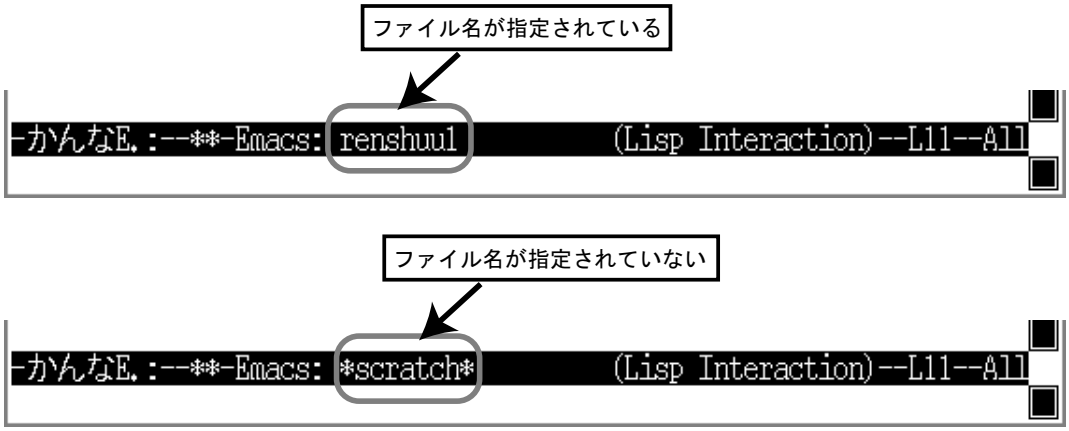
[F.2] で部首入力モードになる



6.4 Mule の使い方: 基本編その 2 – ファイルの保存

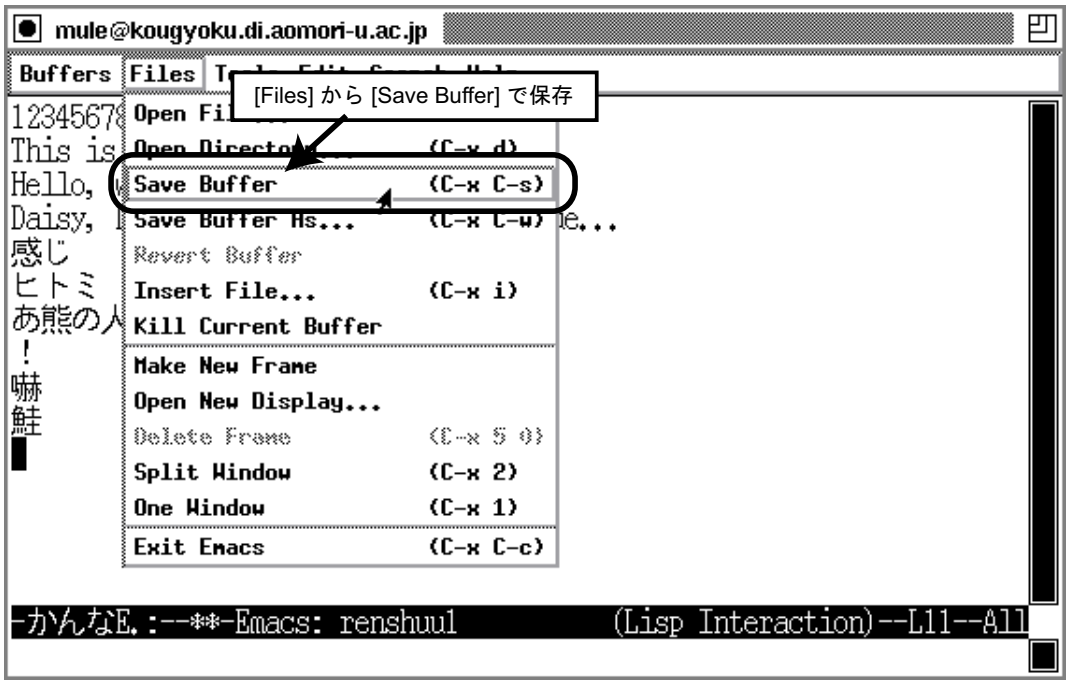
6.4.1 ファイルの保存

ファイルを保存する前に、まずミニ・バッファのファイル名を確認してみよう。

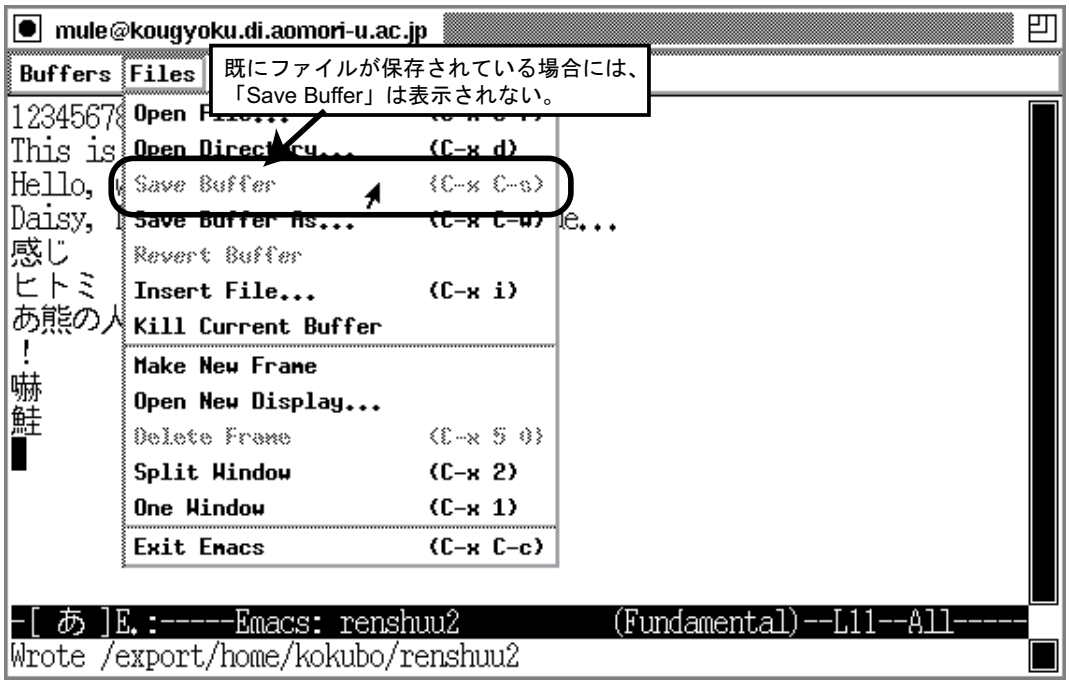


ファイル名が指定されている場合

ファイル名が指定されている場合には、メニュー・バーの [Files] から [Save Buffer] を選択する。または、「C-x C-s」と入力してもよい。

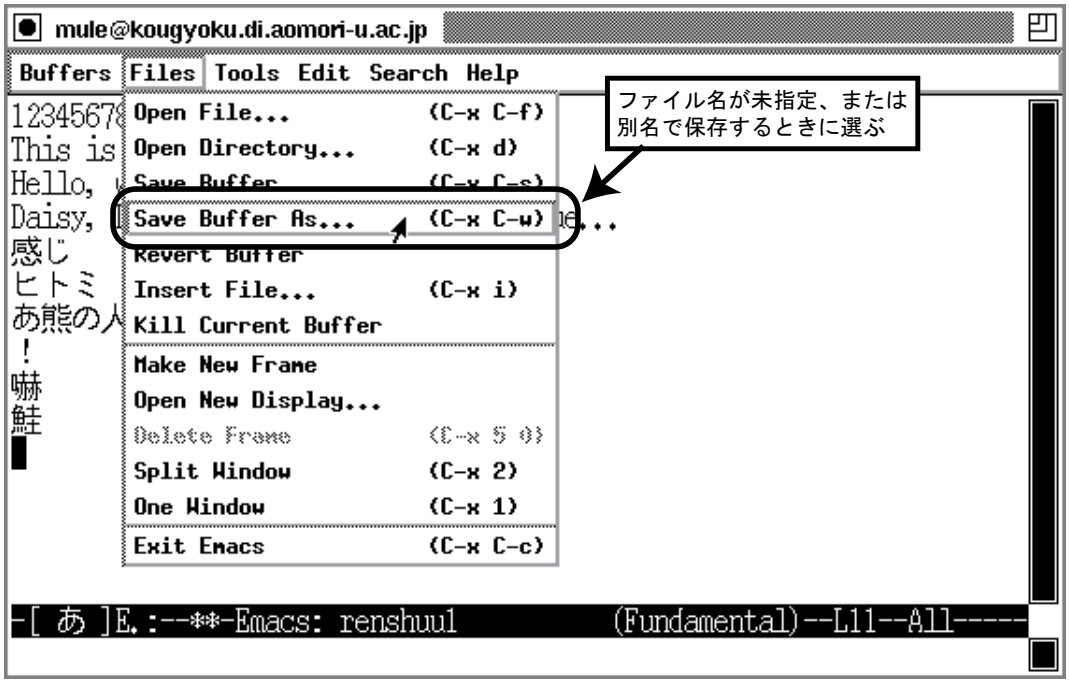


なお、既にファイルが保存されている場合には、[Save Buffer] が表示されない。

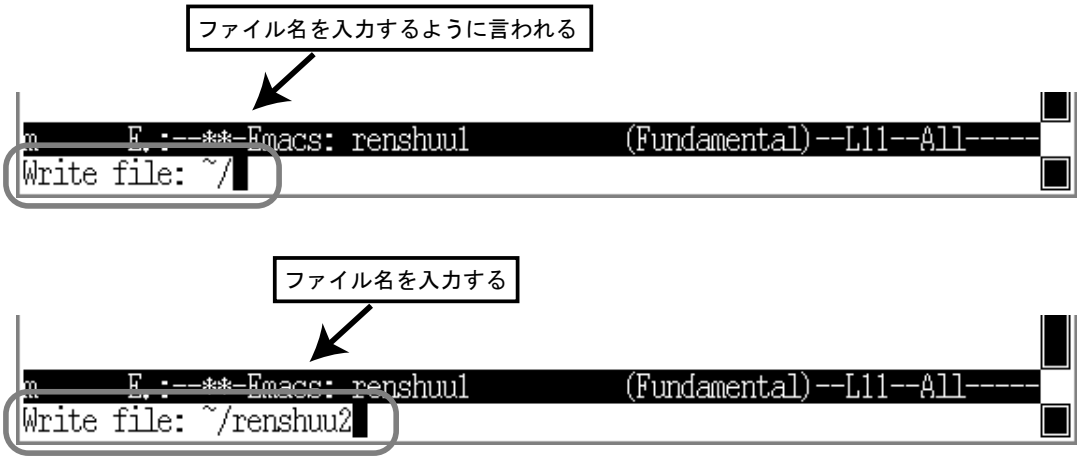


ファイル名が指定されていない、または別なファイル名で保存したい場合

ファイル名を指定し忘れて「*scratch*」になっている場合、または別なファイル名で保存したい場合には、[Files] から [Save Buffer As] を選択する。「C-x C-w」と入力してもよい。

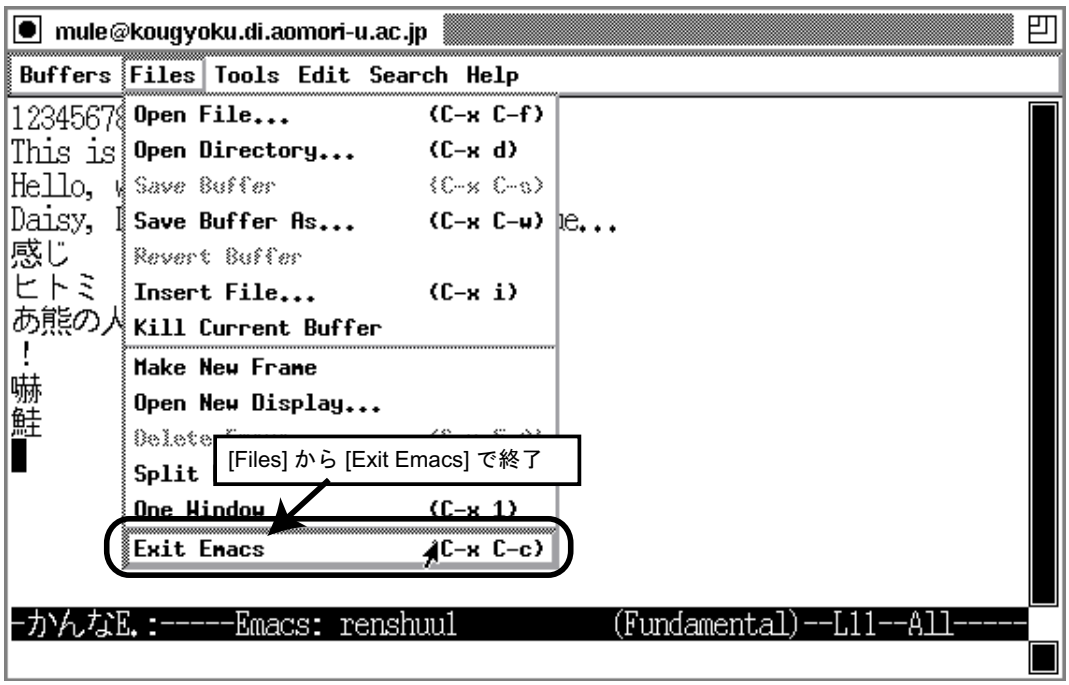


すると、ミニ・バッファにファイル名を入力するように言われるので、ファイル名を入力して、↵する。



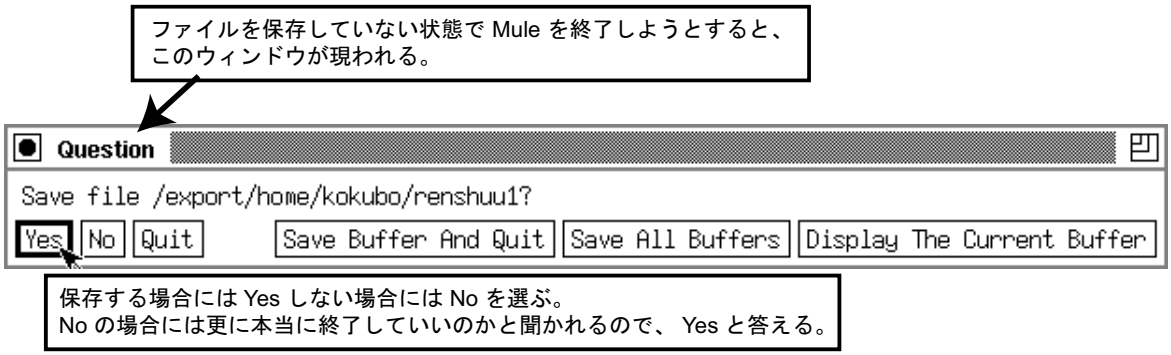
6.4.2　Mule の終了

一部、繰り返しになるが、 Mule の終了の仕方を紹介しよう。`[Files]` から `[Exit Emacs]` を選ぶと終了できる。または、「`C-x C-c`」と入力してもよい。



もしも何かファイルの内容を変更して、保存しないで終了しようとすると、ウィンドウが開いて、保存するかどうか聞かれる。

セーブしたければ、 Yes を、セーブしたくないときは No をクリックする。ここで No を選ぶと、セーブしていないけれど本当に終わっていいかと聞かれ、ここで更に Yes を選ぶと終了する。



なお、「C-x C-c」と入力した場合には、ウィンドウが現れずに、ミニ・バッファに同様の質問がでるので、「y」か「n」などで答える。

6.4.3 [Files] メニューの項目

ここで、 [Files] メニューの主な項目を紹介しておこう。

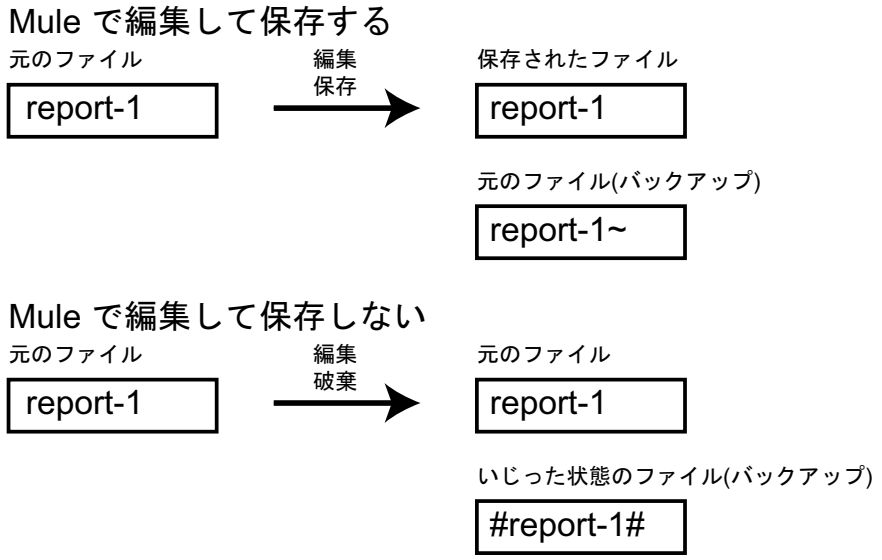
表 6.8: [Files] メニューの主な項目

英語	意味
Open File...	ファイルを開いて編集する
Save Buffer	保存する
Save Buffer As...	名前を付けて保存する
Insert File...	カーソルの位置に別のファイルを読み込む
Kill Current Buffer	現在、編集しているものを廃棄する
Make New Frame	新しいウィンドウを開く
Delete Frame	このウィンドウを閉じる
Split Window	画面を上下に分割する
One Window	分割された画面のうちカーソルがあるもの以外を閉じる
Exit Emacs	Mule を終了する

6.4.4 Mule のつくり出すファイル

Mule では、ファイルを編集して保存すると、バックアップとして元のファイル名の後ろに「~」の付いたファイルを作る。このため、編集した 1 回前の状態のファイルだけは自動的に保存されている。もしも、編集しそこねた場合には、このファイルを元のファイルに上書きコピーすればよい[2]。

また、ファイルの編集中に、元のファイル名の最初と最後に「#」を付けたファイルを作る。これは、ファイルをセーブして終了すると自動的に消えるが、セーブしない場合にはそのまま残る。



[2] もしもファイル名が「A」なら、「cp A~ A」とか。

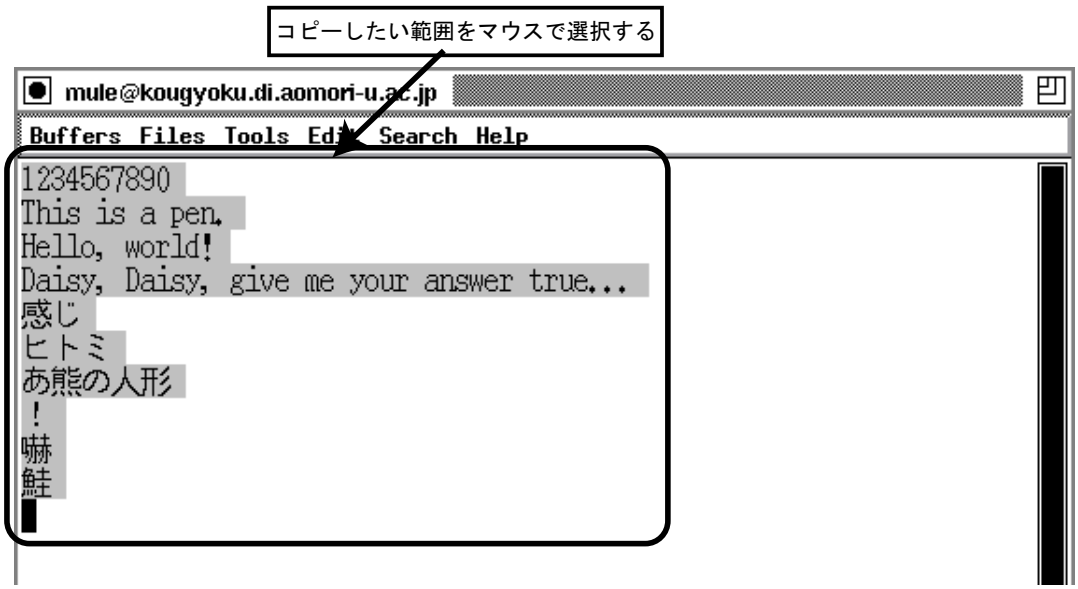
6.5 Mule の使い方: 応用編 – カット & ペースト

ここでは、知っていると便利な機能を紹介しよう。

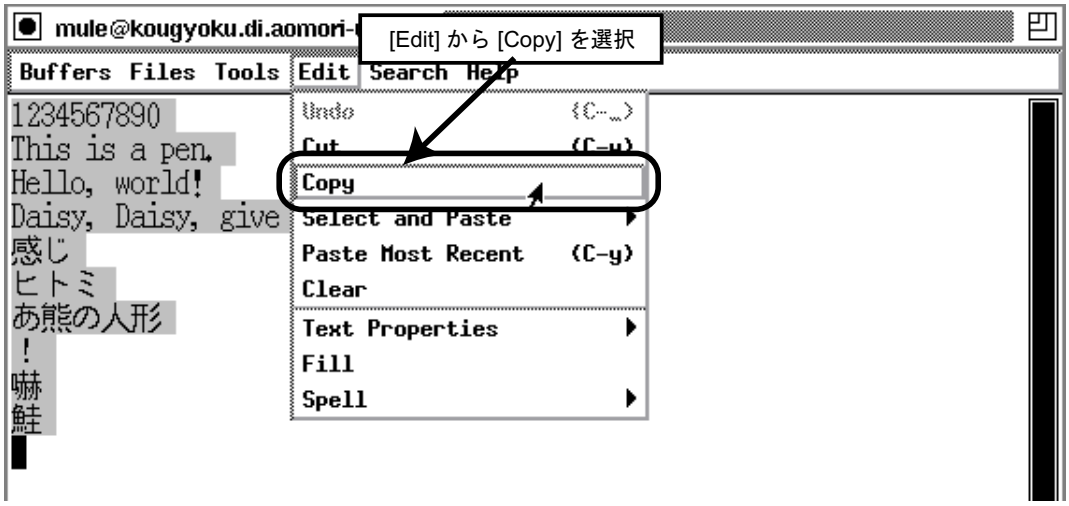
6.5.1 部分を選択してコピーして貼り付け

エディタには、部分を選択して、コピーして貼り付けたり、切り取ったりする機能が備わっている。Mule の場合を紹介しよう。

まず、部分を選択するには、次の図のようにマウスで ボタン 1 を押しながらドラッグしてやればよい。なお、選択したい範囲の最初の部分にカーソルを移動し、そこで「C-@」ないしは「C-スペース」と入力し、選択したい範囲の最後の部分までカーソルを移動しても同様の効果がある。

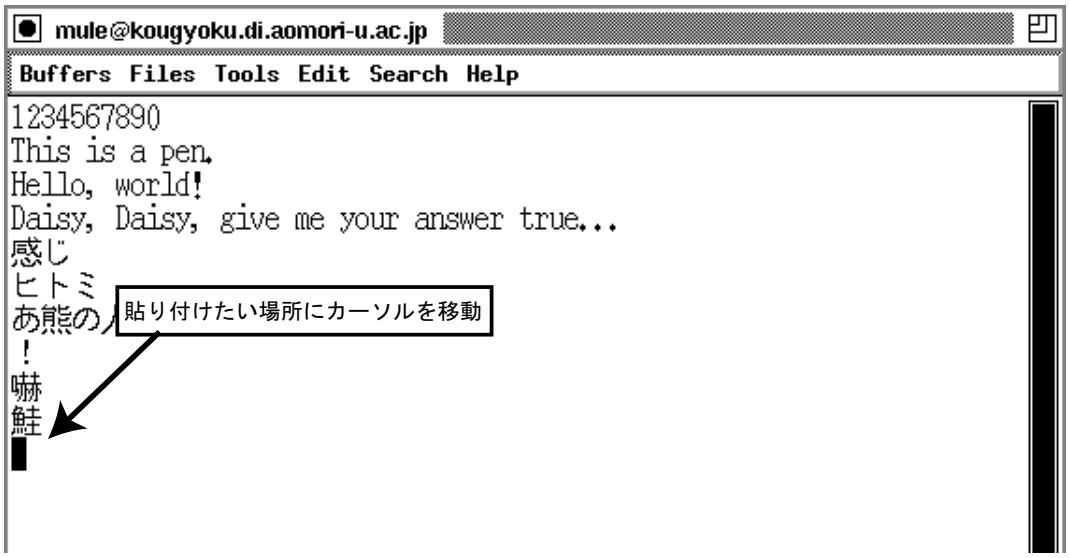


この部分をコピーする (コンピュータに記憶させる) には、メニュー・バーの [Edit] から [Copy] を選ぶ[3]。または、「M-w」と入力してもよい。

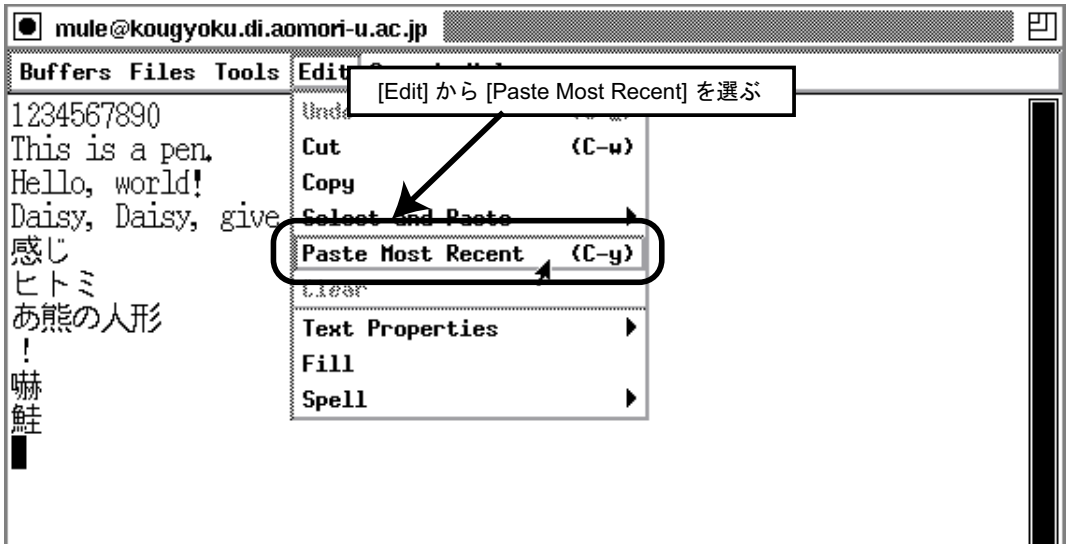


[3]本当は X Windows System でマウスを使って選択した場合には、自動的にここまで行われてしまうので不要である。

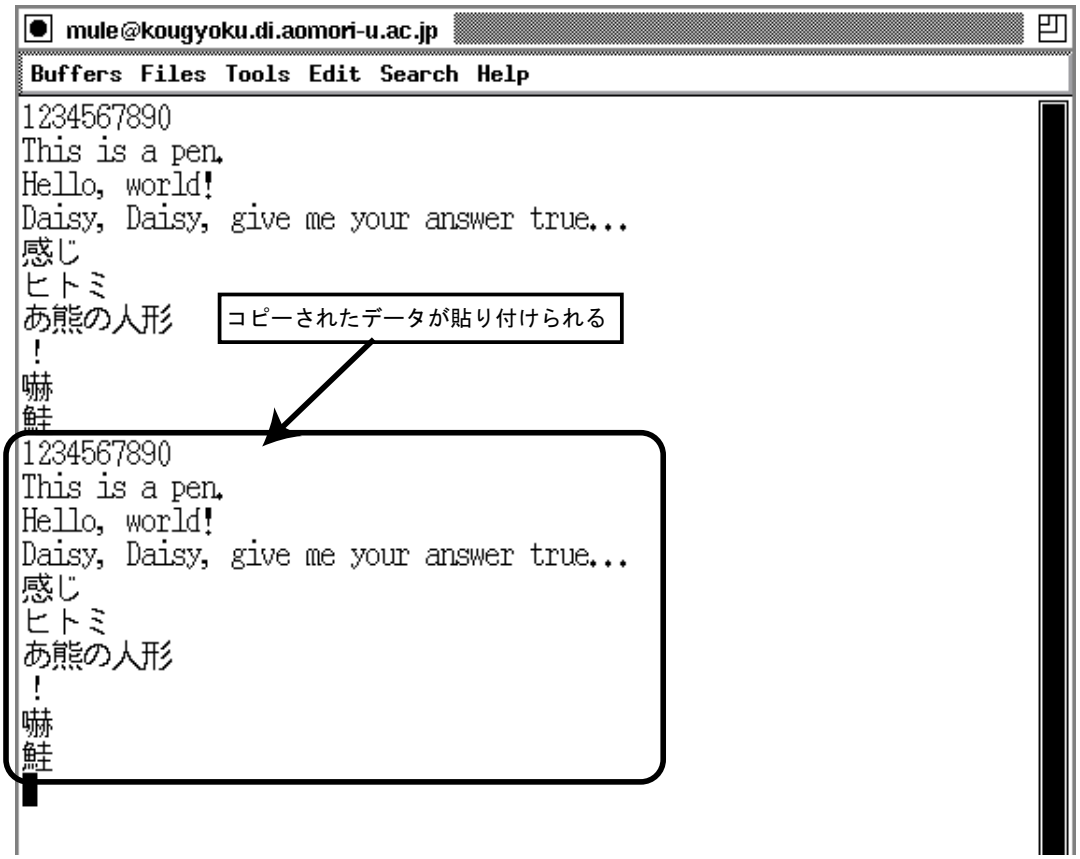
次に、今、記憶させたものを、貼り付けたいところにカーソルを移動する。



メニュー・バーの [Edit] から [Paste Most Recent] を選ぶ。または、「C-y と入力してもよい。

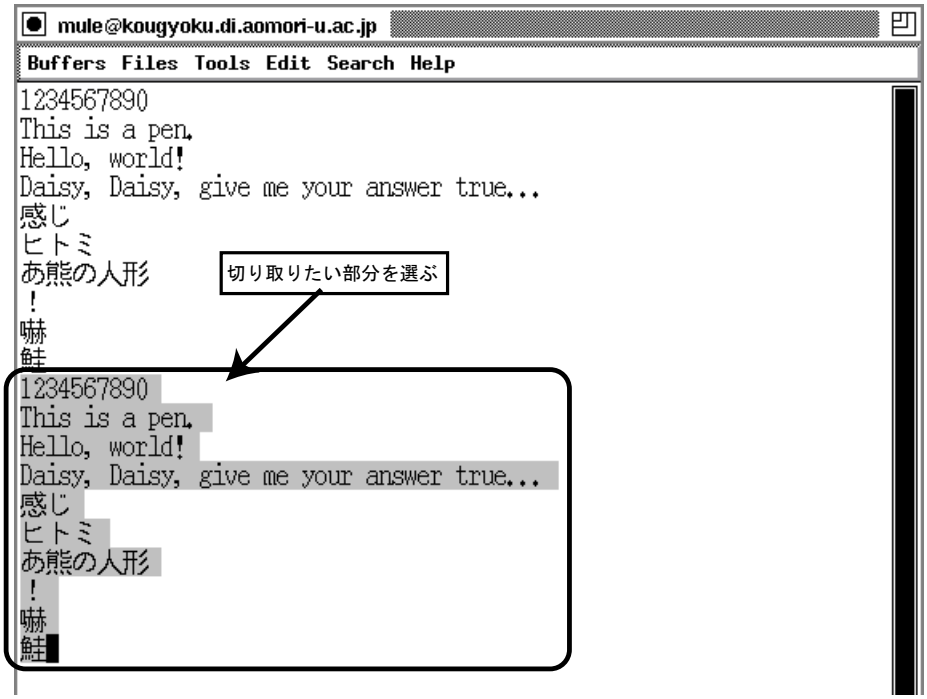


記憶された内容が、貼り付けられる。

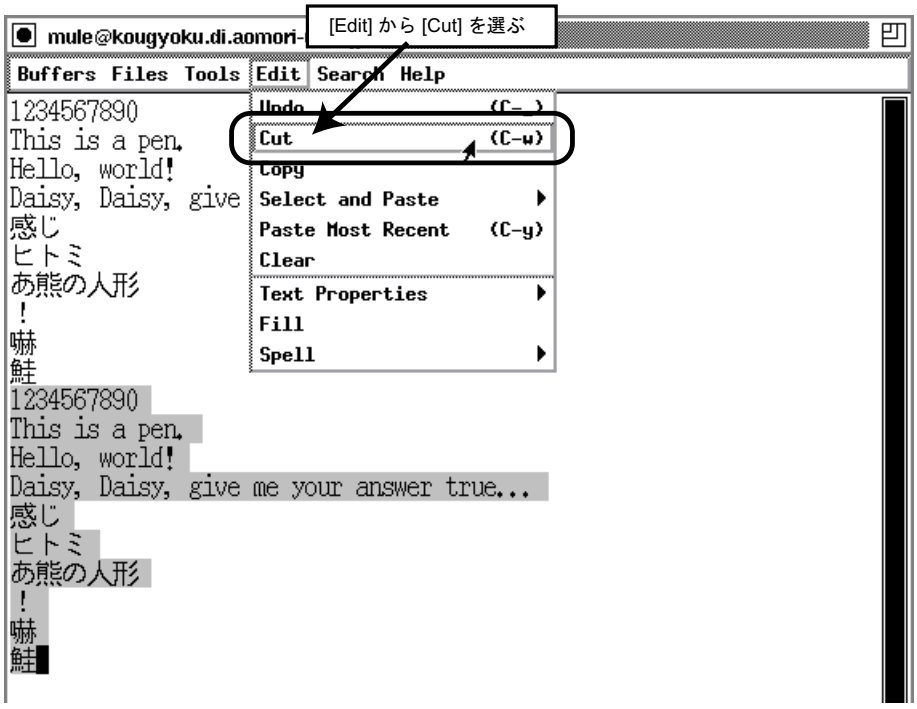


6.5.2 部分を選択して切り取り

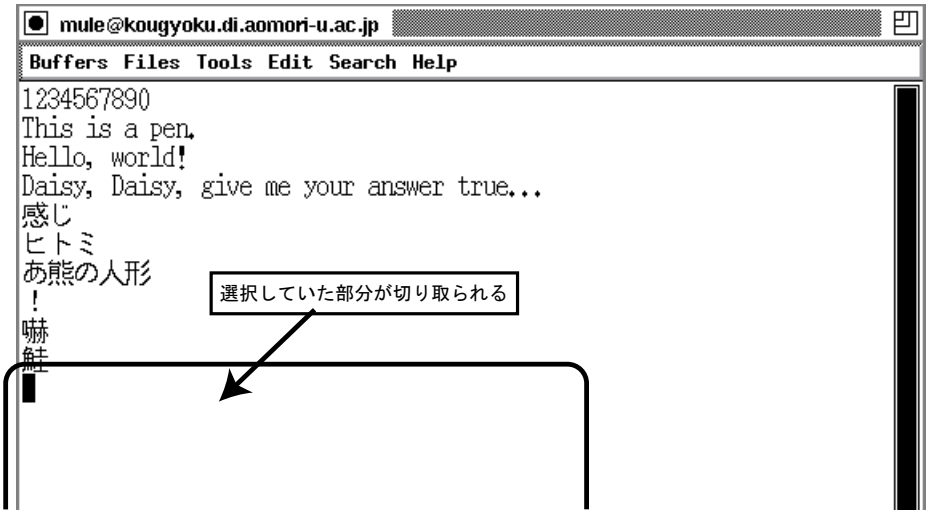
さきほどと同様に、切り取りたい部分を、マウスでドラッグして選択する。これは選択したい範囲の最初の部分にカーソルを移動し、そこで「C-@」ないしは「C-スペース」と入力し、選択したい範囲の最後の部分までカーソルを移動しても同様の効果がある。



この部分を切り取るには、メニュー・バーの [Edit] から [Cut] を選ぶ。または、「C-w」と入力してもよい。



選択した部分が、切り取られる。なお、この切り取った部分は、コピーして貼り付けるときと同様に、別な場所に貼り付けることもできる。

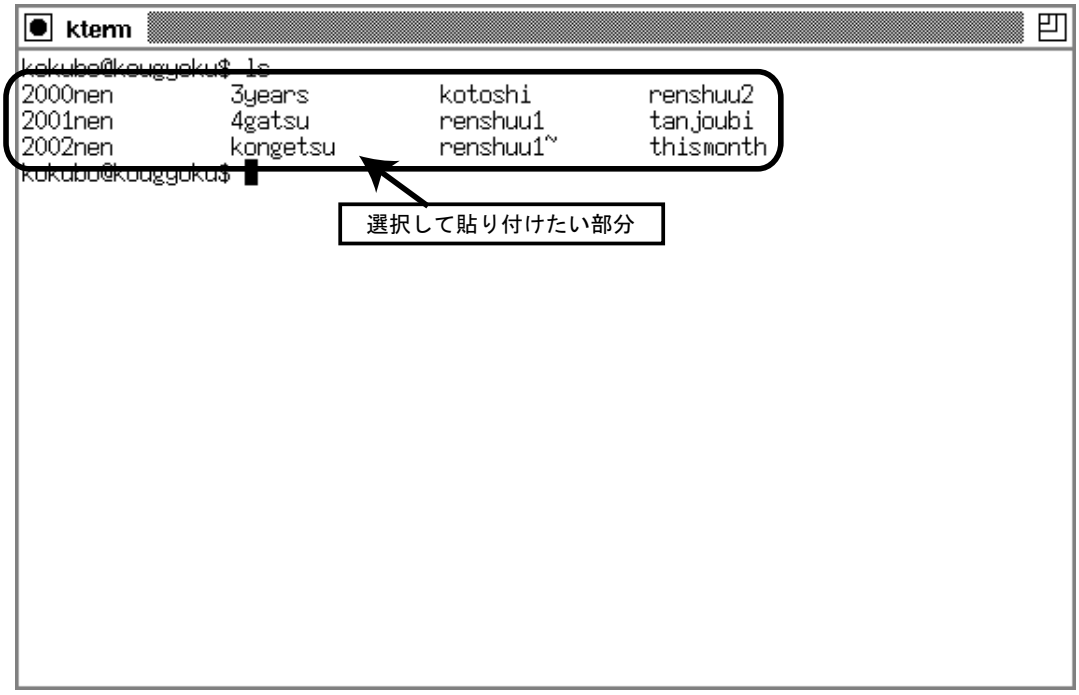


6.5.3 ウィンドウ間で選択して貼り付け

Mule と Kterm など、他の X Window System のアプリケーションとの間で選択して貼り付ける方法を紹介しよう。

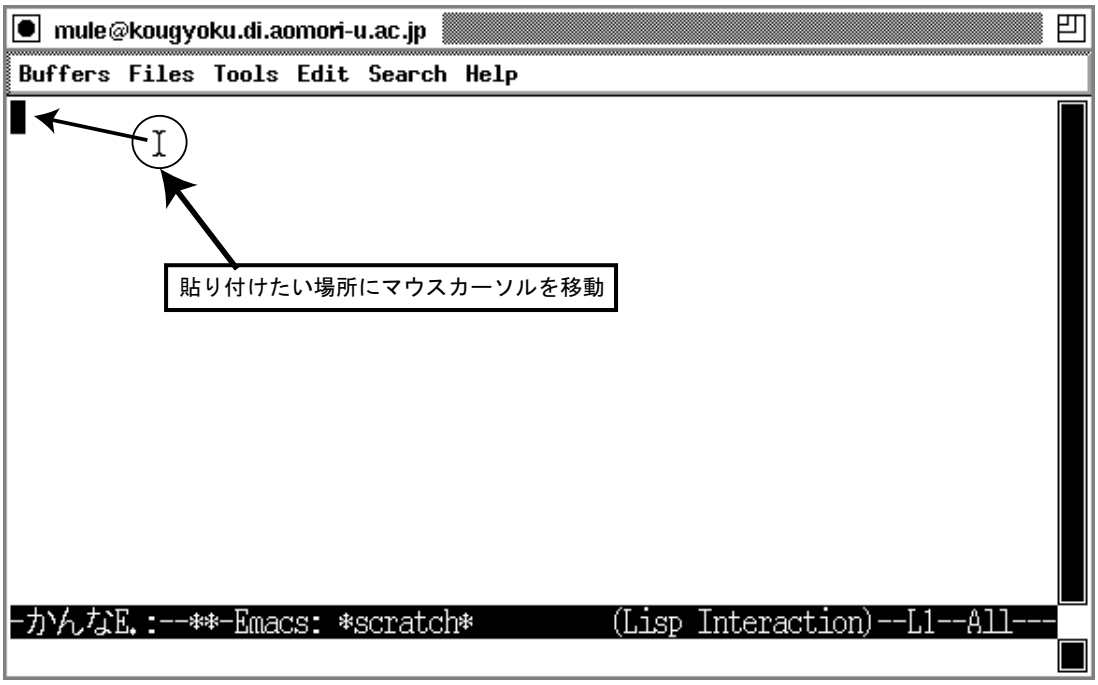
Kterm などでは、ボタン 1 でドラッグして選択、ボタン 2 で貼り付けが行われる。これは、 Mule と Kterm 間でも、 Mule と Mule 間でも、 Kterm と Kterm 間でも同様に行える。

具体的にやってみよう。まず、 Kterm の画面を開き、貼り付けたい元の部分を表示する。

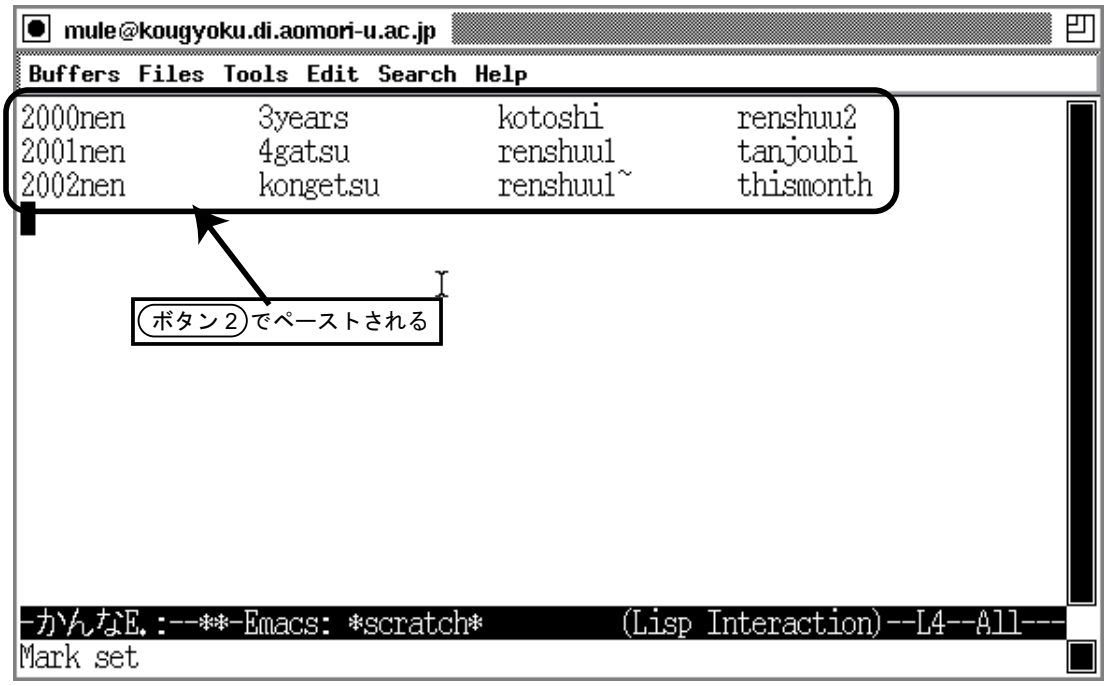


ボタン 1 を押しながらドラッグして、貼り付けたい部分を選択する。

Mule の画面で、貼り付けたい先の部分に**「マウス」**カーソルを移動する。



ボタン 2 を押すとペーストされる。



この方法で、 Mule と Mule 間、 Kterm と Kterm 間でも選択して貼りつけたりすることができる。

第7章　印刷

B 演習室には、 5 台のプリンタがある。これらのプリンタは、ネットワークにつながっていて、ネットワークを使ってそれぞれのマシンから印刷することができる。

プリンタはすべて PostScript の白黒プリンタで、PostScript 形式のファイルを送ると印刷することができる。

この章では、テキスト・ファイルの PostScript への変換や、印刷方法などについて紹介する。

7.1　テキスト・ファイルの PostScript への変換: a2ps-j

B 演習室のプリンタは、 PostScript プリンタで、ファイルを印刷するには、 PostScript 形式に変換する必要がある。

a2ps-j というコマンドを使うと、テキスト・ファイルを PostScript 形式に変換することができる。

例えば、「renshuu1」というテキスト・ファイルから、「renshuu1.ps」という PostScript 形式のファイルを作るには、次のようにする。

```
% a2ps-j renshuu1 > renshuu1.ps
```

ls で確認してみよう。確かに「renshuu1.ps」ができていることが確認できると思う。このようにして変換した PostScript ファイルは、1 枚に 2 ページ分が横向きに印刷される。

もしも、大きめに 1 枚に 1 ページ分を印刷したい場合には、次のようにして PostScript ファイルを作る。

```
% a2ps-j -p renshuu1 > renshuu1.ps
```


Oct 20 1999 12:25 unix14.txt Page 1

印刷

ファイルを印刷するには

B 演習室には、5 台のプリンタがある。これらのプリンタは、ネットワークにつながっていて、ネットワーク越しにそれぞれのマシンから印刷することができる。

プリンタはすべて PostScript の白黒プリンタである。これらのプリンタは、PostScript 形式のファイルを送ると印刷することができる。

この章では、テキスト・ファイルの PostScript への変換や、印刷方法などについて紹介する。

テキスト・ファイルの PostScript への変換: a2ps-j

B 演習室のプリンタは、PostScript プリンタなので、テキスト・ファイルを印刷するには、これを PostScript 形式に変換する必要がある。

a2ps-j というコマンドを使うと、テキスト・ファイルを PostScript 形式に変換することができる。

例えば、exam1 というテキスト・ファイルから、exam1.ps という PostScript 形式のファイルを作るには、次のようにする。

% a2ps-j exam1 > exam1.ps

ls で調査してみよう。確かに exam1.ps ができていることが確認できると思う。このようにして変換した PostScript ファイルは、1 枚に 2 ページ分が裏同志に印刷される。

もしも、大きめに 1 枚に 1 ページ分を印刷したい場合には、次のようにして PostScript ファイルを作る。

% a2ps-j -p exam1 > exam1.ps

a2ps-j の使い方をまとめると次のようになる。

a2ps-j 「テキスト・ファイル名」 > 「PostScript ファイル名」
a2ps-j -p 「テキスト・ファイル名」 > 「PostScript ファイル名」

PostScript ファイルの印刷: lpr

PostScript ファイルができたら印刷してみよう。印刷するには、lpr コマンドを使う。

lpr コマンドは、次のように使う。

lpr -P「プリンタ名」 「PostScript ファイル名」

B 演習室のプリンタの名前は、pr01〜pr05 だ(どんなプリンタが使用可能かは、/etc/printcap に書かれており、jless /etc/printcap などとすれば、中身を読むことができる。)。

では、次のようにして、pr01 というプリンタから、exam1.ps を印刷してみよう。

% lpr -Ppr01 exam1.ps

他のプリンタに出すには、pr01 の部分を、pr02 とか、pr03 などに変えればよい。

テキスト・ファイルから直接印刷

ここまでのところでは、テキスト・ファイルを一旦 PostScript ファイルに変換してから印刷したが、これらの作業は一括して行なうことができる。

unix14.txt

a2ps-j の使い方をまとめると次のようになる。

1 枚に 2 ページ分印刷したいとき

a2ps-j **「テキスト・ファイル名」** >**「PostScript ファイル名」**

1 枚に 1 ページ分印刷したいとき

a2ps-j -p **「テキスト・ファイル名」** >**「PostScript ファイル名」**

7.2 PostScript ファイルの印刷: lpr

PostScript ファイルができたら印刷してみよう。印刷するには、次のようにして lpr コマンドを使う。

lpr -P**「プリンタ名」** **「PostScript ファイル名」**

なお B 演習室のプリンタの名前は、 pr01〜pr05 である[1]。

では、次のようにして、 pr01 というプリンタから、「renshuu1.ps」を印刷してみよう。

[1]どんなプリンタが使用可能かは、 /etc/printcap に書かれており、jless /etc/printcap などとすれば、中身を読むことができる。

```
% lpr -Ppr01 renshuu1.ps
```

他のプリンタに出すには、 pr01 の部分を、 pr02 とか、pr03、… などと変えればよい。

7.3 テキスト・ファイルから直接印刷

ここまでのところでは、テキスト・ファイルを一旦 PostScript ファイルに変換してから印刷したが、これらの作業は一括して行なうことができる。

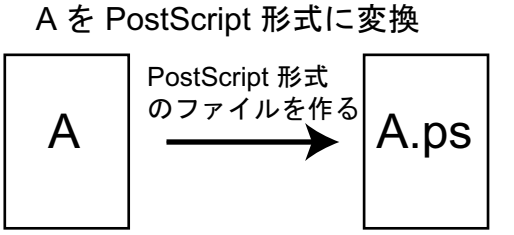
具体的には、次のようにする。

```
% a2ps-j renshuu1 | lpr -Ppr01
```

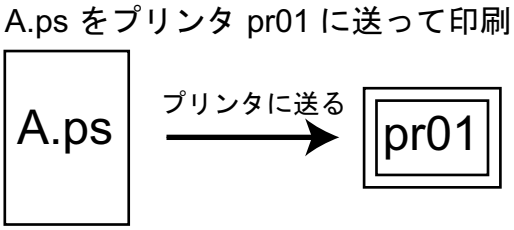
このように「|」を用いると、 2 つのコマンドをまとめて一括することができる。

テキスト・ファイル印刷の流れ

a2ps-j A > A.ps

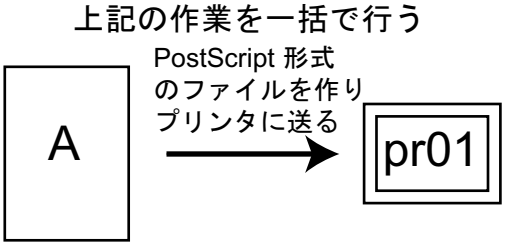


lpr -Ppr01 A.ps



---

a2ps-j A | lpr -Ppr01



7.4 印刷状況の確認と取消: lpq と lprm

ほとんどの場合、印刷は高速に行なわれるため、現実には取り消すことは不可能な場合が多いだろう。しかし、長い文章を印刷したり、紙がつまったりして、印刷状況を確認したり、印刷を取り消したくなる場合もある。

7.4.1 印刷状況の確認: lpq

現在の印刷状況を確認するには、 lpq を使う。使い方は、次のようになる。

```
lpq -P「プリンタ名」
```

たとえば、プリンタ pr01 の印刷状況を調べたい場合には、次のようにすればいい。

```
% lpq -Ppr01
pcux48.edu3.aomori-u.ac.jp: sending to edu3pr05
Rank   Owner      Job  Files                                 Total Size
1st    ei00000    259  renshuu1.ps                           22719 bytes
2nd    ei00000    260  (standard input)                      792049 bytes

no entries
```

このとき表示されるものの意味は以下の通りである。

Rank	印刷の順番
Owner	印刷を実行しているユーザ
Job	印刷ジョブの番号
Files	ファイル名
Total Size	ファイルの大きさ

なお、「a2ps-j renshuu1 | lpr -Ppr01」のように、一括で印刷を行なっている場合、ファイル名は「(standard input)」となる。

7.4.2 印刷の取り消し: lprm

印刷を取り消すには、次のようにして lprm を使う。

```
lprm -P「プリンタ名」 「印刷ジョブの番号」
```

例えば、上の例では、印刷ジョブの番号は「259」である。これを取り消すには、次のようにすればいい。

```
% lprm -Ppr01 259
dfA259pcux00.edu3.aomori-u.ac.jp dequeued
cfA259pcux00.edu3.aomori-u.ac.jp dequeued
```

これで印刷はキャンセルされる。

7.5　PostScript ファイルの表示: ghostview

以降の節は、おまけである。

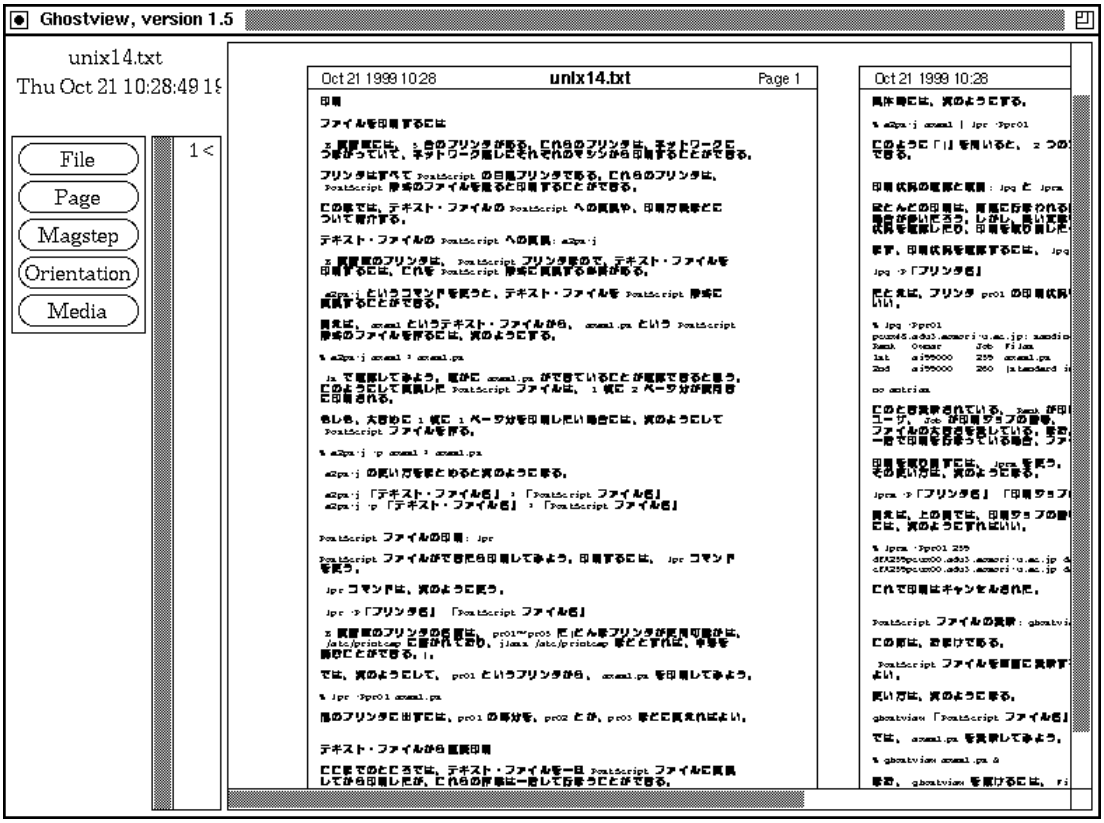
PostScript ファイルを画面に表示するには、 ghostview コマンドを使えばよい。

使い方は、次のようになる。

ghostview 「PostScript ファイル名」 &

では、「renshuu1.ps」を表示してみよう。

```
% ghostview renshuu1.ps &
```



なお、 ghostview を抜けるには、 File メニューから、 Quit を選べばよい。

7.6 PostScript とはどんなものか?

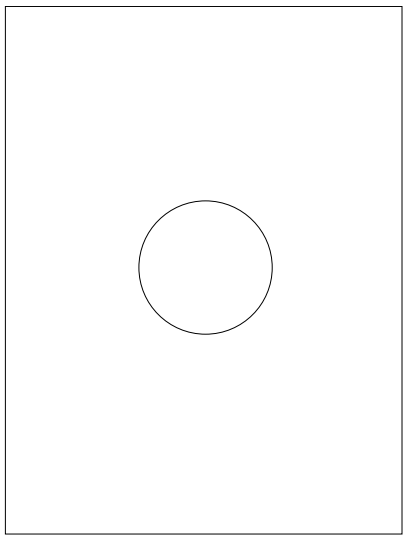
PostScript は、一種のプログラムで、手で書くこともできる。

例えば「mule circle.ps &」と Mule を起動して、「circle.ps」というファイルに次のプログラムを入力して保存してみよう。

circle.ps

```
%!PS-Adobe-1.0
%%Title: Circle
%%Pages: 1
300 400 100 0 360 arc
stroke
showpage
%%Trailer
```

これを ghostview で表示したり、 lpr で印刷したりすると、確かに円が描かれているのがわかると思う。



このプログラムの意味は別にわからなくてもかまわないが、一応解説しておこう。まず、「%」で始まる行は、コメントである。それから、「300 400 100 0 360 arc」の部分で、紙の左上から x 方向に 300 ポイント、 y 方向に 400 ポイントの位置を中心に、半径 100 ポイント、 0〜360 度の範囲の円弧を指定している。次の「stroke」で線を引くように指定。そして最後の「showpage」が描画である。

PostScript は奥がとても深いので、ここできちんとした説明はしない。ただ、 PostScript プリンタにデータを送るときに、コンピュータはこのようなプログラムに画像データを変換していることは、知っておいて損はない。

この章で紹介したコマンド

PostScript への変換と印刷

a2ps-j : テキスト・ファイルを PostScript に変換

使い方: a2ps-j「テキスト・ファイル名」>「PostScript ファイル名」

大きく印刷するには、 -p オプションを付ける。

lpr : PostScript ファイルの印刷

使い方: lpr -P「プリンタ名」「ファイル名」

なお、 PostScript への変換と印刷を一括で行なうには、

a2ps -j「ファイル名」| lpr -P「プリンタ名」

lpq : 印刷状況の確認

使い方: lpq -P「プリンタ名」

lprm : 印刷の取消

使い方: lprm -P「プリンタ名」「印刷ジョブの番号」

ghostview : PostScript ファイルの表示

使い方: ghostview「PostScript ファイル名」 &

第8章　ディレクトリとパス

ファイル・システムの全体像

UNIX システムでは、ファイル・システムは、ツリー状の階層化ディレクトリ構造をしている。
これは簡単に言うと次のようになる。

1. たくさんあるファイルを、種類ごとに「**ディレクトリ**」と呼ばれる箱に分けてしまうことで整理する。
2. **ディレクトリ**の中にも、また**ディレクトリ**を作ることができ、更に細かく分類して整理できる。
3. **ディレクトリ**の構造図を書くと、木が枝をひろげたよう (ツリー状) に見える。

このようなディレクトリ構造は、後に MS-DOS や Windows などにも採り入れられた。Windows では、「ディレクトリ」のことを「フォルダ」と呼んでいる。

UNIX システムには、いろいろなファイルやコマンドがある。ディレクトリの中を自由に移動できるようになれば、これらのことを詳しく知ることができるようになる。また、自分のファイルをディレクトリに分けてしまっておくと、きれいに整理されて、便利に使うことができるようになる。
この章では、ディレクトリの移動、作成、消去などについて紹介しよう。

8.1　UNIX システムのディレクトリ構造

8.1.1　ディレクトリの全体像

最初に UNIX システムの全体像を紹介してしまう。
次のページの図 8.1 は、 UNIX のディレクトリ構造の主な部分を書いたものである。これには、次のようなディレクトリがある。

- **ルート・ディレクトリ** /

  一番上のディレクトリを「**ルート・ディレクトリ**」といい、「/」で表す。なお、「ルート – root」というのは「(木の) 根」のことだ。

  ルートの下には、「`home`」、「`bin`」、「`etc`」など、いろいろなディレクトリが、枝をひろげている。そして、それらのディレクトリの先にも、またディレクトリがあり、さらに枝がひろがっている。

これらがツリー状に見える。絵としては、木が上下さかさまに描かれていると思ってくれればいい。

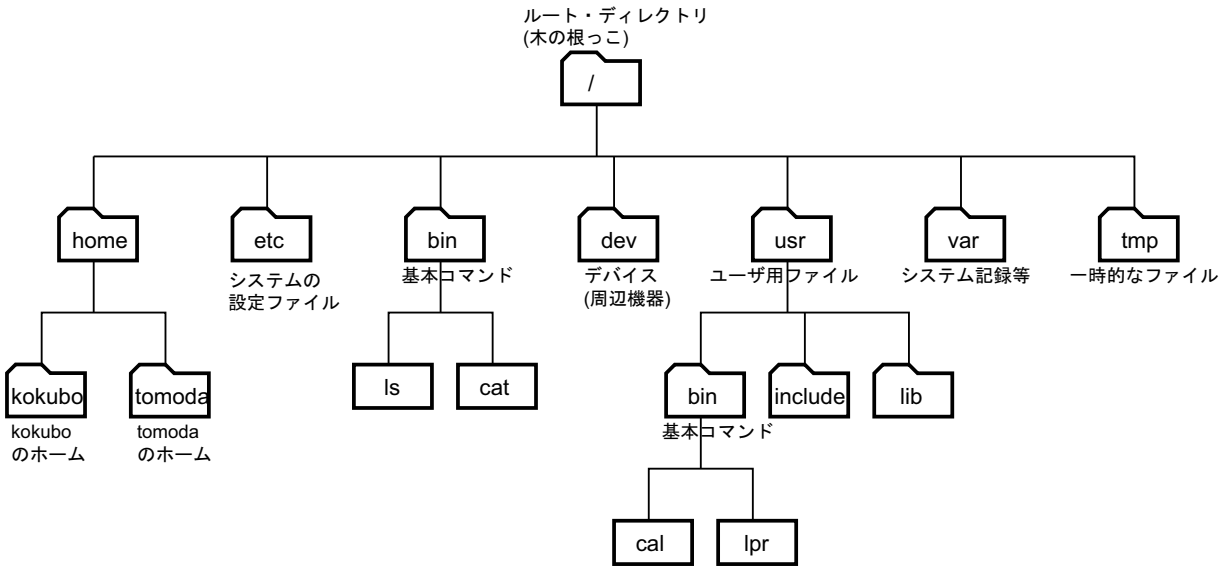


図 8.1: UNIX システムのディレクトリ構造

では、ルート・ディレクトリの下を一つずつ説明していこう。

- **ホーム・ディレクトリ** home

まず、「/」の下には、「home」というディレクトリがあり、この中にユーザ ID と同じ名前のついたディレクトリがある。これらは、 UNIX システムで「**ホーム・ディレクトリ**」と呼ばれている。

たとえば上の図では、 kokubo さんのホーム・ディレクトリは、「/」の下の「home」の下の「kokubo」になる。これを、「/home/kokubo」と表す。このようにディレクトリの区切りを表すマークは「/」である。

また同様に、 tomoda さんのホーム・ディレクトリは、「/」の下の「home」の下の「tomoda」で、これを「/home/tomoda」と表す。

login した直後、それぞれのユーザは、自分のホーム・ディレクトリにまず入る。これまでの演習で作ったファイルも、みんな自分のホーム・ディレクトリにあるはずである。

上の例では、 kokubo さんは login すると、「/home/kokubo」に入る。そして、これまでの演習で kokubo さんが作ったファイルは、そこに置いてあるということだ。

- **bin**

「/」の下の「bin」、つまり「/bin」には、UNIX システムの一番基本的なコマンド (たとえば ls や cat など) が置いてある。

「bin」というのは「バイナリ – binary」の略で、本来は文字ではないデータが入っているファイルのことを意味する[1]。

- **etc**

  「etc」は「エトセトラ – etc.」である。UNIX システムのシステム設定用のファイルが置かれている。

- **tmp**

  「tmp」は「テンポラリ – temporary」である。主に、システムが一時的にファイルを作るときに、ここを使う。

- **usr**

  「usr」は「ユーザ – user」という意味だ。

  この中は更に分かれていて、「/usr/bin」にはユーザが使う各種コマンド、「/usr/include」には C 言語のヘッダ・ファイル、「/usr/lib」にはユーザ・コマンドが使うライブラリと呼ばれているデータなどが置かれている。

- **var**

  「var」は「変化する – variable」という意味だ。login をはじめとするシステムの記録、メールなど、日々変化するデータが置かれている。

[1]マシン語で書かれたコマンドや、画像、音声データなどが、本来の意味ではバイナリである

8.1.2　B 演習室の実際

ここまで紹介してきたのは、あくまでも標準のスタイルである。ディレクトリの構造は、システムの管理者の好みに応じて変えることもできる。

B 演習室では、ホーム・ディレクトリの様子は、図 8.2 のように変更されている。なお、ホーム・ディレクトリ以外は標準の構成のままである。

図を見てもらうとわかるが、「/」の下に「home」があるところまでは一緒である。違うのは、更にこの中が入学年度別に「ei00」、「ei01」、「ei02」、「ei03」… と切り分けてあるところだ。

そして、この中にそれぞれのユーザのホーム・ディレクトリがある。

たとえば、 ei00000 さんのホーム・ディレクトリは「/home/ei00/ei00000」だ。同様に ei01001 さんは「/home/ei01/ei01001」となる。

自分のホーム・ディレクトリの中には、これまでの演習で作った「kongetsu」、「renshuu1」などのファイルが入っている。

また、後の方で説明するが、ユーザはホーム・ディレクトリの中に「report」や「prog」などのディレクトリを、自分で自由に作ってファイルを整理することができる。

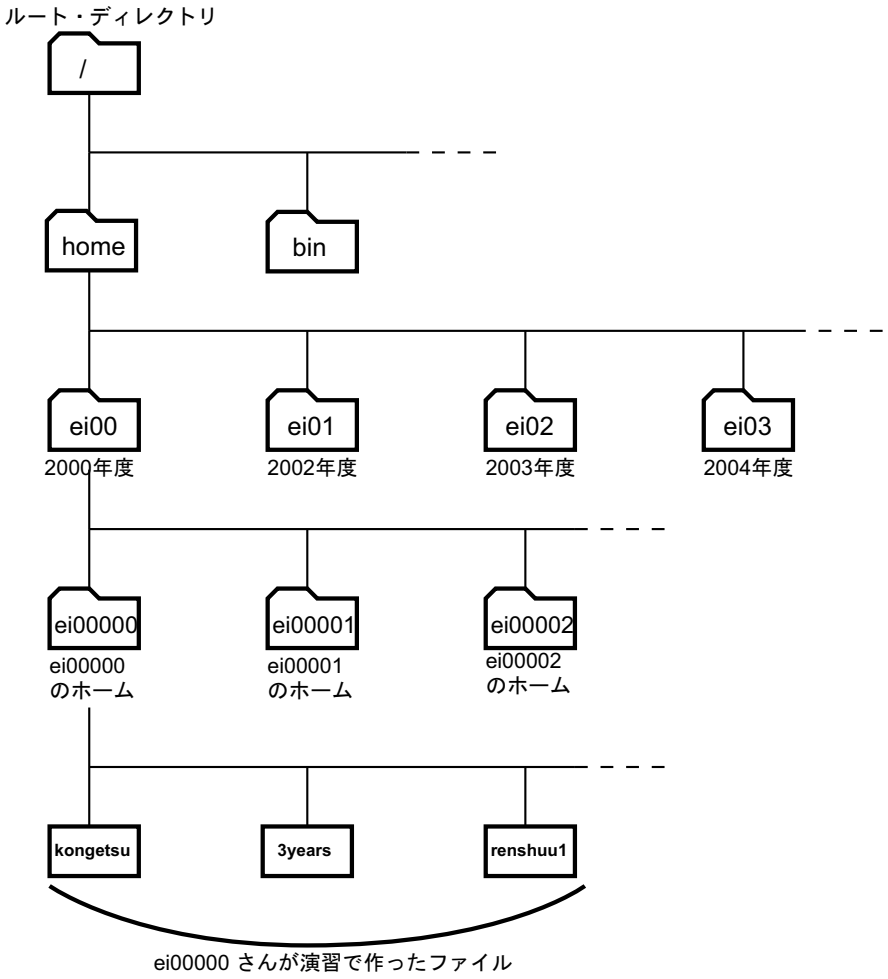


図 8.2: B 演習室の実際のホーム・ディレクトリの構造

8.2 パス

UNIX システムでは、ファイルやディレクトリの場所を指定するのに、「**パス**」を使う。

「パス – path」というのは、「通り道」や「経路」という意味だ。パスには、「**絶対パス**」と「**相対パス**」がある。

このパスをマスターすることで、好きな場所に移動して UNIX の中を探検したり、ファイルを好きな場所にコピーしたりすることができる[2]ようになる。

パスは UNIX システムをマスターする上で、最重要ポイントの一つだ。

8.2.1 根っこからの通り道: 絶対パス

既に、ここまでのところで紹介してしまったが、「/home/kokubo」のように、一番上のルート・ディレクトリから、ひとつずつ順番に書いて表す方法を、**「絶対パス」**と呼ぶ。

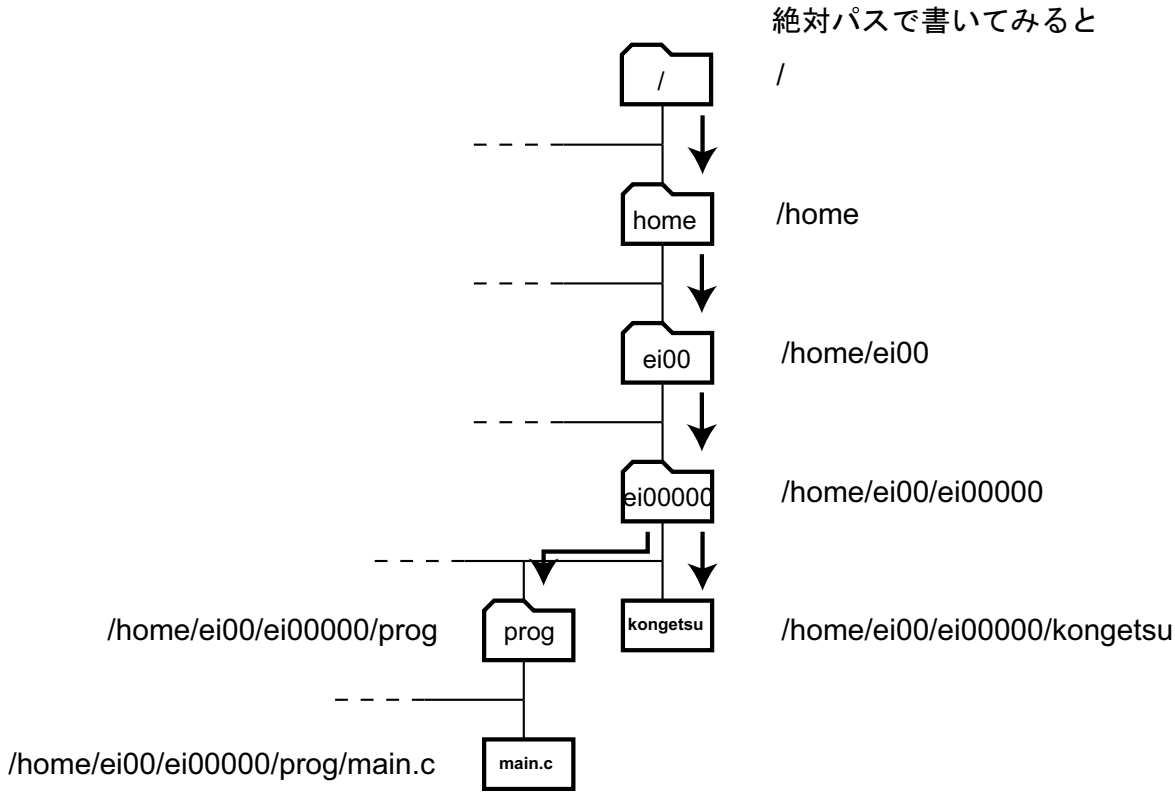


図 8.3: 絶対パスでの表し方

図 8.3 を使って説明しよう。まず、一番上にルート・ディレクトリがあり、これは絶対パスで表すと「/」になる。

「/」の下には「home」があって、これは絶対パスでは「/home」となる。

その下は学年ごとにディレクトリが用意されていて、「ei00」は絶対パスでは「/home/ei00」となる。

[2]セキュリティ上、いじれないようになっている場所もある。

たとえばその中に、ei00000 さんのホーム・ディレクトリがあって、これは「/home/ei00/ei00000」だ。

そして、 ei00000 さんは、自分のホーム・ディレクトリの中に「kongetsu」というファイルを作ることができて、これは「/home/ei00/ei00000/kongetsu」と表される。同様に「renshuu1」は「/home/ei00/ei00000/renshuu1」と表される。

このように一番上の根っこ (ルート・ディレクトリ) から順番に、書いていくのが絶対パスである。

ここで、注意しておいて欲しいのは、UNIX のパスでは「ファイル」も「ディレクトリ」も全く同じように表されるということだ。たとえば、図 8.4 の「kongetsu」はファイルである。これを絶対パスで表すと「/home/ei00/ei00000/kongetsu」になる。ところが、「kongetsu」がディレクトリであっても絶対パスで書けば同様に「/home/ei00/ei00000/kongetsu」になる。

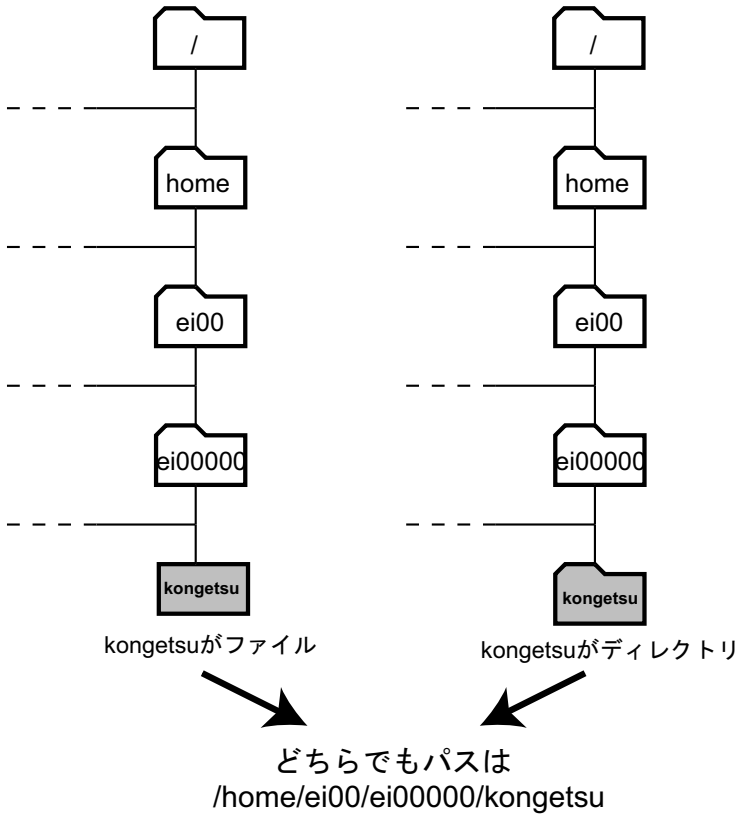


図 8.4: ファイルでもディレクトリでもパスは同じ

これは UNIX では、普通のファイル、ディレクトリ、周辺機器 (画面、プリンタ、ネットワーク・カード、その他) などのさまざまなものを、統一的に「ファイル」として取り扱っているためである。このようになっていると、プログラムを書いたりするときに、これらのさまざまなものをだいたい同じような方法で取り扱うことができるという利点がある。

8.2.2 自分がいるところが中心: 相対パス

絶対パスを使って、ルート・ディレクトリから順番に書いていけば、どんなディレクトリでも表せる。ところが、ディレクトリが下の方になってくると、どんどん長くなっていき面倒になる。

そこで、 UNIX システムでは、**「相対パス」**という、もう一つの方法が用意されている。相対パスでは、俺が世界の中心という感じで、「自分が今いるところ」を基準にして指定する。

では、図 8.5 を見ながら説明しよう。

まず、 ei00000 さんは、今自分のホーム・ディレクトリである「/home/ei00/ei00000」にいるとしよう。

この中に「kongetsu」というファイルがあるとする。これは相対パスでは単に「kongetsu」と書く。絶対パスだと「/home/ei00/ei00000/kongetsu」と長くなってしまうが、相対パスだと簡単になる。

また、ホーム・ディレクトリの中に「prog」というディレクトリがあるとしよう。これも相対パスでは「prog」となり、とても簡単に書ける。

更に、「prog」の中に、「main.c」というファイルがあるとする。これは相対パスでは「prog/main.c」である。

このように相対パスとは、自分が今いるところから、順番にパスを書いていく 方法である。

なお、絶対パスと相対パスの見分け方は簡単で、最初が「/」ではじまっていれば絶対パス である。

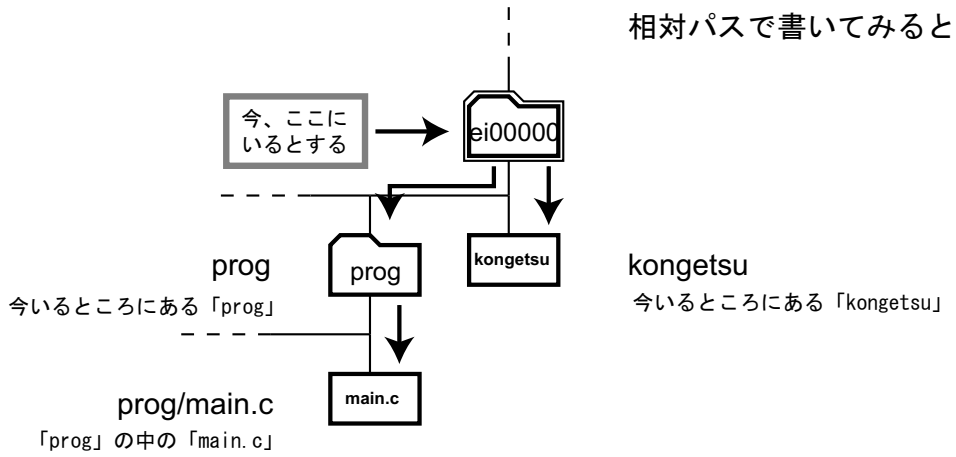


図 8.5: 下にあるファイルやディレクトリを相対パスで表す

自分が今いるディレクトリ:「.」

自分が今いる場所は、相対パスでは「.」と、ピリオド 1 つで表わす。

図 8.6 を見てみよう。今、「/home/ei00/ei00000」にいるとする。ここは相対パスで表すと「.」となる。

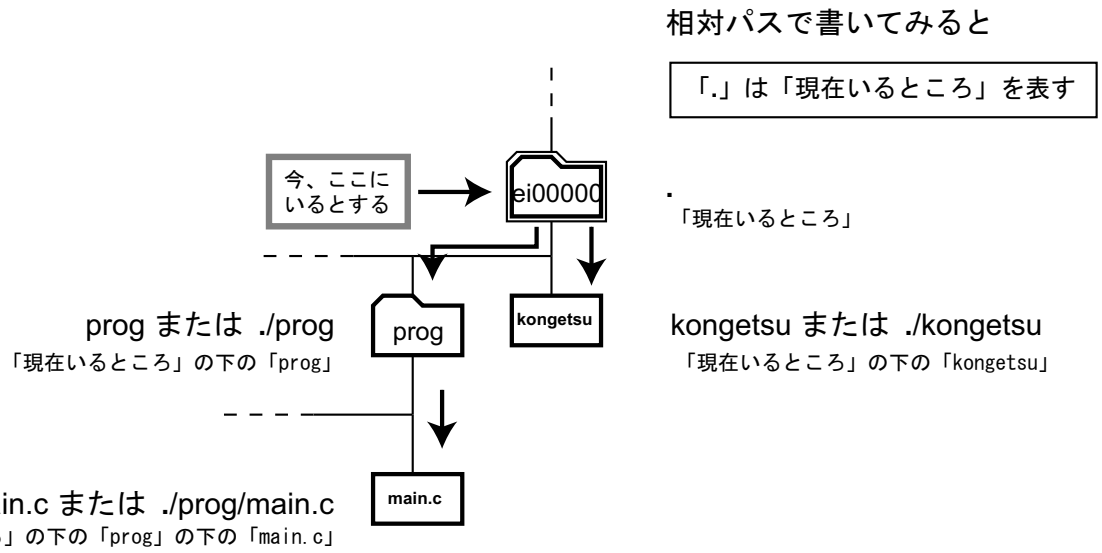


図 8.6: 自分のいるディレクトリを相対パスで表す

また、さきほど「kongetsu」は相対パスで「kongetsu」と書くと紹介した。これは、「現在いるところ」の下の「kongetsu」と見ることもできて、「./kongetsu」というふうにも表せる。

同様に「prog」は「./prog」、「prog/main.c」は「./prog/main.c」とも表せる。

1 つ上のディレクトリ:「..」

ここまでは自分よりも下のディレクトリの表し方を紹介してきた。では、逆に上の方はどうなるのかを紹介しよう。

図 8.7 を見てみよう。UNIX システムでは、「一つ上」のディレクトリは「..」と表す。つまり、自分のホームの一つ上の「/home/ei00」は、「..」と表される。

この**「ピリオド 2 つで、 1 つ上」**は、とてもよく使うので覚えておこう。

では、 2 つ上はどうか？

1 つ上が「..」で、ディレクトリの区切りが「/」なので、2 つ上は「../..」となる。つまり、「/home」は、相対パスでは「../..」となる。同様にルート・ディレクトリ「/」は、相対パスでは「../../..」である。

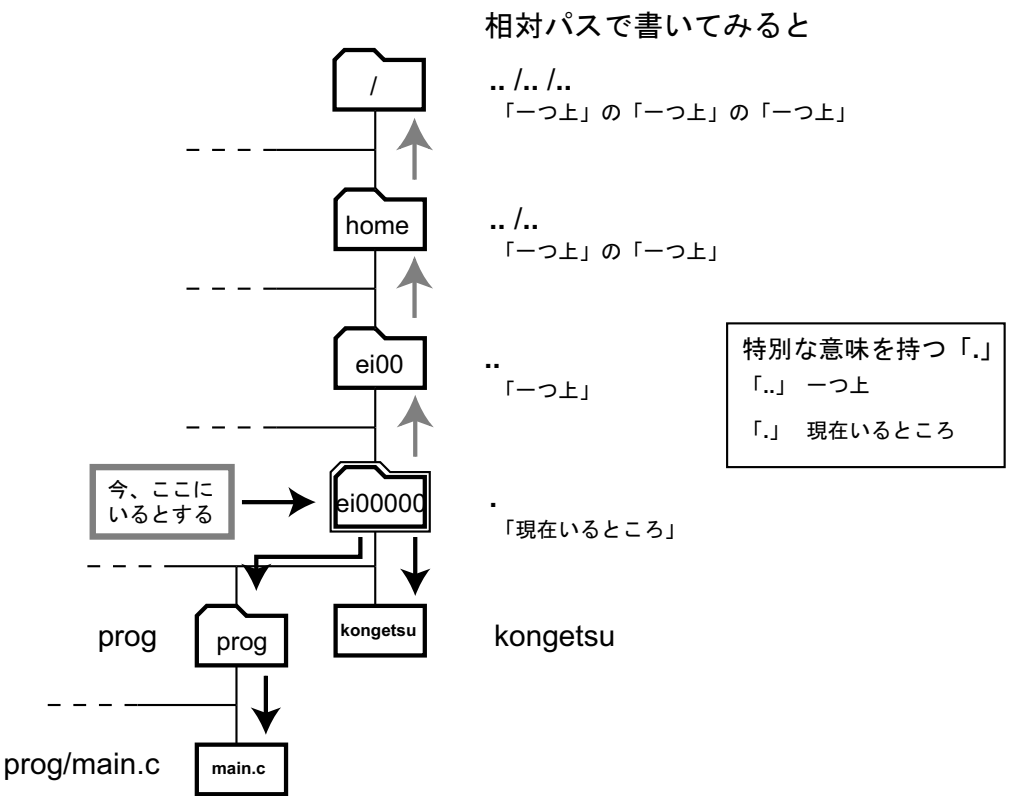


図 8.7: 上にあるファイルやディレクトリを相対パスで表す

自分のいる場所によって相対パスは変わる

ここまでの例は、あくまで「/home/ei00/ei00000」にいる場合の話である。

相対パスは自分が今いるところを中心に指定するので、別なディレクトリにいる場合、話が全く変わってくる。

たとえば、図 8.8 のように「/home/ei00/ei00000/prog」にいるとしよう。

今度は、自分のホーム・ディレクトリは一つ上にあり、これは相対パスでは「..」となる。

また、「main.c」は、今いるところのすぐ下にあるので、相対パスでは「main.c」になる。

このように相対パスは、現在どこのディレクトリに自分がいるかによって変わってしまう。

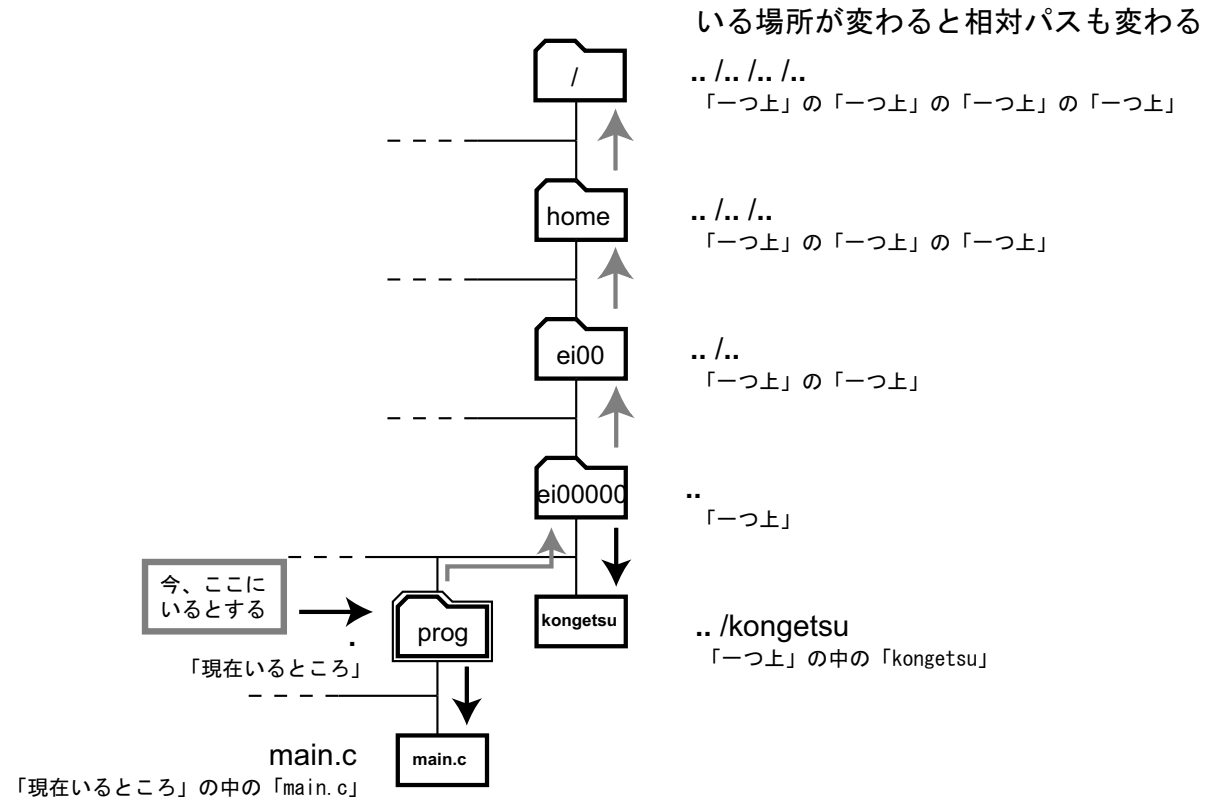


図 8.8: prog の中にいるときの相対パス

8.2.3 絶対パスと相対パス、どっちがお特?

UNIX システムでは、絶対パスで指定しても、相対パスで指定しても、効果は全く同じである。

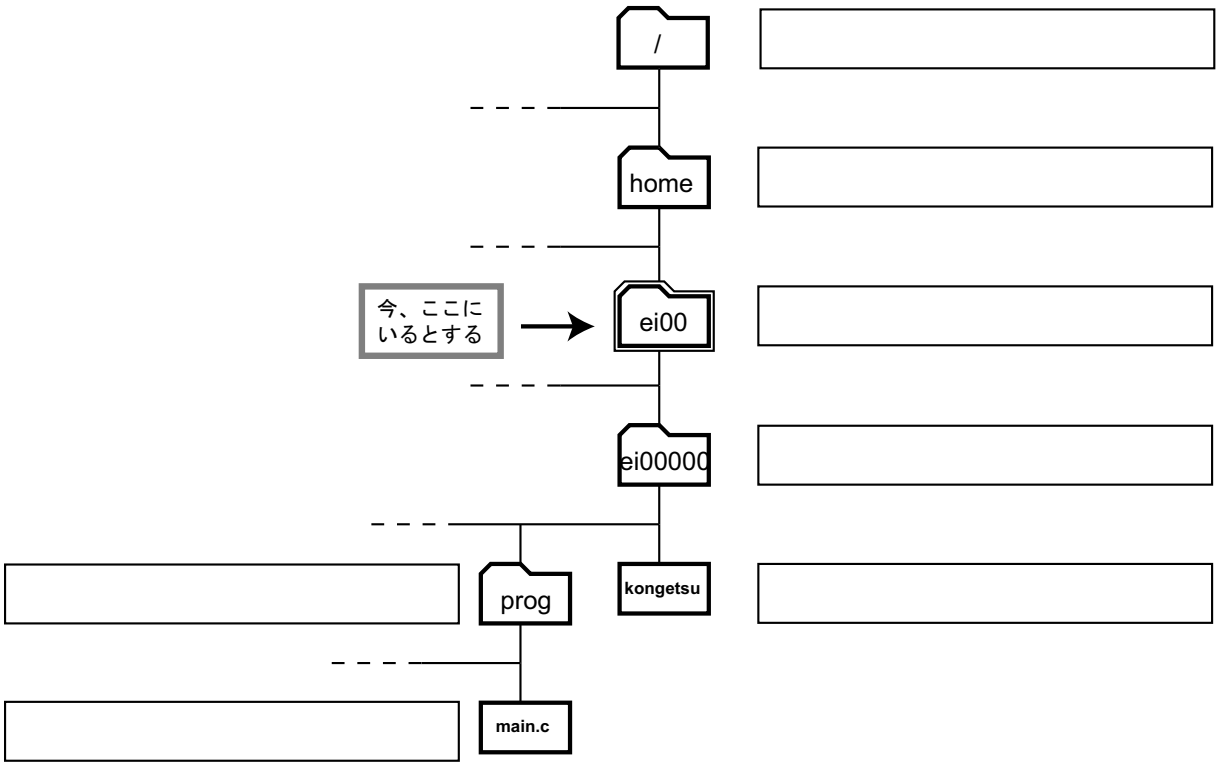
だから、自分で好きな方を使っていい。

ただし、自分が今いるところの近くを指定するには「相対パス」を、ルート・ディレクトリの近くを指定するには「絶対パス」を使った方が簡単である。

このように使い分けるので、両方を知っておいた方が便利だ。

☆**練習**☆

下の図にそれぞれ相対パスを記入しよう。



解答はこの章の最終ページにある。

8.3 ディレクトリ操作コマンド

では、具体的に、コマンドを使ってディレクトリを移動したり、ディレクトリを作ったりしてみよう。

8.3.1 現在いるディレクトリの表示: pwd

login した直後にいる場所が、自分のホーム・ディレクトリであると、前の方で説明した。では、それがどこか見てみよう。

今いるディレクトリを表示するには、 pwd コマンドを使う。pwd は「**p**rint **w**orking **d**irectory」の略で、「今お仕事をしているディレクトリを表示」という意味だ。

使い方は、単に次のように打てばいい。

```
pwd
```

さて、図 8.9 のようなディレクトリ構造だとしよう。ei00000 さんは、今 login した直後で、ホーム・ディレクトリの「/home/ei00/ei00000」にいるとしよう。このとき、 pwd を実行すると、

```
% pwd
/home/ei00/ei00000
%
```

となり、 今いるのが「/home/ei99/ei99000」だと表示される。

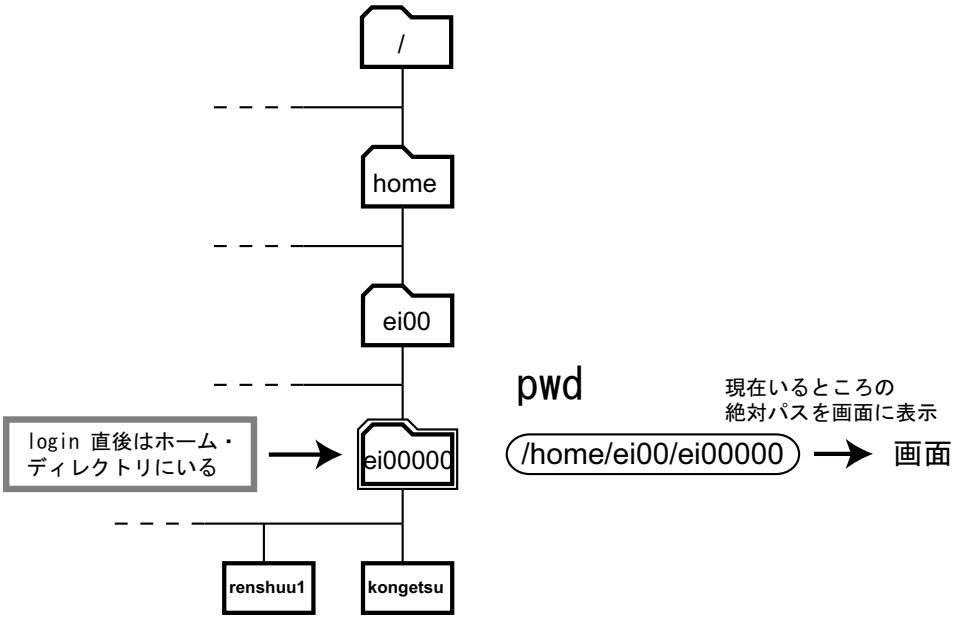


図 8.9: login 直後の ei00000 さん

8.3.2 ディレクトリを移動: cd

では、ディレクトリの冒険の旅に出かけよう。ディレクトリを移動するには、 cd を使う。これは「**c**hange **d**irectory – チェンジ・ディレクトリ」の略だ。使い方は以下の通り。この「パス」というのは、絶対パス、相対パスのどちらでもいい。

cd 「パス」

1 つ上へ移動

では、 1 つ上のディレクトリに移動してみよう。

現在は「/home/ei00/ei00000」にいる。よって、1 つ上のディレクトリは、絶対パスでは「/home/ei00」となり、相対パスでは「..」となる。よって 1 つ上に移動するには、「cd /home/ei01」、または「cd ..」のどちらでもいい。

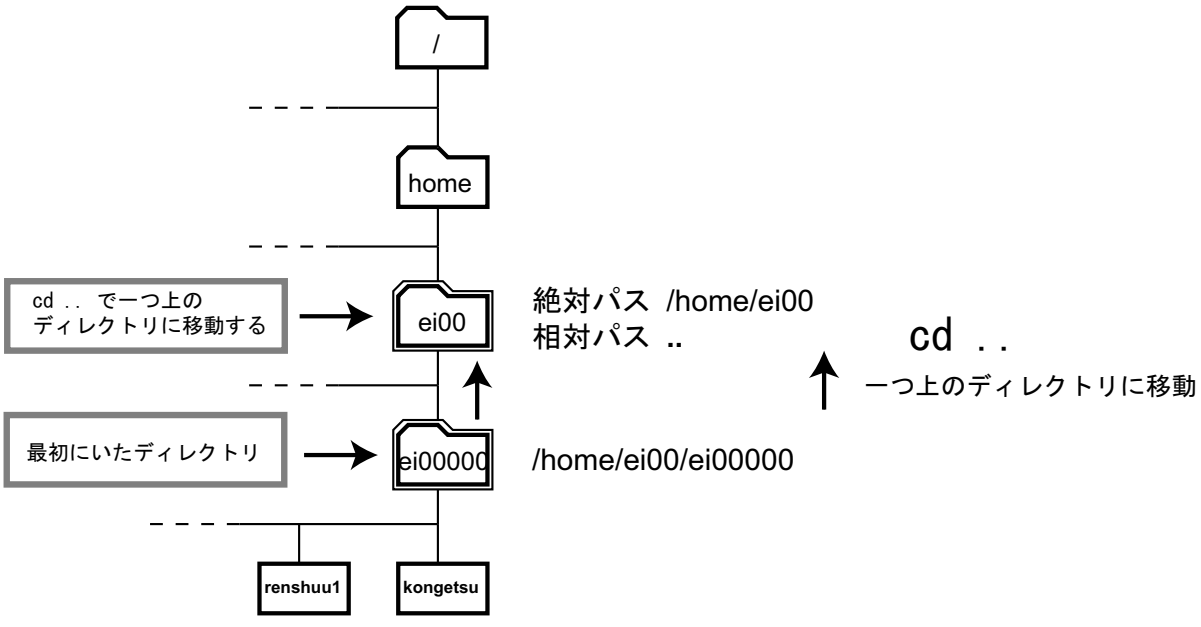
この場合、明らかに「cd ..」の方が楽なので、こちらを実行する。

```
% cd ..
%
```

これで、ディレクトリを一つ登れた。pwd で確認してみよう。

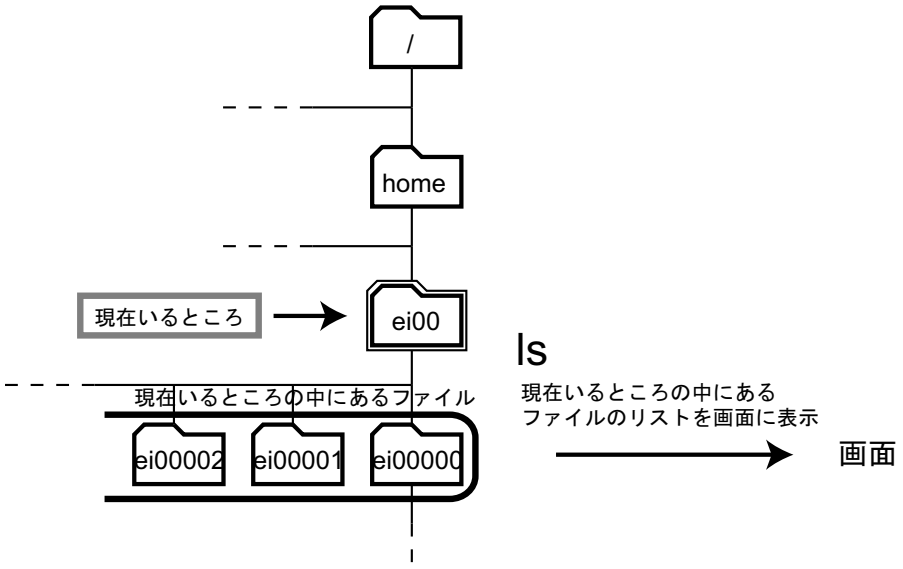
```
% pwd
/home/ei00
%
```

確かに、「/home/ei00」にいることがわかる。



では、ここにどんなファイルがあるか ls で見てみよう。すると、同級生のホーム・ディレクトリのリストが見える。

```
% ls
ei00001 ei00006 ei00011 ei00016 ei00021 ei00026 ei00031 ei00036 ei00041
ei00002 ei00007 ei00012 ei00017 ei00022 ei00027 ei00032 ei00037 ei00042
ei00003 ei00008 ei00013 ei00018 ei00023 ei00028 ei00033 ei00038 ei00043
ei00004 ei00009 ei00014 ei00019 ei00024 ei00029 ei00034 ei00039 ei00044
ei00005 ei00010 ei00015 ei00020 ei00025 ei00030 ei00035 ei00040 ei00045
%
```



ファイルの種類を識別: ls の -F オプション

では、ここで「ls -F」と打ってみよう。

「-F」は「file – ファイル (の種類)」の意味で、次のようにディレクトリの場合、後ろに「/」を付けて表示してくれるので、ずいぶん見やすくなる。

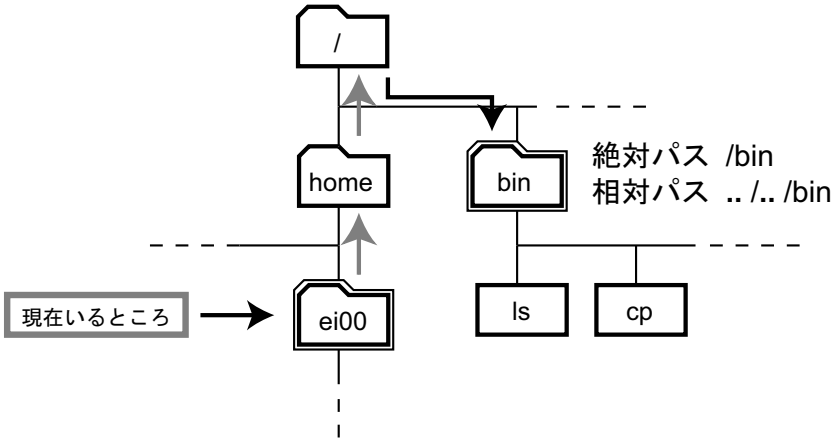
```
% ls -F
ei00001/        ei00010/        ei00019/        ei00028/        ei00037/
ei00002/        ei00011/        ei00020/        ei00029/        ei00038/
ei00003/        ei00012/        ei00021/        ei00030/        ei00039/
ei00004/        ei00013/        ei00022/        ei00031/        ei00040/
ei00005/        ei00014/        ei00023/        ei00032/        ei00041/
ei00006/        ei00015/        ei00024/        ei00033/        ei00042/
ei00007/        ei00016/        ei00025/        ei00034/        ei00043/
ei00008/        ei00017/        ei00026/        ei00035/        ei00044/
ei00009/        ei00018/        ei00027/        ei00036/        ei00045/
%
```

「ls -F」の「-F」のように、「-なんとか」のような形の引数を**「コマンドのオプション」**という。

コマンドには、それぞれのコマンドによって、いろいろなオプションがある。そして、オプションを付けると、コマンドの働きを変えることができるようになっている。他にどんなオプションがあるかは ls のオンライン・マニュアルに載っている。

一気に絶対パスで移動

次に、 UNIX の最も基本的なコマンドの置いてある「/bin」に移動してみよう。



これは絶対パスではそのまま「/bin」である。一方、相対パスで考えると、現在「/home/ei00」にいるので、「一つ上」の「一つ上」の中にある「bin」ということで、「../../bin」になる。

つまり、「cd /bin」か「cd ../../bin」だ。今回は絶対パスの方が簡単なので、こちらを使おう。

```
% cd /bin
%
```

これで、一気に「/bin」まで移動できたはずだ。pwd で確認してみよう。

```
% pwd
/bin
%
```

確かに「/bin」にいることがわかる。

では、どんなファイルがあるか ls で見てみよう。

```
% ls
[               dd              kill            pwd             sleep
cat             df              ln              rcp             stty
chio            domainname      ls              red             sync
chmod           echo            mkdir           rm              test
cp              ed              mv              rmail
csh             expr            pax             rmdir
date            hostname        ps              sh
%
```

ここで ls -F すると、次のようになる。「*」マークは実行できるコマンドの目印である。

```
% ls -F
[*              dd*             kill*           pwd*            sleep*
cat*            df*             ln*             rcp*            stty*
chio*           domainname*     ls*             red*            sync*
chmod*          echo*           mkdir*          rm*             test*
cp*             ed*             mv*             rmail*
csh*            expr*           pax*            rmdir*
date*           hostname*       ps*             sh*
%
```

他のコマンドでも同様なのだが cd には、ここまで見てきたように、「相対パス」と「絶対パス」の両方が使える。

では、どっち使えばいいのかというと、「自分のいるとこの近くなら、相対パス」、「遠くなら絶対パス」が便利なことが多い。

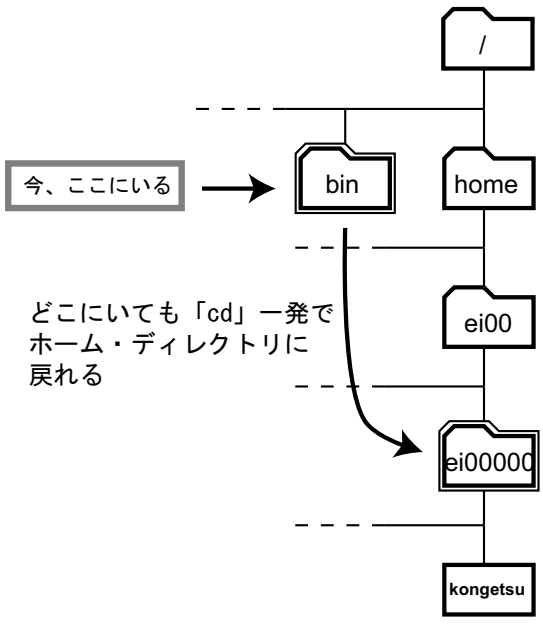
ホームに戻る

では、ホーム・ディレクトリに戻ろう。

これはとても簡単で、単に「cd」と打てばいい。いつ、どこにいても、自分のホーム・ディレクトリには、「cd」と打つだけで戻れる。

では、やってみよう。

```
% cd
%
```



pwd で確認してみよう。

```
% pwd
/home/ei00/ei00000
%
```

確かに、ホーム・ディレクトリに戻れたことがわかる。

ホームを表す記号: 「~」

UNIX システムでは、自分のホーム・ディレクトリを表すのに「~」を使う。さきほどの例では、「cd」と単に打ったが、「cd ~」でも全く同様の効果がある。では、ためしに使ってみよう。

まず、どこでもいいのだが、今回は簡単のためルート・ディレクトリ「/」に移動しよう。

```
% cd /
%
```

pwd で確認してみよう。

```
% pwd
/
%
```

確かにルート・ディレクトリ「/」にいることがわかる。

ここから、「cd ~」で自分のホーム・ディレクトリに戻ってみよう。

```
% cd ~
%
```

pwd で確認してみよう。

```
% pwd
/home/ei00/ei00000
%
```

確かにホーム・ディレクトリに戻れた。

「~」は自分のものだけでなく、他人のホーム・ディレクトリも「~その人の ID」としてやると使うことができる。たとえば、「kokubo」さんのホーム・ディレクトリは「~kokubo」、「ei00000」さんのものは「~ei00000」となる。

「~」はホーム・ディレクトリを現す

「~」　　→ 自分のホーム・ディレクトリ

「~kokubo」 → kokubo のホーム・ディレクトリ

「~ei00000」→ ei00000 のホーム・ディレクトリ

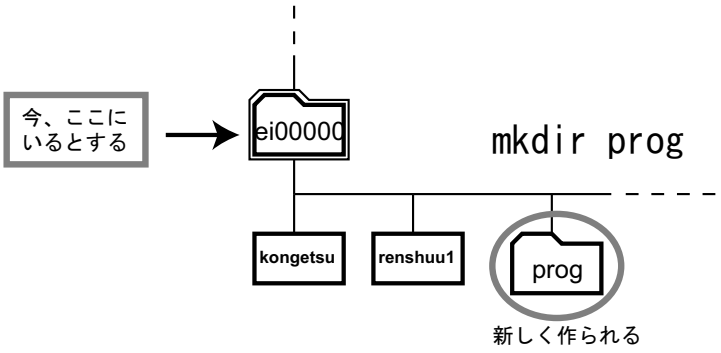
8.3.3 ディレクトリを作る: mkdir

ディレクトリを作るには mkdir だ。これは「**m**ake **dir**ectory – メイク・ディレクトリ」の短縮形である。使い方は次の通り。

mkdir 「作りたいディレクトリの名前」

では、自分のホームの下に「prog」というディレクトリを作ってみよう。

```
% mkdir prog
%
```



では、 ls で確認してみよう。

```
% ls
2000nen         3years          kotoshi         renshuu1.ps     thismonth
2001nen         4gatsu          prog            renshuu2
2002nen         kongetsu        renshuu1        tanjoubi
%
```

これでは、「prog」があるのはわかるが、ファイルかディレクトリか区別がつかない。

区別をはっきりつけるには ls -F と打つ。

```
% ls -F
2000nen         3years          kotoshi         renshuu1.ps     thismonth
2001nen         4gatsu          prog/           renshuu2
2002nen         kongetsu        renshuu1        tanjoubi
%
```

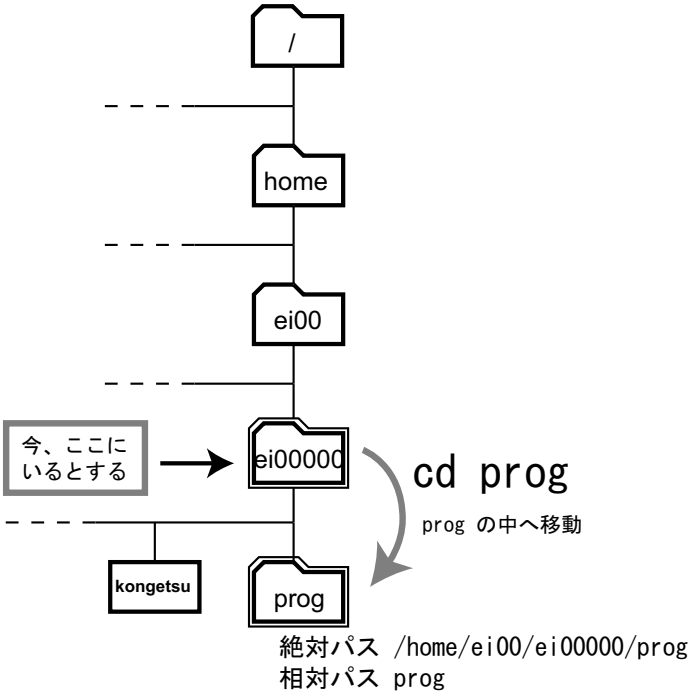
たしかに、「prog」というディレクトリがあるのがわかる。

では、「prog」の中に入ってみよう。これは、絶対パスなら「/home/ei00/ei00000/prog」で、相対パスでは単に「prog」なので、相対パスの方が簡単だ。よって、「cd prog」と打とう。

```
% cd prog
%
```

では、 pwd で確認してみよう。

```
% pwd
/home/ei00/ei00000/prog
%
```



8.3.4 ディレクトリを消す: rmdir

ディレクトリを消すには、 rmdir だ。これは「**rem**ove **dir**ectory – ディレクトリ削除」の略である。使い方は次の通りだ。

rmdir 「消したいディレクトリの名前」

では、実際にやってみよう。まず、自分のホーム・ディレクトリにもどろう。ホームに戻るには単に cd でいい。

```
% cd
%
```

次のように pwd で確認すると、確かにホームにいることがわかる。

```
% pwd
/home/ei00/ei00000
%
```

では、 ls -F でファイルのリストを確認しよう。

```
% ls -F
2000nen         3years          kotoshi         renshuu1.ps     thismonth
2001nen         4gatsu          prog/           renshuu2
2002nen         kongetsu        renshuu1        tanjoubi
%
```

ここにある「prog」というディレクトリを次のようにして消してみる。

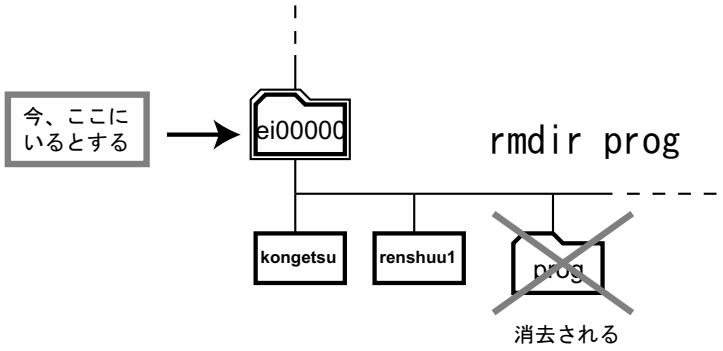
```
% rmdir prog
%
```

これで、消えたはずだ。ls -F で確認してみよう。

```
% ls -F
2000nen         3years          kotoshi         renshuu2
2001nen         4gatsu          renshuu1        tanjoubi
2002nen         kongetsu        renshuu1.ps     thismonth
%
```

確かに消えている。

注・ディレクトリの中に何かファイルが残っていると、 rmdir でディレクトリを消すことはできない。



8.3.5 ディレクトリ間でのファイルの移動、コピー

mv や cp で別のディレクトリにファイルを移動したり、コピーする方法を紹介しよう。

まず、その前に準備をしよう。「prog」というディレクトリがあるかどうかを確認する。

```
% ls -F
2000nen         3years          kotoshi         renshuu2
2001nen         4gatsu          renshuu1        tanjoubi
2002nen         kongetsu        renshuu1.ps     thismonth
%
```

もしも、上のように「prog」がないなら、mkdir で作る。

```
% mkdir prog
%
```

では、 ls -F でファイルのリストを確認しよう。

```
% ls -F
2000nen         3years          kotoshi         renshuu1.ps     thismonth
2001nen         4gatsu          prog/           renshuu2
2002nen         kongetsu        renshuu1        tanjoubi
%
```

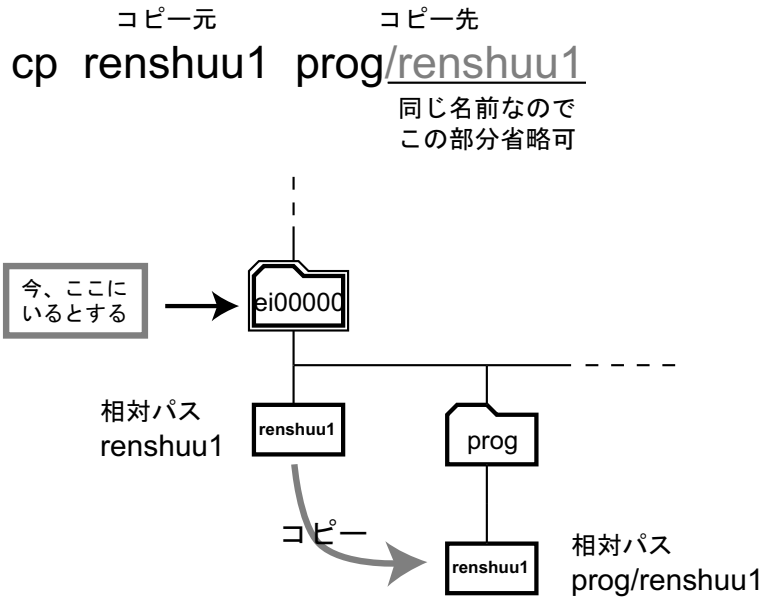
確かに「prog」ができていることがわかる。これで準備は OK である。

「prog」というディレクトリに「renshuu1」というファイルをコピーしたいとする。

ファイルをディレクトリの中に移動やコピーする場合でも、mv や cp の使い方は、普通に移動したりコピーするときと基本的には同じである。ファイル名の部分にパスを書いてあげればいい。

つまり、「renshuu1」を「prog」の下に同じ名前でコピーするには、「cp renshuu1 prog/renshuu1」となる。

また、今回のようにコピー先の名前がコピー元と同じ場合には省略可能で、「cp renshuu1 prog/」や「cp renshuu1 prog」としても OK である。



では、実際にやってみよう。

```
% cp renshuu1 prog
%
```

うまくコピーできたか確かめてみよう。「prog」の中に移動し、ls -F で確かめてみると、たしかに「renshuu1」ができている

```
% cd prog
% ls -F
renshuu1
%
```

それでは、「prog」の中の「renshuu1」を、一つ上のディレクトリ (この場合はホーム・ディレクトリ) に「renshuu3」という名前で移動してみよう。

現在、「prog」の中にいるので、移動元が「renshuu1」で、移動先は相対パスで「../renshuu3」となる。よって、「mv renshuu1 ../renshuu3」である。

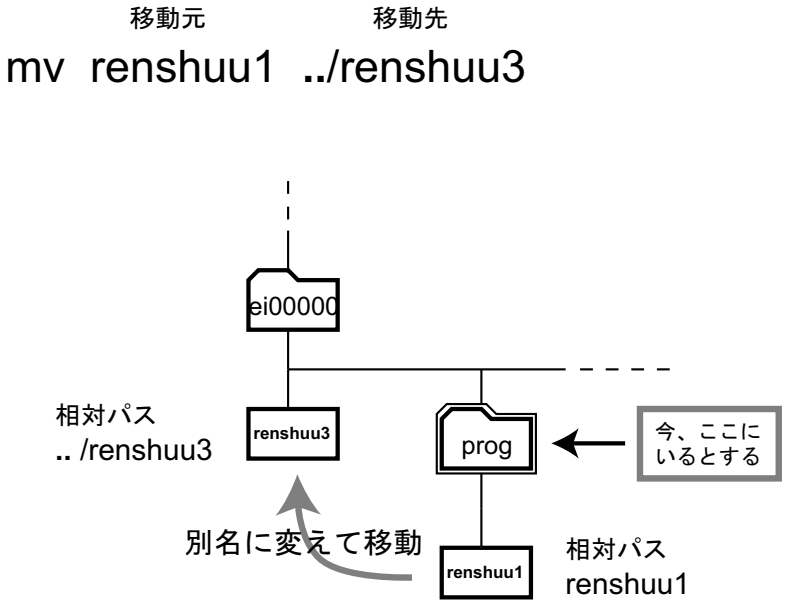
```
% mv renshuu1 ../renshuu3
%
```

まず、「prog」の中でファイルのリストを表示してみると、「renshuu1」がなくなっていることがわかる。

```
% ls
%
```

次に、ホーム・ディレクトリにもどって確認してみると、確かに「renshuu3」ができていることがわかる。

```
% cd ..
% ls -F
2000nen         3years          kotoshi         renshuu1.ps     tanjoubi
2001nen         4gatsu          prog/           renshuu2        thismonth
2002nen         kongetsu        renshuu1        renshuu3
%
```



8.3.6　移動という言葉

UNIX コマンドで mv と言えば「ファイルを**移動**」、cd と言えば「ディレクトリを**移動**」で、どちらも日本語では同じ「**移動**」である。

この 2 つの違いは、とても簡単で、「mv は**ファイルなどを**移動する」、「cd は**自分が**移動する」ということである。

mv では、自分は動かないで、動くのはファイルである。一方、 cd では、自分が移動して、ファイルなどは動かないのだ。

mv A B → **ファイル A を** B に移動（または名前の変更）

cd C → **自分が** C に移動

この章で紹介したコマンド

ディレクトリ操作コマンド

pwd : 自分が現在いる場所を表示

　　使い方: pwd

cd : ディレクトリを移動する

　　使い方: cd 「パス名」

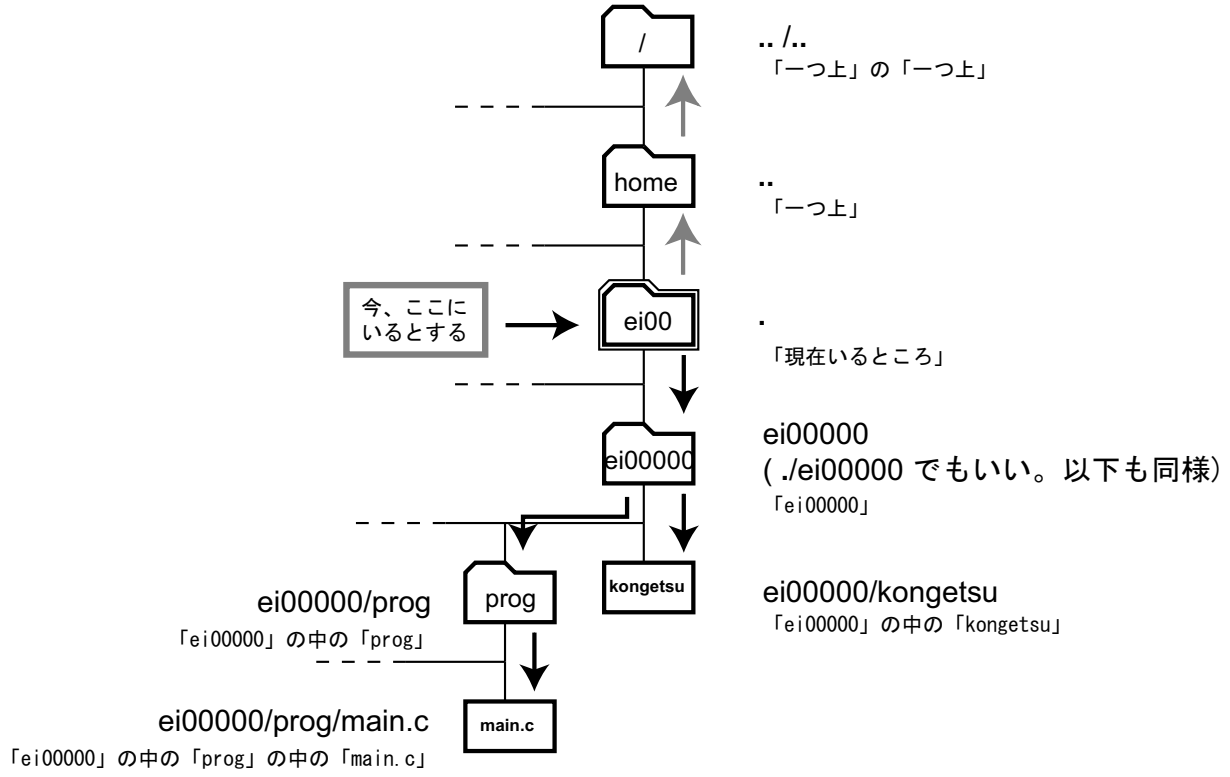
mkdir : ディレクトリの作成

　　使い方: mkdir 「ディレクトリ名」

rmdir : ディレクトリの削除

　　使い方: rmdir 「ディレクトリ名」

問題の解答


/
.. /..
「一つ上」の「一つ上」
home
..
「一つ上」
今、ここに
いるとする
ei00
.
「現在いるところ」
ei00000
ei00000
(./ei00000 でもいい。以下も同様)
「ei00000」
prog
kongetsu
ei00000/prog
「ei00000」の中の「prog」
ei00000/kongetsu
「ei00000」の中の「kongetsu」
main.c
ei00000/prog/main.c
「ei00000」の中の「prog」の中の「main.c」

第9章　プロセスとジョブ

プログラムを同時に実行

最近のコンピュータは、パソコンのような小型のものでも、ブラウザ、ゲーム、エディタなどを同時に起動して使うことができる。これは同時にいくつものプログラムを実行できるということである。

このような機能をマルチタスク (multitask: 複数の仕事) という。小型のコンピュータで、こんなことができるようになったのは、本当に最近のことである。実際、初期のコンピュータは、大型のものでも、一度に一つのプログラムしか実行できなかった。

1970 年頃、 UNIX が開発された頃に、タイム・シェアリング[1]などによって、マルチタスクを実現することができるようになり、ウィンドウをいくつも開いたり、何人もの人で同じコンピュータを同時に使えるようになった。

走っているプログラムのことをプロセスと呼ぶが、マルチタスクのシステムでは同時にプロセスがいくつも走っている。今回は、このプロセスの見方、コントロールの仕方を紹介しよう。プロセスがコントロールできるようになると、システムをスマートに使えるようになるし、暴走したプログラムを止めたりすることもできるようになる。

9.1　プロセス

9.1.1　プロセスの表示: ps

UNIX システムでは、 Kterm をいくつも開いたり、時計 (たとえば xclock) を表示しながら、Mule を使ったりといった具合に、同時にいくつものプログラムを実行できる。

これは、システムが Kterm、xclock、Mule などのプログラムを同時に実行しているということだ。実行中のコマンドのことを**「プロセス」**という。

では、今、どんなプロセスが走っているのかを見てみよう。走っているプロセスは ps (**P**rocess **S**tatus: プロセスの状態) コマンドで見ることができる。

```
% ps
  PID  TT  STAT      TIME COMMAND
  273  p0  Ss     0:00.07 -csh (tcsh)
  288  p0  R+     0:00.00 ps
%
```

[1]Time Sharing: 「時分割」とか訳す。とても短い時間ごとにプログラムを切替えて、次々に少しずつ実行するという方法。

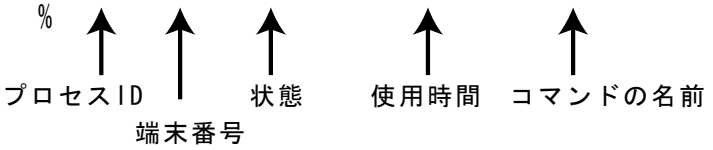
なお、これは 実は全部ではない[2]。

自分のプロセスを全部表示

ある人のプロセスを全部表示するには「ps -U 「その人の ID」」と入力すればいい。

では、実際にやってみよう。たとえば、 ei00000 さんが、自分のプロセスを全部表示するには、「ps -U ei00000」である。

```
% ps -U ei00000
  PID  TT  STAT      TIME COMMAND
  267  ??  Is    0:00.02 /bin/sh /usr/X11R6/lib/X11/xdm/Xsession
  272  ??  S     0:00.08 twm
  273  p0  Ss    0:00.07 -csh (tcsh)
  289  p0  R+    0:00.00 ps -U ei00000
%
```



これが、走らせている全部のプロセスだ。

この表示の見方を説明しよう。

まず、プロセスには、システムがコントロールするために付けた番号がある。「PID(プロセス ID)」というのがそれである。システムの中には、同時に同じ番号のプロセスは存在しないようになっている。

次の列の「TT」は、端末番号という。実は、 Kterm のように、コマンドを打つことができるウィンドウを開くと、システムがウィンドウごとに番号を割り振る。それが端末番号で、これを見ると、どのウィンドウから実行したコマンドかがわかる[3]。

次の「TIME」は、これまでにプロセスが走ったトータルの時間。

最後の「COMMAND」は、コマンドの名前である。

☆練習☆

1. まず、 X Window System で、いくつかのアプリケーションを起動してから、ps を実行し、どんなプロセスが走っているのかを見てみよう。

[2]表示されているのは、現在の端末から実行されているプロセスだけ。少し難しい話なので、ここの部分はわからなくてもよい。

[3]端末番号が「??」になっているのは、 Kterm などからは起動していないコマンドである。これらは、単に「ps」と打つと表示されない。

すべてのプロセス

login した本人が動かしている以外にも、システムそのものが動かしているプロセスがたくさんある。

マシンの中で動いているプロセスを全部表示するには、「ps -aux」と入力する。これはたくさん出るので、次のようにページャー jless にパイプしてやるとよい。

```
% ps -aux | jless
```

すると、次のように表示される。これが、現在実行中の全プロセスだ。なお、 USER が「root」や「daemon」というのは、システムが走らせているプロセスである。

```
USER       PID %CPU %MEM    VSZ  RSS  TT  STAT STARTED       TIME COMMAND
ei00000    297  0.0  1.5    680  948  p0  RV    1Jan70    0:00.00 -csh (tcsh)
root         1  0.0  0.4    408  244  ??  Is    3:23AM    0:00.01 /sbin/init --
root         2  0.0  0.0      0   12  ??  DL    3:23AM    0:00.00  (pagedaemon)
root         3  0.0  0.0      0   12  ??  DL    3:23AM    0:00.00  (vmdaemon)
root         4  0.0  0.0      0   12  ??  DL    3:23AM    0:00.02  (update)
root        27  0.0  0.1    204   84  ??  Is    3:23AM    0:00.00 adjkerntz -i
root        90  0.0  0.8    208  488  ??  Ss    6:23PM    0:00.05 syslogd
daemon     100  0.0  0.9    176  576  ??  Is    6:23PM    0:00.00 portmap
root       104  0.0  0.9    180  544  ??  Ss    6:23PM    0:00.00 ypbind
root       114  0.0  0.2    212   88  ??  I     6:23PM    0:00.00 nfsiod -n 4
root       115  0.0  0.2    212   88  ??  I     6:23PM    0:00.00 nfsiod -n 4
root       116  0.0  0.2    212   88  ??  I     6:23PM    0:00.00 nfsiod -n 4
root       117  0.0  0.2    212   88  ??  I     6:23PM    0:00.00 nfsiod -n 4
root       132  0.0  1.0    240  604  ??  Is    6:23PM    0:00.02 inetd
root       135  0.0  0.8    364  516  ??  Is    6:23PM    0:00.01 cron
root       138  0.0  0.9    208  544  ??  Is    6:23PM    0:00.00 lpd
root       141  0.0  1.3    616  820  ??  Ss    6:23PM    0:00.00 sendmail: acce
root       164  0.0  0.7    168  420  ??  Ss    6:23PM    0:00.13 moused -p /dev
bin        224  0.0  0.9    540  584  ??  I     6:23PM    0:00.01 /usr/local/sbi
root       236  0.0  1.7    268 1064  ??  Is    6:23PM    0:00.01 /usr/X11R6/bin
root       244  0.8 13.7   3836 8640  ??  Ss    6:23PM    0:02.24 /usr/X11R6/bin
root       249  0.0  0.9    180  548  v0  Is+   6:23PM    0:00.01 /usr/libexec/g
:
```

このように ps は、次のように使う。

```
ps
ps -U 「ユーザの ID」
ps -aux
```

9.1.2 CPU をよく使っているプロセス: top

ps 以外にも、プロセスを見る方法はある。top を使うと、 CPU を使っている順に、上位から順番にプロセスを表示してくれる。単に「top」と打って見よう。すると、下のようになる。なお、 top を終了してこの状態から抜けるには「q」と打つ。

```
last pid:   294;  load averages:  0.04,  0.02,  0.00                    18:25:17
30 processes:  1 running, 29 sleeping
CPU states:  1.2% user,  0.0% nice,  0.0% system,  1.9% interrupt, 96.9% idle
Mem: 11M Active, 4956K Inact, 7324K Wired, 3790K Buf, 38M Free
Swap: 64M Total, 64K Used, 64M Free

  PID USERNAME PRI NICE SIZE    RES STATE    TIME   WCPU    CPU COMMAND
  244 root      2   0  3836K  8640K select   0:02  1.31%  1.30% XF86_SVGA
  294 ei00000  28   0   652K   860K RUN      0:00  0.40%  0.04% top
  279 ei00000  18   0   680K   984K pause    0:00  0.00%  0.00% tcsh
  273 ei00000  18   0   680K   940K pause    0:00  0.00%  0.00% tcsh
  135 root     18   0   364K   516K pause    0:00  0.00%  0.00% cron
   27 root     18   0   204K    84K pause    0:00  0.00%  0.00% adjkerntz
  252 root     10   0   352K  1600K wait     0:00  0.00%  0.00% xdm
  267 ei00000  10   0   492K   316K wait     0:00  0.00%  0.00% sh
    1 root     10   0   408K   244K wait     0:00  0.00%  0.00% init
  117 root     10   0   212K    88K nfsidl   0:00  0.00%  0.00% nfsiod
  114 root     10   0   212K    88K nfsidl   0:00  0.00%  0.00% nfsiod
  115 root     10   0   212K    88K nfsidl   0:00  0.00%  0.00% nfsiod
  116 root     10   0   212K    88K nfsidl   0:00  0.00%  0.00% nfsiod
  293 ei00000   3   0   600K   892K ttyin    0:00  0.00%  0.00% vi
  251 root      3   0   180K   548K ttyin    0:00  0.00%  0.00% getty
  250 root      3   0   180K   548K ttyin    0:00  0.00%  0.00% getty
  249 root      3   0   180K   548K ttyin    0:00  0.00%  0.00% getty
```

9.1.3 プロセスを止める: kill

コマンドの実行を止める方法を、ここまでにいくつか紹介してきた。例えば Ctrl+c と入力する方法、また X Window System なら Twm のメニューから「Delete」を選択する方法などである。

ここではより細かい操作ができる方法を紹介しよう。

コマンドの実行を止めるのには、 kill を使ってプロセスを殺せばいい。使い方は以下の通りである。

```
kill 「PID」
```

では、実際にやってみよう。まず、あらかじめ「xclock &」と打って、 xclock を起動しておく。

```
% xclock &
[1] 301
%
```

次に、 kill を使うには、 PID を知る必要があるので、次のように ps で xclock の PID を調べる。

```
% ps -U ei00000
  PID  TT  STAT      TIME COMMAND
  267  ??  Is     0:00.02 /bin/sh /usr/X11R6/lib/X11/xdm/Xsession
  272  ??  S      0:00.08 twm
  273  p0  Ss     0:00.07 -csh (tcsh)
  301  p0  S      0:00.03 xclock
  303  p0  R+     0:00.00 ps -U ei00000
%
```

すると、 xclock の PID は 301 だとわかる。そこで「kill 301」する。

```
% kill 301
%
```

すると、 xclock が消えたはずだ。なお kill を実行すると、「[1] Done xclock」のようなメッセージが出ることがある。

では、念ため ps で確認してみよう。

```
% ps -U ei00000
  PID  TT  STAT      TIME COMMAND
  267  ??  Is     0:00.02 /bin/sh /usr/X11R6/lib/X11/xdm/Xsession
  272  ??  S      0:00.08 twm
  273  p0  Ss     0:00.07 -csh (tcsh)
  306  p0  R+     0:00.00 ps -U ei00000
```

確かに PID が 301 の xclock のプロセスが消えている。

ちなみに、このような方法でもプロセスが殺せない場合は、強制的に殺す方法もある。強制的に殺すには、「kill -KILL 「PID」」と、「-KILL」オプションを付けてやるといい。

☆演習☆

1. 実際に kterm や xclock などを起動しておいて、kill してみよう。

9.2 ジョブ

9.2.1 昔話: フォアグランド・ジョブとバックグランド・ジョブとは何か

まず、イメージをつかみやすくするために、昔のコンピュータの絵を見て欲しい。次の図のように一台の大型のコンピュータがあり、これに何台もの端末と呼ばれるものがつながっていた[4]。

一方、今では X Window System で、一画面の中に何枚でも気軽に Kterm のウィンドウを開くことができる。Kterm のウィンドウ一個一個は、昔の話では、なんとそれぞれの端末に相当する[5]のである。

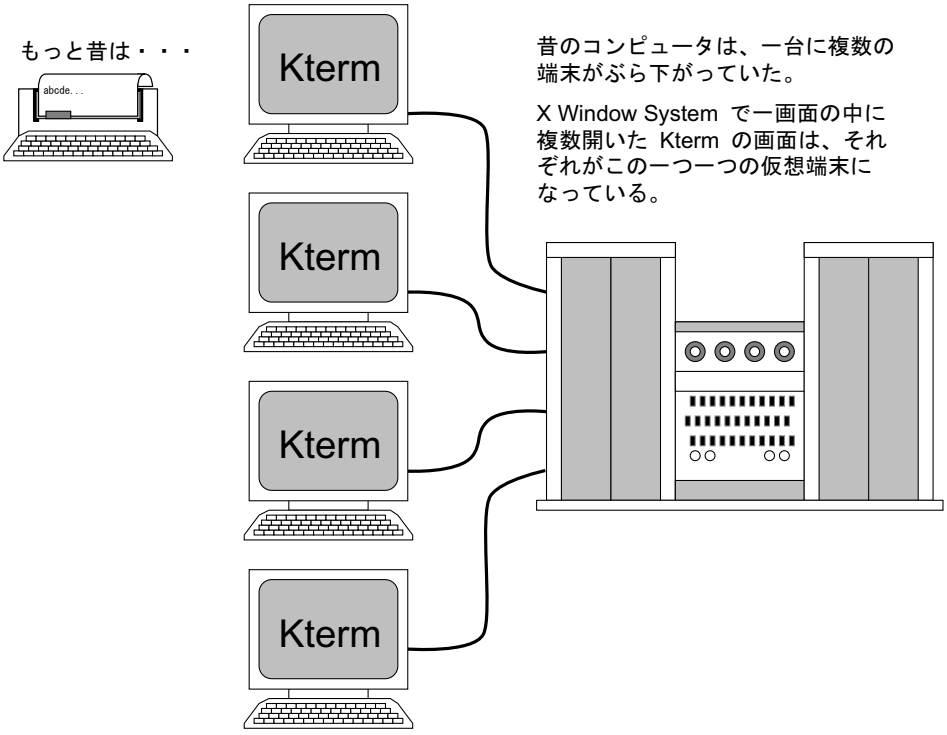


図 9.1: 昔のコンピュータのイメージ

しかも、このそれぞれの端末は、文字しか表示できなくて、画面の中に新しくウィンドウを開いたりすることもできなかった。そして一度コマンドを打ったら、そのコマンドが実行し終わるまで次のコマンドを入力することはできなかったのだ。

たとえば、大型コンピュータなどでも、計算[6]をさせると何日もかかることが昔は (ものによっては現在でも) よくあった。ある端末で時間のかかる計算をやらせるプログラムを書いて実行させる。すると、その端末は計算が終わるまで使えない … 。それではいくらなんでも不便だということで、フォアグランド・ジョブ (foreground job: 画面の上でやる仕事) とバックグランド・ジョブ (background job:

[4]図では主に「画面」と「キーボード」のついた端末を描いている。しかし、実は最初の頃は左側に描いてるようにテレタイプライタといって、「画面」ですらなくて、代わりに「タイプライタの印字部」がついているようなものが使われていた。そしてもっと前は … 。

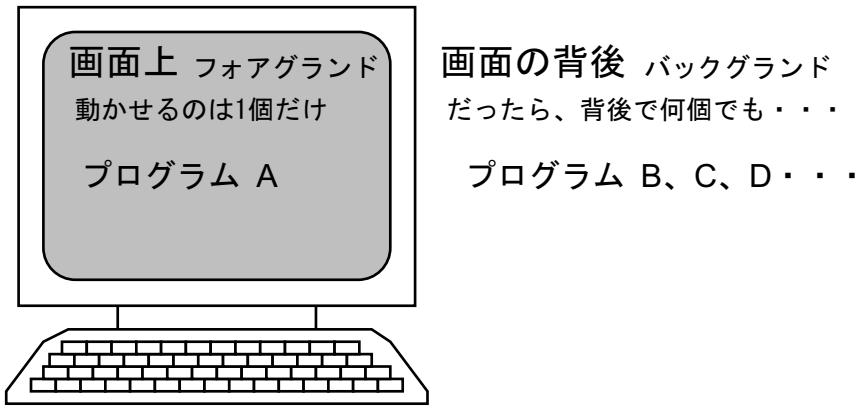
[5]そういうわけで Kterm のようなソフトをターミナル・エミュレータ、あるいは仮想端末などと呼ぶことがある。

[6]有名な話では流体計算、つまり大気、水、油などの流れる様子を計算する問題などがある。天気予報や流体の研究などをするには、よくスーパーコンピュータが使われていて、それでもすごい時間がかかった。

画面のバックでやる仕事) というものが考えられた。

つまり、それぞれのユーザにとって、画面は一枚しかないので、画面を使った仕事 (フォアグランド・ジョブ) は一つしかできない。しかしである、たとえば、計算だけしかプログラムなんて、計算が終わって結果が出るまでは、画面に何も出さなくていいはずじゃないかと。そういう画面を使う必要のない計算は画面のバックでやってもらって、画面は別な仕事[7]に使えばいいじゃないかと。そういう考え方である。

こうして、画面の表でやる仕事 (フォアグランド・ジョブ) の他に、画面の背後でも仕事 (バックグランド・ジョブ) ができるように OS が設計されるようになった。



9.2.2 UNIX システムでのジョブのコントロール

UNIX システムでは、ジョブというのは、だいたいプロセスのようなものと思ってもらえばいい[8]。先ほどの説明通り、ジョブには、フォアグランド (foreground: 表で実行) と、バックグランド (background: 裏で実行) がある。そしてフォアグランドで走らせられるジョブは一つだけ、バックグラウンドでは複数のジョブを走らせることができる。

フォアグランド・ジョブの例

UNIX システムでは、普通にコマンドを打つと、それはフォアグランドのジョブになる。これは、表で動いているジョブで、Kterm のような仮想端末において、それぞれ同時に一つずつしか走らせられない。つまり、普通に「`ls`」とか打った場合、`ls` が終わるまで、次のコマンドは実行できないのだ。とは言え、ほとんどの場合、一瞬で実行が終わるので気がつかないことだろう。

では、敢えて時間がかかるようにして実行してみよう。次のように「`(sleep 30; ls)`」と入力してみよう。「`sleep 30`」というのは「30 秒休止」させるコマンドで、このようにしてやると「30 秒休止してから `ls` を実行」する。

---

[7] メールを読み書きしたり、新しい計算のプログラムを書いたり、ゲームをしたり … 。

[8] 厳密にはジョブとプロセスは異なっている。例えば、単に「`ls`」と打った場合、プロセスもジョブも `ls` の 1 つだ。しかし、「`ls | jless`」と打った場合、プロセスは `ls` と `jless` の 2 つ、一方ジョブは `ls | jless` が 1 セットで 1 つになる。ジョブというのは、このようにユーザが打ったコマンドの 1 まとまりのことである。

```
% (sleep 30; ls)
(30 秒間ストップ)
%
```

30 秒間は、この画面にどんなコマンドを打っても実行できないはずである。この間に打ったコマンドは、30 秒たって `ls` が実行されてから、順番に実行される。

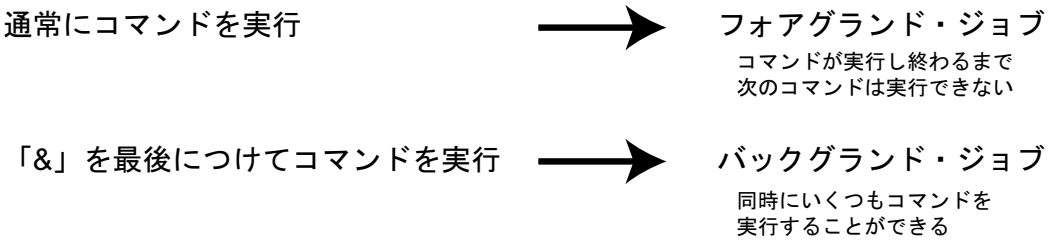
バックグランド・ジョブの例

バックグランド・ジョブは、一つの仮想端末でいくつも同時に走らせることができる。UNIX システムでコマンドをバックグランドで走らせるには、単にコマンドの最後に「`&`」を付けるだけでいい。

では、先ほどと同様のコマンドを今度はバックグランドで実行してみよう。

```
% (sleep 30; ls) &
[1] 62749
%
```

30 秒後に `ls` が実行されるのは変わらない。しかし、今回はその間にも他のコマンドが実行できるので、試してみるといいだろう。また、バックグランド・ジョブは同時にいくつも実行できるので、時間差でいろいろなコマンドを実行してみるとおもしろいだろう。



フォアグランドをバックグランドに切り替え

UNIX システムでは、フォアグランド・ジョブとバックグランド・ジョブを切り替えることができる。また、フォアグランド・ジョブの実行を「一時的に停止 (サスペンド)」することもできる。

具体的にやってみよう。まず非常に時間のかかる (180 秒 = 3 分) フォアグランド・ジョブを実行してみよう。

```
% (sleep 180; ls)
```

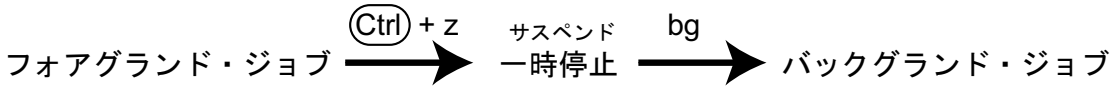
フォアグランド・ジョブをバックグランド・ジョブに切り替えるには、まず一時的にこれを停止 (サスペンド) させる。それには「Ctrl+z」と入力する。すると、次のように一時的にジョブが停止し、画面に「`^Z`」と表示される。

```
% (sleep 180; ls)
^Z
中断
%
```

次に「bg」と打つと、一時停止中のジョブがバックグランドになる。

```
% (sleep 180; ls)
^Z
中断
% bg
[1]    ( sleep 180; ls ) &
%
```

このようにフォアグランド・ジョブは、「Ctrl+z」で一時停止 (サスペンド) し、「bg」でバックグランド・ジョブに切り替えることができる。



バックグランドをフォアグランドに切り替え

次にバッググランド・ジョブを、フォアグランド・ジョブに切り替える方法を紹介しよう。

さきほどと同様に非常に時間のかかる (180 秒 = 3 分) 別なジョブを、今度はバックグランドで実行してみよう。

```
% (sleep 180; cal) &
[2] 62837
%
```

ここで、ジョブの一覧を見てみよう。それには、以下のように jobs コマンドを使う。

```
% jobs
[1]    実行中です                    ( sleep 180; ls -CF )
[2]  + 実行中です                    ( sleep 180; cal )
```

jobs コマンドを使うと、現在実行中のバックグランド・ジョブの一覧を見ることができる。先頭に書かれた「[1]」や「[2]」は「**ジョブ番号**」と呼ばれる。

これらのバックグランド・ジョブをフォアグランド・ジョブに切り替えるには「fg %「ジョブ番号」」と入力する。

では、次のようにしてジョブ番号が「1」の方をフォアグランドに切り替えてみよう。

```
% fg %1
( sleep 180; ls -CF )
```

このジョブがフォアグランドになって、実行が終わるまで他のコマンドは実行できなくなったはずだ。

jobs ⟶ ジョブの一覧の表示

fg　%「ジョブ番号」⟶ 指定したジョブをフォアグランドに

X Window System アプリケーションの場合

ここで思い出して欲しいのは、X Window System で新しくウィンドウを開くときのことである。新しくウィンドウを開くコマンドを入力するときには、最後に「&」を付けていたが、これは実はバックグランドでコマンドを実行していたのだ。

「&」を付けないと、フォアグランドでコマンドが実行されるから、新しく開いたウィンドウを終了するまでは、元のウィンドウでは何もできないのである。

これが昔のコンピュータの話と違うのは、新しくウィンドウを開くコマンド[9]を実行することは、新しい端末を用意することに相当するということである。

この章で紹介したコマンド

プロセスとジョブ

`ps`：プロセスを表示
　　使い方: `ps -U` 「ユーザ ID」、`ps -aux`

`top`：CPU を使っている上位のプロセスを表示
　　使い方: `top`

`kill`：プロセスを殺す
　　使い方: `kill` 「PID」

`jobs`：ジョブの一覧を表示

Ctrl+z：フォアグランド・ジョブを一時停止

`bg`：一時停止中のジョブをバックグランドに

`fg`：バックグランド・ジョブをフォアグランド・ジョブに
　　使い方: `fg` 「ジョブ番号」

[9]X Window System アプリケーションと呼ばれる。

第10章　パーミッション

セキュリティとファイルの共有

「パーミッション – permission」とは、「許可」という意味だ。

UNIX システムでは、ファイルに 3 種類のパーミッションがある。「ファイルを読む許可」「ファイルを書いたり消したりする許可」「ファイルを実行する許可」の 3 つだ。

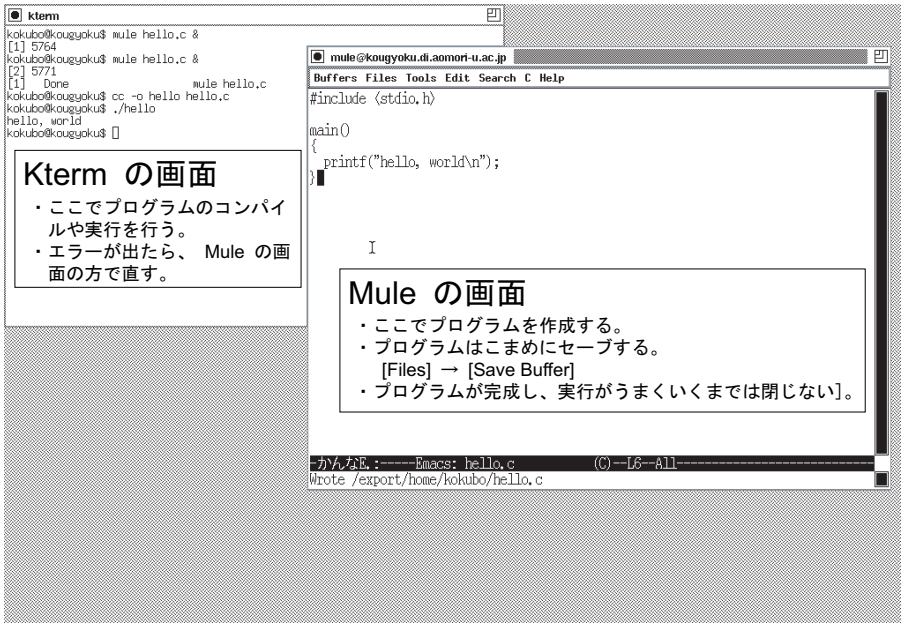
ユーザは自分のファイルに、この 3 種類のパーミッションを自由に設定できる。この機能を使うと、自分で間違って大事なファイルを消さないように設定したり、他人とファイルを共有したり、秘密のファイルを作ったりすることができる。

UNIX システムでは、このパーミッションというしくみによって、セキュリティを確保しつつ、他の人と一緒にプログラムを作ったりすることができるようになっている。

今回は、簡単なプログラムの作り方を説明しながら、パーミッションの設定の仕方を紹介しよう。

10.1　C プログラミング

UNIX システムでプログラムを書くときには、まず Mule をバックグラウンドなどにして他のウィンドウで起動する。そして Mule を使って C などでソース・コード[1]を書いてセーブする。このとき、Mule は終了しないで、ウィンドウを開きっぱなしにしておくとよい。



[1]source code: 「(プログラムの) 元になるコード」という意味。

それから、別のウィンドウでコンパイルするためのコマンドを打つ。もしもエラーが出たら、 Mule のウィンドウに戻って修正し、またセーブする。

そしてうまくコンパイルできたら、実行する。すべてがうまくいくようになったら、 Mule のウィンドウを閉じて終わりにする。

10.1.1　ソース・コードの作成

では、実際に Mule でソース・コードを作成してみよう。

C 言語のテキストの古典中の古典[2]と言えば、ブライアン・カーニハンと、 UNIX を開発したデニス・リッチーが共同で書いた『プログラミング言語 C』(石田 晴久訳 共立出版) だろう。この本の一番最初に載っている、「`hello, world`」と表示するプログラムを作ってみよう。

まず、 Mule で「`hello.c`」いうファイルを作る。UNIX システムでは、 C のプログラムには、ファイル名の最後に「`.c`」を付けるという約束があるので、このような名前を付ける。自分でプログラムを書くときは、「なんとか`.c`」のような感じの名前を付けるといい。

具体的には、次のように Mule を起動する。

```
% mule hello.c &
```

すると、 Mule が自動的に C モードで起動する[3]。なお、 C モードでは、プログラムの入力を助けてくれる機能が使えるが、今回は紹介しない[4]。

「`hello.c`」の中には、次のようなプログラムを書き、[Files] メニューから [Save Buffer] でセーブしよう。このとき、 Mule はセーブするだけで終了はしない。

なお、プログラムの 5 行目で使っている「\」は、¥ のキーを押すと入力できる。[5]。

test.c
```
#include <stdio.h>

main()
{
  printf("hello, world\n");
}
```

以上のファイルが、 C で書かれたソース・コードである。

---

[2]個人的には、初心者にはあまりすすめない。

[3]ファイル名に「`.c`」が付いていると、自動的に C モードになる。

[4]実際には、ウィンドウを複数開かなくてもプログラムの作成、コンパイル、実行まで行なえる。

[5]実は、様々な歴史的な事情により、「\」は、日本語のモードで見ると「¥」に見える。C のプログラムを作るときは、どちらでも一緒である。

10.1.2 コンパイル

C で書かれたソース・コードは、直接コンピュータは理解できないので、C コンパイラで、マシン語に変換する。これをコンパイルという。

では、次のようにして C のコンパイラを起動して、「hello.c」から、マシン語の実行型ファイル「hello」を作ってみよう。ちなみに、 cc は「 C コンパイル – C Compile」という意味である。

```
% cc -o hello hello.c
%
```

うまくいくと、何もメッセージがでないで終了する。エラーがあると、メッセージが出るので、プログラムに打ち間違いがないかどうか、よく確認してみよう。

たとえば、次のように 5 行目の prinf の中の「"hello, world\n"」の終わりの「"」を取ってからセーブして、コンパイルしてみよう。

test.c
```
#include <stdio.h>

main()
{
  printf("hello, world\n);
}
```

すると以下のようなメッセージが出て、5 行目付近に「閉じていない文字列 (unterminated string)」があるというエラー・メッセージがでる。

```
% cc -o hello hello.c
hello.c:5: unterminated string or character constant
hello.c:5: possible real start or unterminated constant
%
```

エラーが出なくなったら、マシン語の実行型ファイル「hello」ができているかを確認してみよう。

```
% ls -F
2000nen         4gatsu          kongetsu        renshuu1.ps     thismonth
2001nen         hello*          kotoshi         renshuu1~
2002nen         hello.c         prog/           renshuu2
3years          hello.c~        renshuu1        tanjoubi
%
```

確かに、「hello」というファイルができていて、実行可能を表す「*」が付いていることがわかる。

10.1.3　コマンドの実行

では、「hello」を実行してみよう。実行するには、ファイル名の先頭に「./」を付けて、「./hello」と打てばいい。

```
% ./hello
hello, world
%
```

すると「hello, world」と表示される。

なお、「./hello」は、相対パスで「現在のディレクトリの中にある hello」という意味だ。UNIX システムでは、このように ファイル名を指定してやるだけ で、実行可能な場合には実行 される。

UNIX システムでは、これまで使ってきた ls、 cal、Mule なども、このように C でソース・コードを書き、それをコンパイルして作られたものである。

なお、 C 以外にも、 B 演習室では、 Java、 BASIC、 FORTRAN、Pascal、 Common LISP、 Scheme、 Perl などのプログラミング言語が使用できる。自分でプログラムを作ってみるといいだろう。

10.2 パーミッション

では、ここでは、前の節で作成した「hello.c」や「hello」を使って、パーミッションについて紹介しよう。

10.2.1 パーミッションの表示: ls -l

パーミッションがどうなっているかを見てみる。

パーミションなどの詳しいファイルの情報を表示させるには、「ls -l」を使う。「-l」オプションは、「long」の意味で、長めの情報を表示する。

では、実際に実行してみよう。

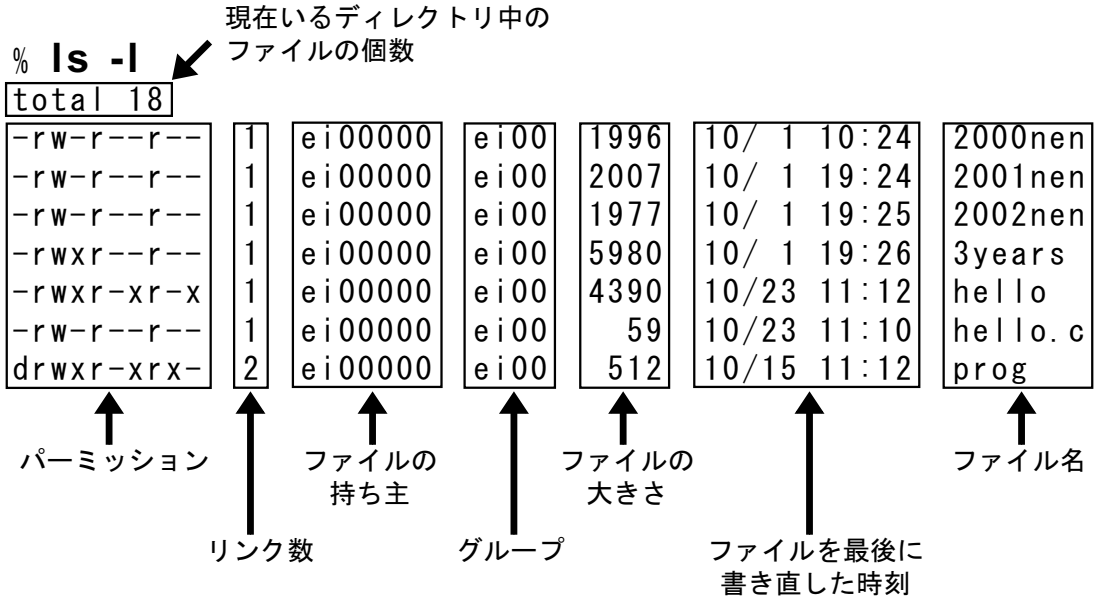


図 10.1: ls -l の出力の見方

この出力の見方は、図 10.1 に示した通りである。ファイルのパーミッションや、持ち主、グループ、ファイルの更新日時などが表示される。

10.2.2 パーミッションの見方

では次に、具体的にパーミッションの見方を説明しよう。

まず、 UNIX システムでは、「読める (r: **r**ead)」「書ける (w: **w**rite)」「実行できる (x: e**x**ecute)」の 3 種類のパーミッションがある。

例えば、図 10.1 を見ると、「2000nen」というファイルのパーミッションは「-rw-r--r--」と表示されている。これは、図 10.2 のように、 4 つのパートに分けて見る。

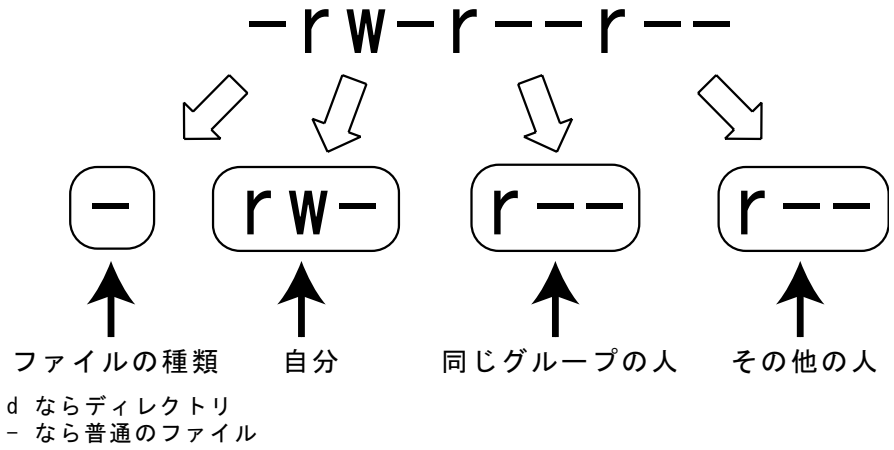


図 10.2: パーミッションの見方

最初の部分は、ディレクトリか普通のファイルかという区別を表す。「d」となっていればディレクトリで、「-」なら普通のファイルである。例えば「-rw-r--r--」ならファイル、「drwxr-xr-x」ならディレクトリだ。

次の部分は 3 つで 1 セットになったパーミッションが、 3 つ並んでいる。

最初の 3 つが自分に対するパーミッション、次の 3 つがファイルのグループに属している人に対するパーミッション、最後の 3 つがそれ以外の人に対するパーミッションを表している。

3 つ 1 セットのパーミッションは、次の図 10.3 のように、最初から順番に rwx と、 3 つのパーミッションを表している。

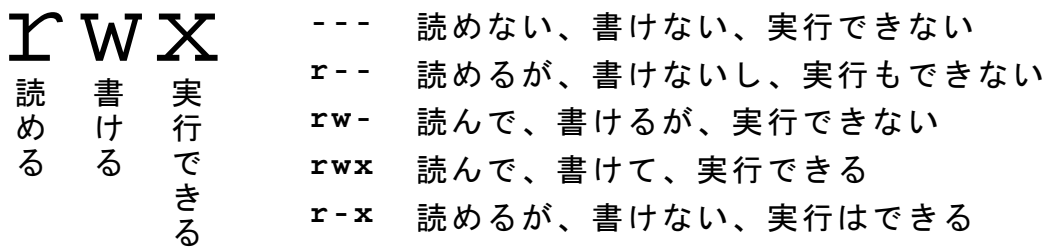


図 10.3: rwx の見方

r の部分に、「r」と書いてあれば読めるファイルで、「-」なら読めないファイルであることを表す。

次の w の部分に、「w」と書いてあれば書き込めるファイル、「-」なら書けないファイルだ。

x の部分に、「x」と書いてあれば実行できるファイルで、「-」なら実行できないファイルだ。なお、ディレクトリの場合、「x」は中に入ることができるもの、「-」は入ることができないものを意味する。

10.2.3 パーミッションの変更: chmod

このパーミッションを変えるには、 chmod コマンドを使う。これは、「**ch**ange **mod**e – モード変更」の略である。chmod は、次のように使う。ここでモードというのは、パーミッションを 8 進数に直したものだ。

```
chmod 「モード」 「ファイル名」
```

モードとは?

モードについては図 10.4 で紹介している。まず、パーミッションは、 rwx の 3 つが並んでいる。このうち、「r」「w」「x」が書いてあるところを「1」、「-」になっているところを「0」にして、 2 進数にする。この 2 進数を、 8 進数に直したものがモードだ。

8 進数と言うと難しそうに聞こえるかもしれないが、1 ケタの 8 進数なので、 10 進数と同じように計算できる。

例えば、「rw-」というパーミッションは、 2 進数で書くと「110」だ。そして、 3 ケタの 2 進数の場合、上の方から順番に、「4 の位」、「2 の位」、「1 の位」になっている。よって、「110」は、「4 の位」と「2 の位」が 1 なので、 $4+2=6$、つまり 8 進数では「6」になる。

モードの計算は、このようにそれほど難しくない。しかし、毎回いちいち計算するのは面倒なので、よく使うパーミッションと、それをモードに直したものを、図 10.5 にあげておいた。

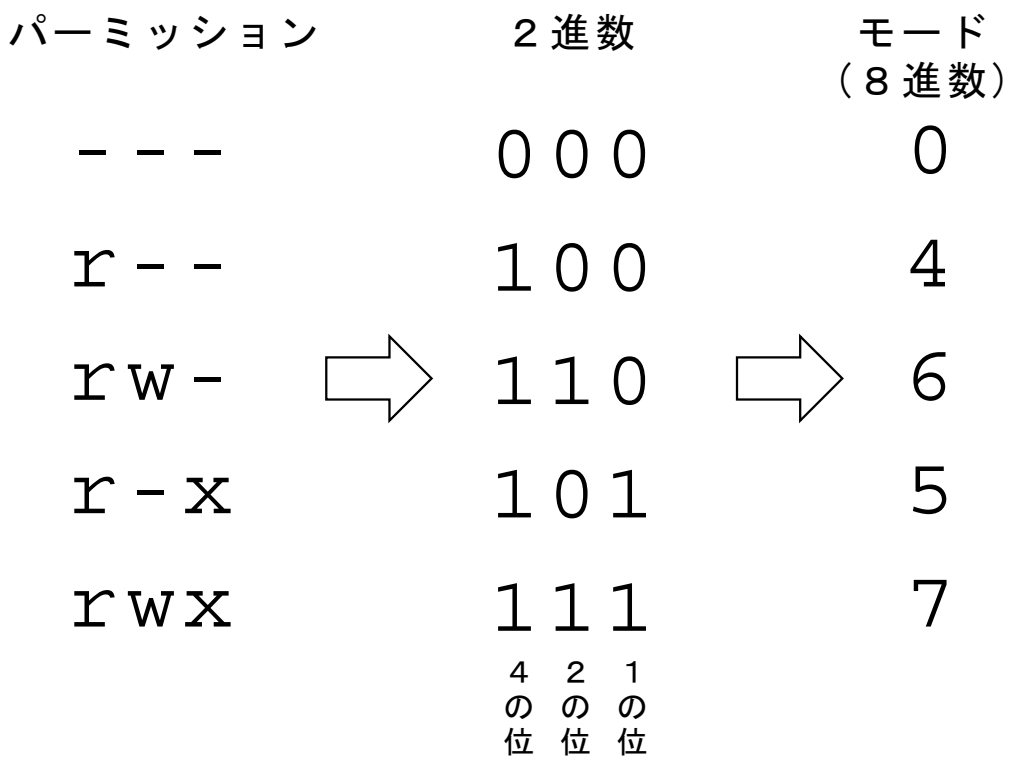


図 10.4: パーミッションからモードを求める方法

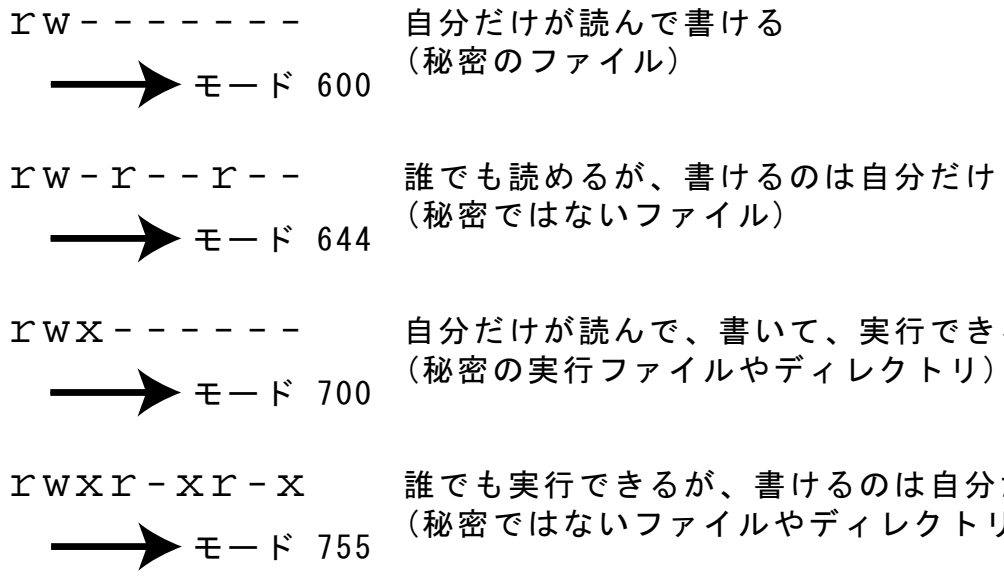


図 10.5: よく使うパーミッションとそのモード

実際のパーミッション操作

では、実際に「hello.c」や「hello」というファイルを使って、パーミッションの変更をしてみよう。

まず、「hello.c」と「hello」のパーミッションを「ls -l」で見てみる。

```
% ls -l
 [中略]
-rwxr-xr-x  1 ei00000  ei00  4390  10/23 11:12  hello
-rw-r--r--  1 ei00000  ei00    59  10/23 11:10  hello.c
 [中略]
%
```

すると「hello」は、誰でも読んで実行できるが、書けるのは自分だけということがわかる。また、「hello.c」の方は、誰でも読めるが、書けるのは自分だけだ。

ここで「hello.c」を、他人から読めない、自分だけの秘密のファイルにしてみよう。自分だけ読み書きできるパーミッションは「rw-------」であり、モードは「600」になる (図 10.5 参照)。よって、次のようにすればよい。

```
% chmod 600 hello.c
%
```

では、 ls -l で確認してみよう。

```
% ls -l
 [中略]
-rw-------  1 ei00000  ei00    59  10/23 11:10  hello.c
 [中略]
%
```

確かに「rw-------」になっている。これで、他人からは読めない。

では次に、ちょっとためしに、この「hello.c」を、誰にも読めない、書けない、実行できないという究極の秘密ファイルにしてみよう。このパーミッションは「---------」になるので、モードは「000」である。というわけで、次のように実行してみよう。

```
% chmod 000 hello.c
%
```

ls -l で確認すると。

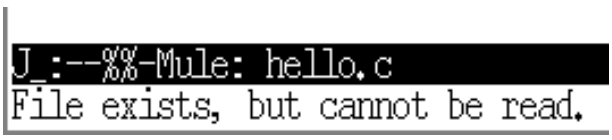
```
% ls -l
 [中略]
----------  1 ei00000  ei00    59  10/23 11:10  hello.c
 [中略]
%
```

では、本当に読んだり書いたりできないか、 Mule を起動して確かめてみよう。

```
% mule hello.c &
%
```

すると次のように、画面下のミニ・バッファに「File exists, but cannot be read. (ファイルはあるけど、読めないよ)」と言われて、画面には何も表示されないはずだ。

確かに読めないし、書けない。



もちろん、 cat で中を見ようとしても、次のように「Permission denied (許可されていません)」とメッセージが出て、表示されない。

```
% cat hello.c
cat: hello.c: Permission denied
%
```

次に自分だけは読めるが、書けないように変更してみよう。なお、間違って消したくないファイルは、このように設定しておくとよい。

この場合、パーミッションは「r--------」なので、モードは「400」だ (図 10.5 参照)。よって、次のようにして変更する。

```
% chmod 400 hello.c
%
```

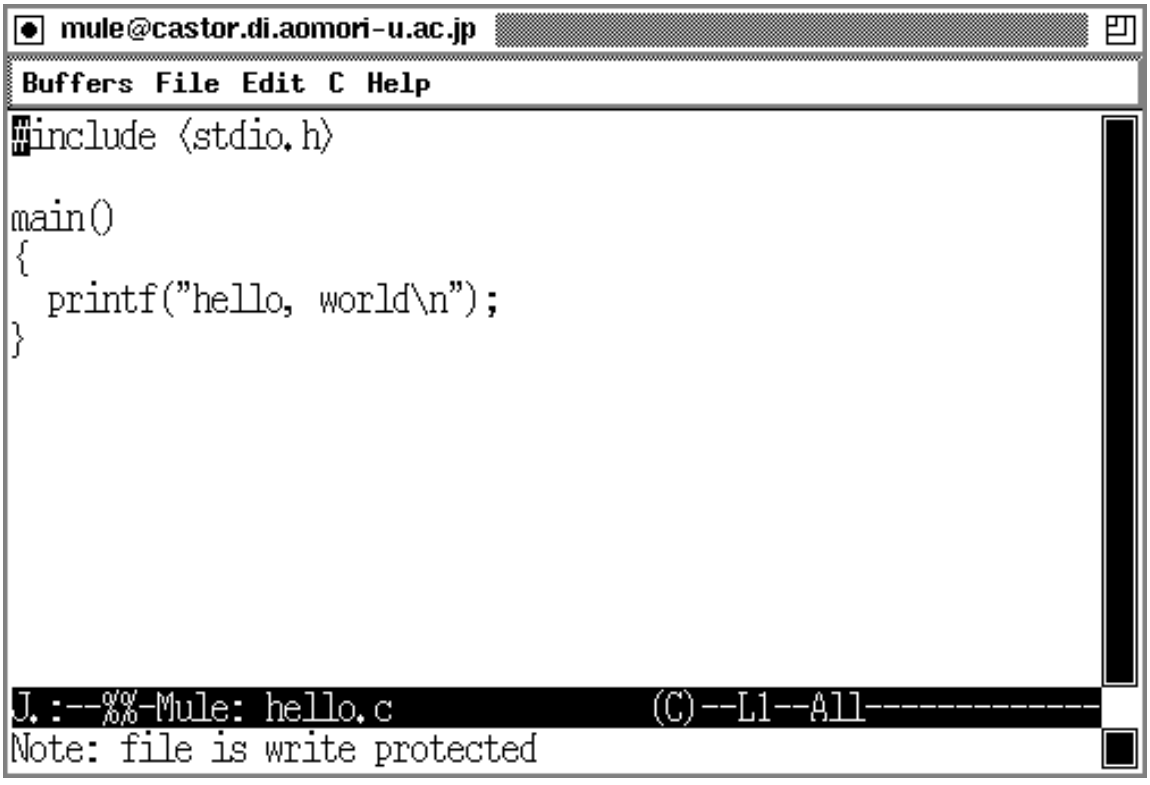
ls -l で確認すると、

```
% ls -l
 [中略]
-r--------  1 ei00000  ei00    59  10/23 11:10  hello.c
 [中略]
%
```

これを Mule でいじってみよう。

```
% mule hello.c &
%
```

すると次の図のように、最初一瞬だけ、画面下のミニ・バッファに「Note: file is write protected (ファイルは書き込み禁止です)」と表示される。そして、ファイルの中身が表示される。



ここに、何か文字を入れようとすると、下の図のようにミニ・バッファに「Buffer is read-only: #<buffer hello.c> (バッファは読むことしかできません)」と表示されて、結局文字を打てないことがわかるだろう。

最後に元にもどしてみよう。最初のパーミッションは rw-r--r-- だったので、モードは 644 だ (図 10.5 参照)。そこで、次のようにする。

```
% chmod 644 hello.c
%
```

ls -l で確認すると、確かに元にもどっている。

```
% ls -l
 [中略]
-rw-r--r--  1 ei00000  ei00    59  10/23 11:10  hello.c
 [中略]
%
```

次に「hello」という実行型ファイルのパーミッションを操作してみよう。まず、このファイルの現在のパーミッションは次のようになっている。

```
% ls -l
 [中略]
-rwxr-xr-x  1 ei00000  ei00  4390  10/23 11:12  hello
 [中略]
%
```

つまり、誰でも読んで実行できるが、書けるのは自分だけだ。これを、読み書きに関してはそのままで、誰も実行できないように変更してみよう。そのようなパーミッションは「rw-r--r--」で、モードは「644」である (図 10.5 参照)。

```
% chmod 644 hello.c
%
```

ls -l で確認してみると、確かに実行できなくなっていることがわかる。

```
% ls -l
 [中略]
-rw-r--r--  1 ei00000  ei00  4390  10/23 11:12  hello
 [中略]
%
```

では、本当に実行できないか、念のため確かめて見よう。

```
% ./hello
./hello: Permission denied.
%
```

確かに、「Permission denied.(許可されていません。)」というメッセージが出て、実行できない。

では、元にもどしてみよう。最初、パーミッションは「rwxr-xr-x」だった、よってモードは「755」だ (図 10.5 参照)。

```
% chmod 755 hello
%
```

ls -l で確認する。

```
% ls -l
 [中略]
-rwxr-xr-x  1 ei00000  ei00  4390  10/23 11:12  hello
 [中略]
```

では、実行できるか試して見よう。

```
% ./hello
hello, world
%
```

確かに実行できることがわかる。

この章で紹介したコマンド

パーミッションの操作

chmod : パーミッションの変更

使い方: chmod 「モード」 「ファイル名 (複数指定可)」

C コンパイラ

cc : C コンパイラ

使い方: cc -o 「実行型ファイル名」 「ソースのファイル名」

第11章　プログラミング言語の使い方

ここでは、 UNIX システムで使用可能なプログラミング言語を簡単に紹介する。紹介しているプログラムは、円の半径を読み込んで、円周 ($2 \times \pi \times$半径) と円の面積 ($\pi \times$半径$\times$半径) を求めて表示するものである。

11.1　コンパイラ系

コンパイラ系の各言語は本格的なプログラミングを行うのに向いている。特に C はさまざまな場面で用いられていて、応用範囲も広い。コンパイラ系では、ソース・プログラムを書き、それをコンパイラでマシン語の実行型ファイルに変換してから実行する。以下に紹介した C、Pascal、FORTRAN では、コンパイルの方法は次のように共通の形式になっている。

「コンパイラ・コマンド」 -o 「実行型ファイル名」 「ソース・ファイル名」

以下の例では、「実行型ファイル名」に、「ソース・ファイル名」から拡張子を取り除いたものを使っているが、別な名前[1]を指定してもいい。

また、コンパイルしてできた実行型ファイルを実行するには、次のようにする。

./「実行型ファイル名」

11.1.1　C

プログラム例

circle.c

```
#include <stdio.h>

int main()
{
  /* 使用する変数の指定 */
  double pi, hankei, enshuu, menseki;
  /* 円周率 */
  pi = 3.141593;
```

[1]たとえば「ls」、「rm」なども指定できる。ただし、そうするとややこしくなる。自分なりに適当な名前を考えて付けよう。

```
  /* 半径を端末から読み込む */
  printf("円の半径を入力してください\n");
  scanf("%lf", &hankei);

  /* 円周と面積を求める */
  enshuu = 2.0 * pi * hankei;
  menseki = pi * hankei * hankei;

  /* 表示 */
  printf("半径 = %f\n", hankei);
  printf("円周 = %f、面積 = %f\n", enshuu, menseki);

  return(0);
}
```

実行例

```
% cc -o circle circle.c
% ./circle
円の半径を入力してください
20
半径 = 20.000000
円周 = 125.663720、面積 = 1256.637200
%
```

11.1.2 Pascal

プログラム例

circle.p

```
program circle(input, output);
 使用する変数の指定
var pi     : real;
   hankei  : real;
   enshuu  : real;
   menseki : real;

begin
   { 円周率 }
   pi := 3.141593;

   { 半径を端末から読み込む }
   writeln(' 円の半径を入力してください');
   readln(hankei);

   { 円周と面積を求める }
   enshuu := 2.0 * pi * hankei;
   menseki := pi * hankei**2;

   { 表示 }
   writeln(' 半径 = ', hankei);
   writeln(' 円周 = ', enshuu, '、面積 = ', menseki);
end.
```

実行例

```
% gpc -o circle circle.p
% ./circle
円の半径を入力してください
20
半径 =  2.000000000000000e+01
円周 =  1.256637200000000e+02、面積 =  1.256637200000000e+03
%
```

11.1.3 FORTRAN

プログラム例

circle.f

```
      PROGRAM CIRCLE
C 使用する変数の指定
      IMPLICIT NONE
      REAL PI, HANKEI, ENSHUU, MENSEKI
C 円周率
      PI = 3.141593
C 半径を端末から読み込む
      WRITE(*,*) '円の半径を入力してください'
      READ(*,*) HANKEI
C 円周と面積を求める
      ENSHUU = 2.0 * PI * HANKEI
      MENSEKI = PI * HANKEI**2
C 表示
      WRITE(*,*) '半径 = ', HANKEI
      WRITE(*,*) '円周 = ', ENSHUU, '、面積 = ', MENSEKI
      STOP
      END
```

実行例

```
% f77 -o circle circle.f
/usr/lib/libg2c.so: warning: tempnam() possibly used unsafely; consider using mk
stemp()
% ./circle
 円の半径を入力してください
20
 半径 =   20.
 円周 =   125.663719、面積 =   1256.63721
%
```

11.2 バイト・コンパイラ系

11.2.1 Java

バイト・コンパイラ系の Java では、ソース・プログラムを書き、それをコンパイラ (javac) で中間コードと呼ばれる形式に変換し、Java 仮想機械と呼ばれるプログラム (java) にロードして実行する。このことによって、さまざまなマシン上で同じように動くプログラムを書くことができ、 WWW 関係のプログラミングでよく用いられている。

まず、コンパイルするには次のようにする。

```
javac 「ソース・ファイル名」
```

Java 仮想機械にロードして実行するには次のようにする。なお、「クラス名」というのは、以下の例では「Circle」である。

```
java 「クラス名」
```

プログラム例

Circle.java

```
import java.io.*;
class Circle {
  public static void main(String[] args) throws IOException {
    /* 円周率 */
    double pi = 3.141593;

    /* 半径を端末から読み込む */
    System.out.println("円の半径を入力してください");
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    String str = br.readLine();
    double hankei = Double.parseDouble(str);

    /* 円周と面積を求める */
    double enshuu = 2.0 * pi * hankei;
    double menseki = pi * hankei * hankei;

    /* 表示 */
    System.out.println("半径 = " + hankei);
    System.out.println("円周 = " + enshuu + "、面積 = " + menseki);
  }
}
```

実行例

```
% javac Circle.java
% java Circle
円の半径を入力してください
20
半径 = 20.0
円周 = 125.66372、面積 = 1256.6372
%
```

11.3 インタプリタ系

インタプリタ系では、プログラムを書き、それをインタプリタにロードして実行する。手軽にプログラミングできるのが特徴である。Perl、 Python、 Ruby は、 WWW で掲示板などに使われる CGI を作るのによく用いられる言語だ。また、ここでは紹介していないが、 Web ブラウザ上で動くインタプリタ系の言語の一種である JavaScript は、HTML と組み合わせて利用し、たとえばマウスをボタンに重ねるとボタンの画像が変わったりするようなしくみに使われている。

以下に紹介したもののうち、 Python、Perl、 Ruby の場合、個別に書かなかったが、プログラムを書いたらパーミッションを実行可能にして (「chmod 755 「プログラム・ファイル名」」) から実行すること。BASIC の実行の仕方はこれらと異なっているので個別に紹介した。

11.3.1 Python

プログラム例

circle.py

```
#!/usr/local/bin/python
import sys
# 円周率
pi = 3.141593

# 半径を端末から読み込む
print "円の半径を入力してください";
hankei = float(sys.stdin.readline());

# 円周と面積を求める
enshuu = 2.0 * pi * hankei;
menseki = pi * hankei * hankei;

# 表示
print '半径 = ', hankei;
print '円周 = ', enshuu, '、面積 = ', menseki;
```

実行例

```
% ./circle.py
円の半径を入力してください
20
半径 =  20.0
円周 =  125.66372 、面積 =  1256.6372
%
```

11.3.2 Perl

プログラム例

circle.pl

```
#!/usr/bin/perl
# 円周率
$pi = 3.141593;

# 半径を端末から読み込む
print "円の半径を入力してください\n";
$hankei = <STDIN>;
chomp $hankei;

# 円周と面積を求める
$enshuu = 2.0 * $pi * $hankei;
$menseki = $pi * $hankei * $hankei;

# 表示
print "半径 = $hankei\n";
print "円周 = $enshuu、面積 = $menseki\n";
```

実行例

```
% ./circle.pl
円の半径を入力してください
20
半径 = 20
円周 = 125.66372、面積 = 1256.6372
%
```

11.3.3 Ruby

プログラム例

circle.rb

```
#!/usr/local/bin/ruby
# 円周率
pi = 3.141593

# 半径を端末から読み込む
print "円の半径を入力してください\n"
hankei = readline()
hankei = hankei.to_f

# 円周と面積を求める
enshuu = 2.0 * pi * hankei
menseki = pi * hankei * hankei

# 表示
print "半径 = ", hankei, "\n"
print "円周 = ", enshuu, "、面積 = ", menseki, "\n"
```

実行例

```
% ./circle.rb
円の半径を入力してください
20
半径 = 20.0
円周 = 125.66372、面積 = 1256.6372
%
```

11.3.4 BASIC

BASIC の場合、プログラムを書き、 BASIC を起動 (この場合は「bwbasic」と入力) して、その中でプログラムをロード (load "「ファイル名」") し、「run」コマンドで実行する。BASIC を終了するには、「quit」コマンドである。

プログラム例

circle.bas

```
10 REM 円周率
20 LET PI = 3.141593
30 REM 半径を端末から読み込む
40 PRINT "円の半径を入力してください"
50 INPUT HANKEI
60 REM 円周と面積を求める
70 LET ENSHUU = 2 * PI * HANKEI
80 LET MENSEKI = PI * HANKEI * HANKEI
90 REM 表示
100 PRINT "半径 ="; HANKEI
110 PRINT "円周 ="; ENSHUU; "、面積 ="; MENSEKI
120 END
```

実行例

```
% bwbasic
Bywater BASIC Interpreter/Shell, version 2.20 patch level 1
Copyright (c) 1993, Ted A. Campbell

bwBASIC: load "circle.bas"
bwBASIC: run
円の半径を入力してください
? 20
半径 = 20
円周 = 125.6637200、面積 = 1256.6372000
bwBASIC: quit
%
```

余談だが、最近の Windows などでは「cscript」というコマンドで、 VBscript(Visual Basic Scripting Edition) などのプログラムをロードして使うこともできるようになっている。VBscript で BASIC のものと同様なプログラムを書くと以下のようになる。同じ BASIC という名前がついていても、見た目が随分異なったプログラムになることがわかるだろう。

プログラム例

circle.vbs

```
' 円周率
pi = 3.141593

' 半径を端末から読み込む
Wscript.echo("円の半径を入力してください")
hankei = CDbl(Wscript.StdIn.ReadLine)

' 円周と面積を求める
enshuu = 2 * pi * hankei
menseki = pi * hankei * hankei

' 表示
Wscript.echo("半径 = " & hankei)
Wscript.echo("円周 = " & enshuu & "、面積 = " & menseki)
```

実行例

実行するには、 Windows XP で「スタート」→「すべてのプログラム」→「アクセサリ」→「コマンド プロンプト」として、「コマンド プロンプト」をまず起動する。起動したら、以下のように実行する。

```
E:¥ei00¥ei00000>cscript circle.vbs
Microsoft (R) Windows Script Host Version 5.6
Copyright (C) Microsoft Corporation 1996-2001. All rights reserved.

円の半径を入力してください
20
半径 = 20
円周 = 125.66372、面積 = 1256.6372

E:¥ei00¥ei00000>
```

第12章　シェル

システムを包み込むシェル

「シェル (shell)」とは、英語で「貝殻」のことを言う。昭和シェル石油のマークは、貝殻がモチーフになっているが、これは英語で考えるととてもわかりやすい。

ところで、今回紹介するシェルとは、 UNIX システムのコアの部分を包み込み、ユーザとシステムの仲介役をしているしくみのことだ。シェルは、 UNIX システムでコマンドの入力を手助けしてくれる。シェルにはとても強力な機能が備わっていて、これを使いこなせれば、UNIX システムを少ない入力で、これまでの何倍も活用することができる。

12.1　ユーザ・インターフェース

UNIX システムは、基本的にコマンドをキーボードから打ち込むコマンド・ラインをベースにしたユーザ・インタフェース[1]を採用している。一方、 MacOS や Windows は、アイコンをマウスで操作する GUI[2] をベースにしたシステムだ。

シェルは、コマンド・ライン・ベースのインタフェースを使うときに、ユーザをサポートする。その役割をはっきりさせるために、コマンド・ライン・ベースのインタフェースと GUI の話を最初にしよう。

コマンド・ライン・ベースと GUI、この 2 種類のインタフェースには、それぞれ特徴がある。

12.1.1　GUI の特徴

まず GUI の方は、絵の描かれたカードを持って、コンピュータとお話をしているというイメージを持ってもらえばいいだろう。

例えば、 WWW にアクセスするときには、 Web ブラウザの描かれたカードを持ってコンピュータに見せる。すると、コンピュータが Web ブラウザを起動してくれるというわけだ。

このシステムには、絵が描かれたカードが最初から用意されていて、はじめてコンピュータを使う人にも、「なんとなく」使い方がわかり、とっときやすい。このシステムは、ブラウザとか、お絵描きソフトとか、ゲームなどを単に起動して使うときにはとても便利だ。

---

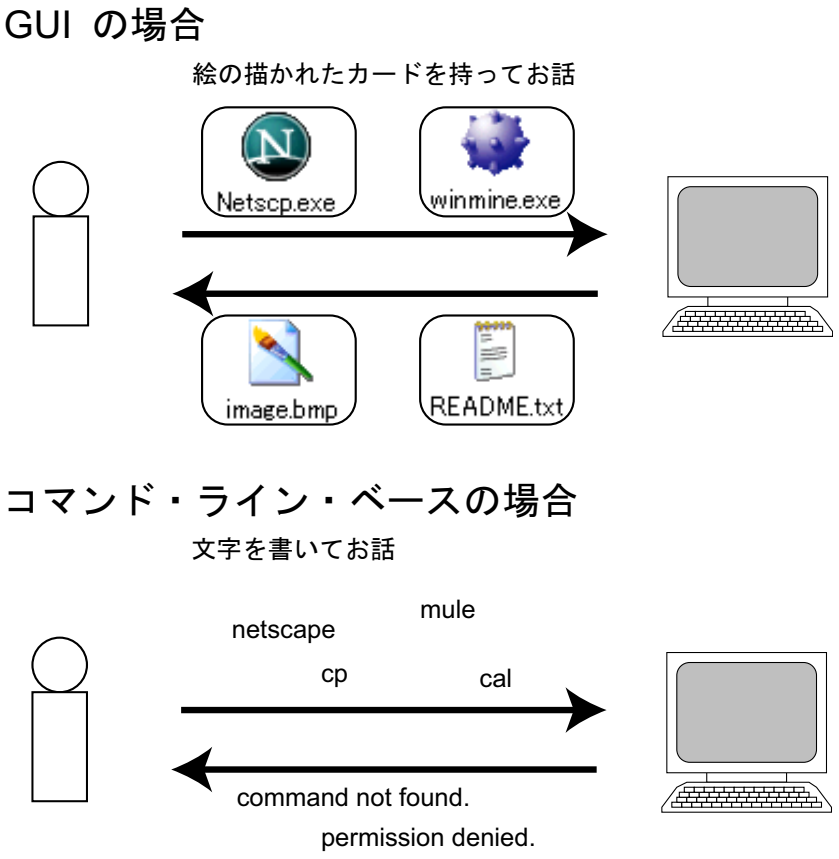
[1] user interface: ユーザがシステムを使うときに相手にする部分

[2] Graphical User Interface: グラフィカル・ユーザ・インタフェース

12.1.2 コマンド・ライン・ベースのインタフェースの特徴

一方、コマンド・ライン・ベースのシステムの方は、文字を書いてコンピュータとやりとりをしているようなイメージになる。

つまり、例えば WWW にアクセスするには「netscape」とかいう文字を書いて、コンピュータに指示を出し、 Web ブラウザを起動してもらう。このコンピュータに指示をするときに使う言葉が、コマンドである。このシステムでは、言葉 (コマンド) を知らないと、コンピュータと全くお話できないため、最初に少し勉強する必要がある。また、文字を一々書かなければならないので、単にブラウザや、お絵描きソフト、ゲームなどを起動するにも、たくさんタイプしなければならない。



12.1.3 GUI の欠点

こう書いてくると、 GUI の方が便利そうだが、本当にそうなのだろうか?

GUI にも問題はある。まず、コンピュータに細かい指示を出すには向いていないことだ。

例えば、ある箱 (フォルダ) にファイルが入っているとしよう。このファイルを持って、別の箱に突っ込んだら、コンピュータはどう判断するだろうか？ ファイルの移動？ それともコピー？

実際 Windows では、同じドライブ間で実行した場合と、別なドライブ間で実行した場合で、異なった判断をコンピュータがする[3]。また、それが通常のファイルの場合と、実行型ファイルの場合でも異

[3]同じドライブにあるフォルダ間で実行すると、コンピュータは「移動」を実行する。別のドライブであれば、「コピー」

なる[4]ことがある。しかも、これらの挙動は、同じ Windows でもバージョンによって異なっていたりすることがあるのである。

また GUI は、抽象的な指示を出すのには向いていない。例えば、「data001」～「data100」という 100 個のディレクトリ (フォルダ) を作る必要があったとしよう。Windows で GUI のみを使う場合、右クリックで［新規作成］→［フォルダ］を実行し、フォルダの名前 (「data001」～「data100」) を指定するという作業を 100 回実行しなければならない。

また GUI は、いくつかの連続した作業を一括して自動化するのにも向いていないのである[5]。

12.1.4 コマンド・ライン・ベースの欠点を補完するシェル

コマンド・ライン・ベースの欠点は、基本的にはたくさんタイプする必要があることだ。UNIX システムのシェルは、この欠点を補完する。シェルは、過去に打ったコマンドを覚えておいたり、同じような繰り返しの作業を簡単な指定で何百回も実行させたり、いくつかのコマンドをまとめて自動化する機能を持っている。この機能を活用すると、ユーザは少ない入力で、さまざまなことを実行できる。

を実行する。

[4]Windows のバージョンによっては、通常のファイルと異なり、実行型のファイルでこの作業を行なうと、コンピュータはショートカットを作るだけである。

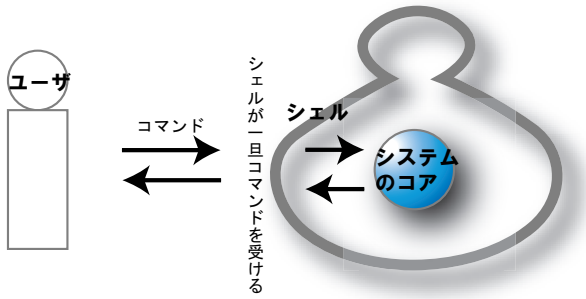
[5]最近のバージョンの Windows では GUI のツールを自動化してコントロールするために WSH(Windows Scripting Host) のようなしくみが使えるようになってきている

12.2 シェルの役割

シェルは、 UNIX システムのコアの部分を包み込んでいる。ユーザと UNIX システムの間にシェルがある。ユーザが UNIX システムを使うときに、相手をしているのは、実は全部シェルなのだ。

ここまで、読んできて、「は?」と思うかもしれない。「え? シェルなんて今日の今日まで知らなかったぞ。一体、いつそんなもの相手にしたっけ?」と思うかもしれない。

ところが、実は UNIX システムでコマンドを打つと、シェルが一回受け取って、シェルがシステムに渡してから実行されているのだ (ユーザの知らないうちに)。



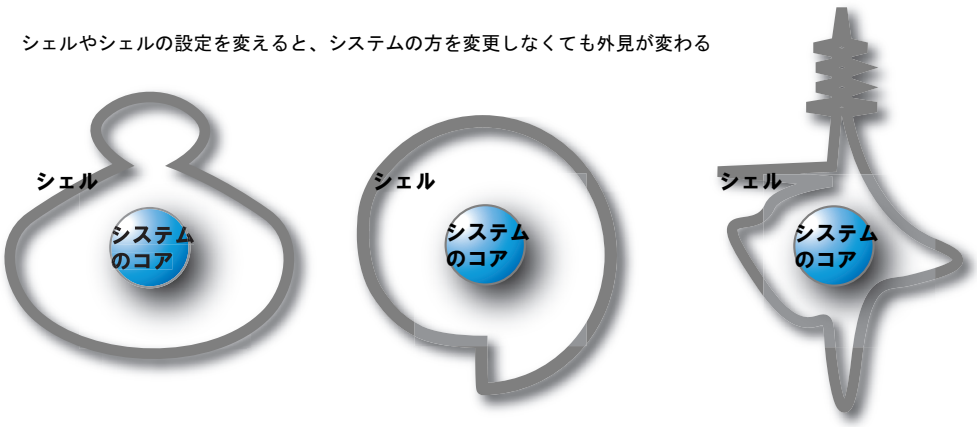
なぜ、そんな余計なものが間に入っているのだろう?

例えば、「自分好みにシステムを改造したい」と、そう思ったことはないだろか?

そんなとき、コマンドをダイレクトにシステムに渡すように設計されていたら、システムそのものを改造しなければならない。ユーザとシステムの間にシェルでワン・クッション入れておけば、システムを直接改造しなくても、シェルの方をいじるだけでいろいろ自分好みにカスタマイズすることができるのである。

また、間にシェルが入っていてくれることで、システムの方には単純なコマンドだけを用意しておいて、ユーザ相手の部分はシェルがていねいに面倒を見るということも可能になる。

要するに システムは単純に、後はシェルにおまかせ というコンセプトである。



12.3 いろいろなシェル

では、シェルを紹介しよう。

代表的なシェルには sh、csh、tcsh、 ksh、 bash、 zsh などがある。

これらを一時的に使ってみた場合には、単に「`sh`」とか「`csh`」と打てばいい。なお、デフォルトでは「tcsh」になっている。

それぞれ簡単に紹介しておこう。

sh 一番単純なシェル。ボーン (Borune) さんが作ったので、ボーン・シェルとも言う。

csh BSD のビル・ジョイくんが作った、C 言語に似たプログラム機能を持ったシェル。

tcsh TENEX というシステムにあった、補完機能などを組み込んだ、csh の機能拡張版。ただし、FreeBSD の場合、 csh と tcsh は同じになっている。

ksh コーン (Korn) さんの作った sh の機能拡張版で、AT & T 社の製品。ただし、 ksh 互換のフリー・ソフトも作られていて、 FreeBSD にはそちらが入っている。

bash GNU のソフトでボーン・アゲイン (再び)・シェルという。sh の機能拡張版で、 tcsh などの機能も取り入れている。

zsh 作者曰く、究極のシェル。sh と csh の機能を取り入れ、独自のプログラム機能を持っている。

12.4　シェルの機能

12.4.1　ヒストリー機能

シェルの代表的な機能の一つがヒストリー機能である。これは要するに、昔打ったコマンドを、システムが覚えていて、それを後から使えるということだ。

では、具体的に使ってみよう。

まず、シェルにコマンドを覚えさせてみよう。who 、 ls 、 cal 、 ps などと 4 つくらいコマンドを実行してみよう。ここまで打ったコマンドのリストは記憶されていて「history」と打つと表示される。なお、システムの設定により、過去 100 個分を記憶しているため、非常に長い出力がでる。

```
% history
 [中略]
   101  11:23   who
   102  11:23   ls
   103  11:23   cal
   104  11:24   ps
   105  11:25   history
%
```

この表示は、最初が「何番目」のコマンドか、次が実行したのが「何時」か、最後が「どんなコマンド」を実行したのかを表している。過去 100 個分を記憶しているために、ここでは login して最初に実行した who が 101 番目と表示されている。

!「数字」:「数字」番目のコマンド呼び出し

シェルが記憶しているコマンドは簡単に呼び出せる。例えば、 101 番目の w をもう一度実行するには「!101」と入力すればいい。全く同様に、 103 番目の cal なら「!103」である。

では、試してみよう。

```
% !101
who
ei00000  ttyp0    11/01 01:06   (:0.0)
% !103
cal
     11 月 2000
日 月 火 水 木 金 土
          1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30

%
```

このようにとても簡単である。

!「文字」: もっとも最近打った「文字」で始まるコマンド呼び出し

ヒストリーには、別な呼び出し方もある。一番最近打った、「c」で始まるコマンドは、この例では cal だが、これを呼ぶには「!c」と入力すればよい。もちろん、 history を呼ぶには単に「!h」である。

では、実行してみよう。

```
% !c
cal
     11 月 2000
日 月 火 水 木 金 土
          1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30

%
```

なお、これらの機能は、プログラム書いて、コンパイルするときに使うと便利だ。つまり、コンパイルするには「cc -o 「実行型ファイル名」 「コンパイルするファイル名」」だが、これは一度実行してしまえば、コンパイルしなおすには、単に「!c」で済んでしまうというわけだ。

!!: 直前のコマンドのくり返し

また、直前のコマンドをくり返すこともできて、それは単に「!!」と入力すればよい。

たとえば、直前に cal を実行していれば、「!!」で cal が実行される。

```
% !!
cal
     11 月 2000
日 月 火 水 木 金 土
          1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30

%
```

12.4.2　コマンド・ライン編集機能

tcsh などの場合、次のような機能もある。

まず、↑ (または Mule と同様に C-p) を押していくと、一つずつ前のコマンドを呼び出すことができる。また、行き過ぎた場合、↓ (C-n) で一つ進むことができる。

更に、呼び出したコマンドは、←(C-b) や、→(C-f) や、Delete などを使って、書き直すことができる。

これらをコマンド・ライン編集機能という。

12.4.3　コマンド別名

csh 以降に作られたシェルには、たいていコマンドに別名を付ける機能がついている。このような機能のことを、「エイリアス – alias」という。

例えば、単に「ls」と打つと、ディレクトリもファイルも同じように見えてしまう。そして、ディレクトリの後ろに「/」を付けたり、実行型ファイルの後ろに「*」を付けてて表示させるには、「ls -F」と打てばいいことは既に紹介した。

しかし、それを毎回打つのは面倒である。そこで、エイリアス機能を使うと、「ls」と打っただけでも、「ls -F」と打った場合と同じことをするように設定できる。

具体的には「alias ls 'ls -F'」としてやる。これで「ls」と打つと、「ls -F」が実行されるようになる。なお、このように設定したエイリアスは、そのシェルを終了するまで有効である。これを毎回使いたい場合には、後ろの方で紹介するようにシェルの初期設定ファイルに記述しておけばよい。

では、実際にやってみよう。まず、「ls」と打ってみよう。

```
% ls
2000nen         Circle.java     circle.pl       kongetsu        renshuu2
2001nen         circle          circle.py       kotoshi         tanjoubi
2002nen         circle.bas      circle.rb       prog            thismonth
3years          circle.c        hello           renshuu1
4gatsu          circle.f        hello.c         renshuu1.ps
Circle.class    circle.p        hello.c~        renshuu1~
%
```

これでは、ディレクトリと普通のファイルの区別は全くつかない。かといって、毎回「ls -F」と打つのも面倒である。そこで、エイリアス機能を使う。

```
% alias ls 'ls -F'
%
```

では、「ls」と打ってみよう。

```
% ls
2000nen         Circle.java     circle.pl*      kongetsu        renshuu2
2001nen         circle*         circle.py*      kotoshi         tanjoubi
2002nen         circle.bas      circle.rb*      prog/           thismonth
3years          circle.c        hello*          renshuu1
4gatsu          circle.f        hello.c         renshuu1.ps
Circle.class    circle.p        hello.c~        renshuu1~
%
```

確かに「ls -F」と入力した場合と同じになる。

エイリアス機能は、長いコマンドを打つのが面倒なときに使ったり、上の例のように、毎回使うオプションをはぶいたりするのに使うと便利である。

実は、「alias cp rm」といったアブナイこと[6]も、できてしまうので、使うときは注意しよう。

12.4.4 名前の補完

tcsh などの特徴に、ファイル名やコマンドの補完機能がある。これは、該当するファイルやコマンドが 1 つしかない場合、Tab を押すと補完してくれる機能である。

この機能は、口で説明するよりは、やってみた方が早いので、実際にやってみよう。

まず、「ls」と入力してみよう。

```
% ls
2000nen         Circle.java     circle.pl*      kongetsu        renshuu2
2001nen         circle*         circle.py*      kotoshi         tanjoubi
2002nen         circle.bas      circle.rb*      prog/           thismonth
3years          circle.c        hello*          renshuu1
4gatsu          circle.f        hello.c         renshuu1.ps
Circle.class    circle.p        hello.c~        renshuu1~
%
```

この中で、「p」で始まるのは、「prog」というディレクトリしかない。この場合、 cd で「prog」の中に移動するには、「cd prog」と入力すればよいのだが、このコマンドを全部打つ必要はない。

まず、「cd p」と打つ。

```
% cd p
```

ここで、 Tab を押すと、「p」で始まるファイルやディレクトリは「prog」の他にないので、次のように名前が補完される。

[6] こうすると、「cp」と打っても「rm」になってしまう。

```
% cd prog
```

また、上の例では、「c」で始まるファイルは「circle」～「circle.rb」まで 8 個ある。このとき Mule で「circle.c」を編集したければ、「mule c」まで入力し、Tab を押してあげると、「mule circle」まで補完される。その後、「.c &」だけを補えば「mule circle.c &」となり、入力の手間がはぶける。

12.5　シェル・スクリプト

シェルには、いくつかのコマンドをまとめて自動する機能がある。シェル・スクリプトという簡単なプログラムを書いて、実際に実行してみよう。

まず、次のように「test1」というファイルを Mule で作る。

```
% mule test1 &
%
```

test1 の中身には、次のように書いて、セーブして Mule を終了する。

test1
```
ls -F
cal
```

次に「test1」のパーミッションを、実行可能に変更してみよう。

```
% chmod 755 test1
%
```

すると、この「test1」は、なんと普通のコマンドのように実行可能になり、中に書いたコマンドをそのまま実行してくれる。

```
% ./test1
2000nen         Circle.java     circle.pl*      kongetsu        renshuu2
2001nen         circle*         circle.py*      kotoshi         tanjoubi
2002nen         circle.bas      circle.rb*      prog/           test1*
3years          circle.c        hello*          renshuu1        thismonth
4gatsu          circle.f        hello.c         renshuu1.ps
Circle.class    circle.p        hello.c~        renshuu1~
     11月 2000
日 月 火 水 木 金 土
          1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30

%
```

このような簡単なプログラムをシェル・スクリプトと呼ぶ。これを使うと、いくつかのコマンドをまとめて一気に実行できる。

また、ここでは触れないが、条件分岐や、ループなどを使った高度なプログラムもシェル・スクリプト中では行なうことも可能である。

12.6 ワイルドカード

シェルではいつくかの記号が特別の意味を持っている。今回は、その内のワイルドカードと呼ばれているものを紹介しよう。

ワイルドカードは、似たような名前のファイルを、コマンド一発で何か同じ処理をしたいときによく使う。

たとえば、「I」、「my」、「me」、「mine」、「the」、「this」、「these」、「those」、「who」、「which」、「when」というファイルがあったとしよう。

このとき、代表的なワイルドカードを使うとどうなるか紹介しよう。

「*」　0 文字以上[7]のさまざまな文字を表す

例えば「th*」と書くと「the」、「this」、「these」、「those」のような、「th」で始まるすべてのファイルにマッチする。もしも、「th」というファイルがあれば、それにもマッチする。

また、「*e」と書くと、「e」で終わる「me」、「mine」、「the」、「these」、「those」にマッチする。

同様に「t*e」と書くと、「the」、「these」、「those」にマッチする。

更に単に「*」と書くと、すべてのファイルにマッチする。

「?」　何か 1 文字を表す

例えば「m?」なら、 最初が「m」で、次に何か 1 文字だけあるファイルということで、「my」と「me」にマッチする。

また、「th?se」なら、「these」と「those」にマッチする。

これらのワイルドカードを使うと複数のファイルを一気に処理できる。

例えば、「cat th*」と打てば、「cat the this these those」と打ったのと同じことになる。

また、「*」は、一気にファイルを消すのによく使う。例えば、「rm th*」と打てば、「the」、「this」、「these」、「those」 を一気に消すことができる。

また Mule は「~」の付いたバックアップ・ファイルを作成するが、これを全部消すには「rm *~」と打てばよい。なお、間違って「rm *」とすると全部のファイルが消えてしまうので、注意しよう。

[7]つまり 0 文字でもいい

12.7　シェルの初期設定をカスタマイズ

tcsh や csh の初期設定ファイルは、「.cshrc」である。この中をいじると、シェルの初期設定を変えることができる。これらの「.」ではじまるファイルは隠しファイルになっていて、普通に ls を実行しても表示されない。「.」ではじまるファイルを表示するには、ls に「-a」オプションを指定する。

では、「cp .cshrc .cshrc-orig」などと入力して、現在のシェルの初期設定ファイルバックアップしてから、Mule で中をのぞいてみよう。

```
% mule .cshrc &
%
```

ファイルの中身は、次のようになっている。

```
#
# .cshrc - Csh And Tcsh Personal Initialization File
#
# see csh(1) or tcsh(7)
#

# default file permission
umask 022

# command search path
set path=(~/bin /bin /usr/bin /sbin /usr/sbin ¥
/usr/X11R6,local,local/java/bin /usr/games)

if ($?prompt == 0) exit

# manual search path
setenv MANPATH /usr/share/man:/usr/X11R6/man:/usr/local/man:/usr/local/java/man

# block size unit
setenv BLOCKSIZE K

# prompt setting
set prompt="‘whoami‘@‘hostname |sed ’s/.*//’‘% "

# editor and pager
setenv EDITOR mule
setenv TEXEDIT ’mule +%d %s’
#setenv PAGER ’jless -i -e’
setenv PAGER ’jless’
setenv JLESSCHARSET japanese

# japanese input method
setenv CANNAHOST localhost
setenv XMODIFIERS @im=kinput2

# mail and news
setenv ORGANIZATION ’Dept. of Info. Sys. Eng., Aomori Univ.’
setenv NNTPSERVER news0.mono.aomori-u.ac.jp
setenv SMTPSERVER msedu0.edu.aomori-u.ac.jp
setenv POP3SERVER msedu0.edu.aomori-u.ac.jp
```

```
setenv HTTP_PROXY 'http://cache.edu.aomori-u.ac.jp:8080/'

# aliasese
#alias less 'jless'
#alias ls 'ls -CF'
#alias cp 'cp -i'
#alias rm 'rm -i'
#alias mv 'mv -i'
#alias cd 'cd ¥!*; echo $cwd'
#set noclobber

# other setting
set ignoreeof
set history = 100
set savehist = 100
set symlinks = expand

# for tcsh
if ($?tcsh) then
unset autologout
if ($?EMACS) then
unset edit
stty nl
endif
endif


# locale
setenv LANG ja_JP.EUC
```

このファイルの中で「#」で始まっているところは、コメントである。

これを見ると、半分よりも少し後の方で、いくつか alias を設定しているが、全部コメントになっているのがわかる。これらのコメントをはずすと、 ls、 cp、 rm、 mv の動作を変えることができる。

```
# aliasese
alias ls 'ls -CF'              ディレクトリや実行型ファイルに目印が付く。
alias cp 'cp -i'               上書きしそうなときに警告する。
alias rm 'rm -i'               ファイルを本当に消していいか確認する。
alias mv 'mv -i'               上書きしそうなときに警告する。
alias cd 'cd \!*; echo $cwd'   移動先のディレクトリを表示する。
set noclobber                   リダイレクトの上書きを警告する。
```

これはあくまで例である。自分の好みによって書き換えてほしい。

また、自分でコマンドの別名を登録するときにも、ここに続けて書いておけばいい。

12.8　シェルの変更

シェルが何種類かあることは説明したが、自分の基本にするシェルを変更することができる。シェルを変更するには、 chsh を使う。これは「**ch**ange **sh**ell – チェンジ・シェル」の略だ。

では、実際にやってみよう。

```
% chsh
```

すると、 Mule が自動的に起動して、次のようなファイルが開かれる。

```
#Changing NIS information for ei00000.
Shell: /bin/tcsh
Full Name:
Location:
Office Phone:
Home Phone:
```

この中の Shell: という項目を変更すると、シェルが変更される。ここに指定できるのは、次の 6 つのうちのどれか 1 つである。また、 Full Name: などに自分の名前を書いておくと、システムに名前が登録される[8]。

```
/bin/sh
/bin/csh
/bin/tcsh
/usr/local/bin/bash
/usr/local/bin/ksh
/usr/local/bin/zsh
```

変更したら、 Mule をセーブして抜ける。すると、次のようにパスワードを聞かれる。

```
Changing NIS information for ei99000 on uxsv1.edu3.aomori-u.ac.jp
Please enter password:
```

正しくパスワードを入力すると、次のメッセージが出て、シェルの変更は成功する。

```
chfn: NIS information changed on host uxsv1.edu3.aomori-u.ac.jp
```

ちなみにシェルによって、初期設定のファイルは異なるので注意すること。それぞれ csh と tcsh の初期設定ファイルは「.cshrc」、sh と ksh は「.profile」、bash は「.bashrc」、zsh は「.zshrc」となっている。

[8]他人からこの情報は調べられるので、Phone: などに自分の電話番号を入れることはすすめられない。

第13章　WWW と電子メール

もともと、WWW (World Wide Web) は UNIX の世界を中心に広まったものだ。この章では Netscape Communicator などを使って WWW と電子メールの使い方を紹介しよう。

13.1　Web ブラウザいろいろ

UNIX にはいろいろな Web ブラウザがある。はじめて画像を表示することができた Mosaic、Netscape の各種バージョン、テキスト・ベースの Lynx、試験的に作られた Arena、HTML エディタと Web ブラウザが融合した Amaya、最近脚光を浴びている Opera。

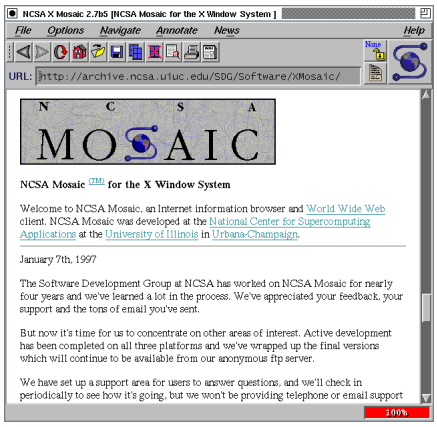


図 13.1: XMosaic

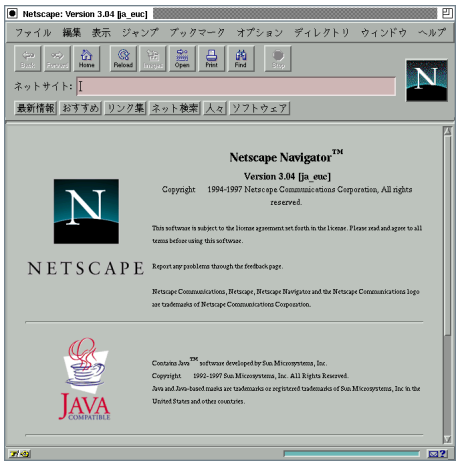


図 13.2: Netscape 3

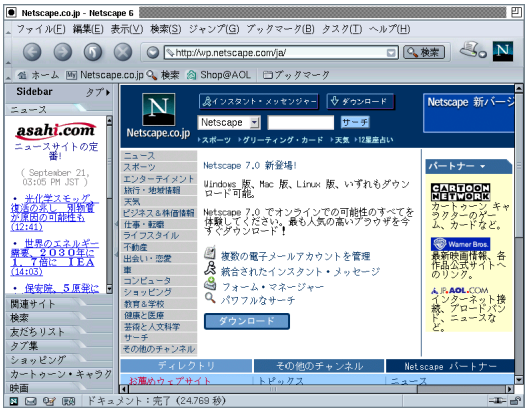


図 13.3: Netscape 6

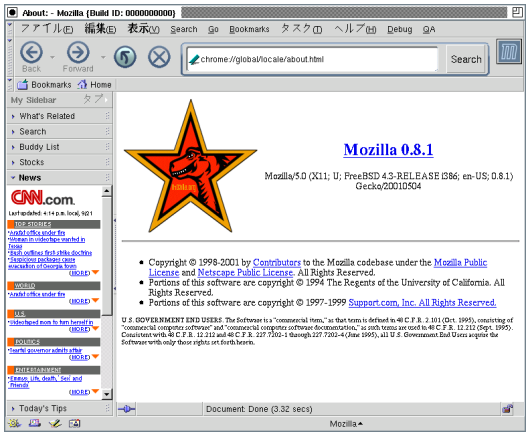


図 13.4: Mozilla

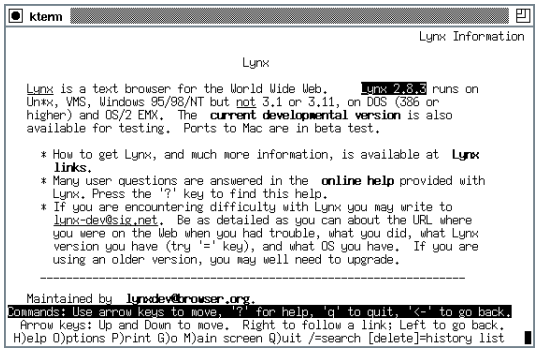


図 13.5: lynx

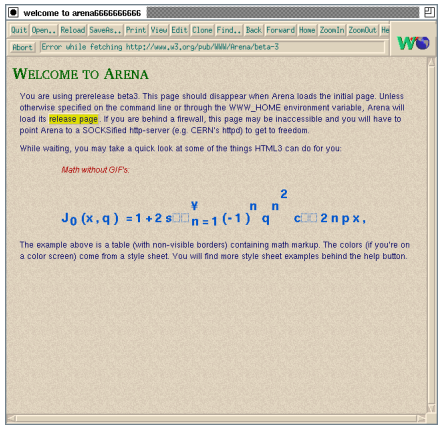


図 13.6: Arena

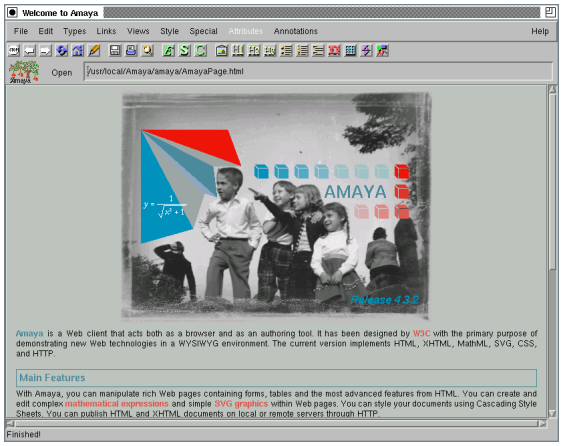


図 13.7: Amaya

図 13.8: Opera

今回は、これら Web ブラウザのうち Netscape Communicator (ver. 4 系)、それから w3m について紹介する。

13.2 Netscape Communicator の使い方

現在、 UNIX の世界で一番メジャーだと思われるブラウザが Netscape Communicator である。

Netscape は、 UNIX の場合、完全に個人で環境設定でき、ブックマークの使用も問題なくできる。

13.2.1 起動

まず、 Netscape Communicator を起動するには、`netscape` というコマンドを入力する。

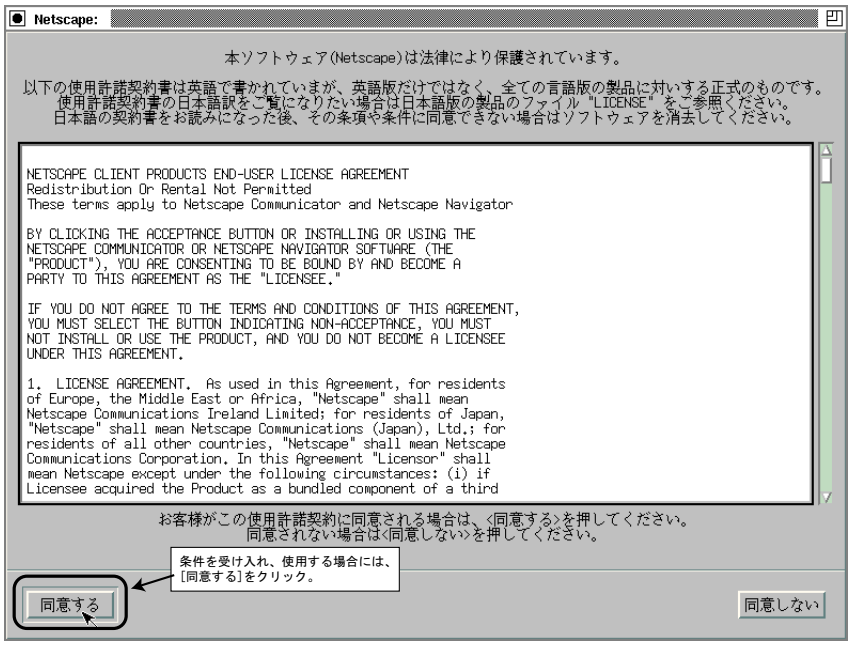
```
% netscape &
```

すると、ウィンドウが次々と開いて図 13.9 のような画面になる。

図 13.9: Netscape Communicator の起動直後

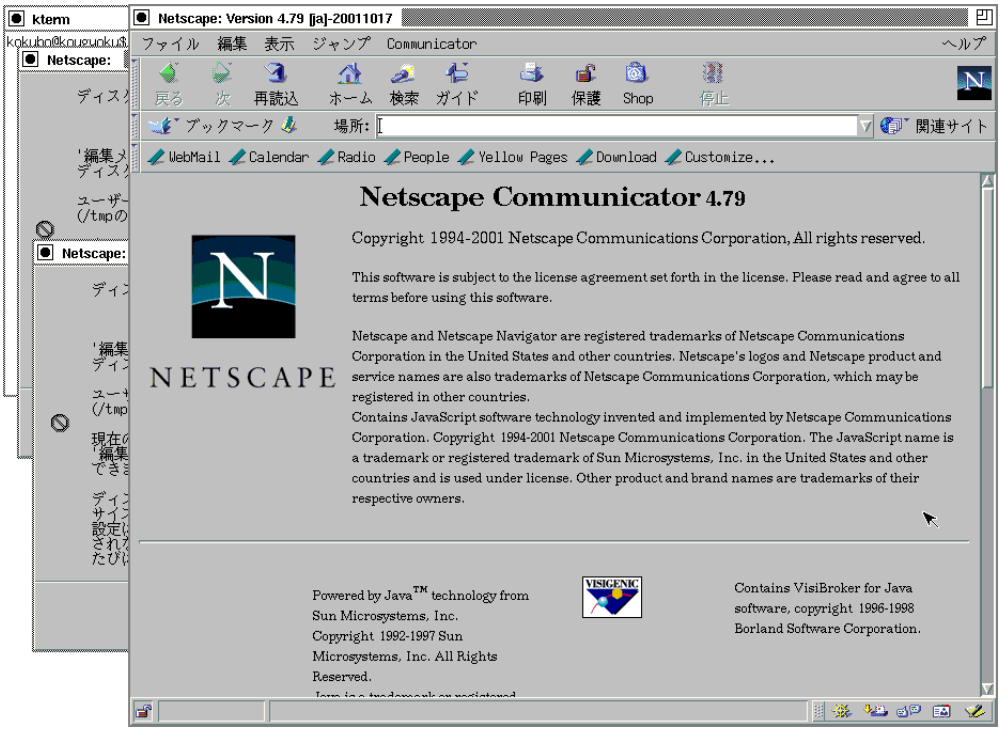
13.2.2 使用許諾条件

使用許諾条件が表示される。これを受け入れる場合にのみ使用できる。受け入れるならば、同意する をクリックする。



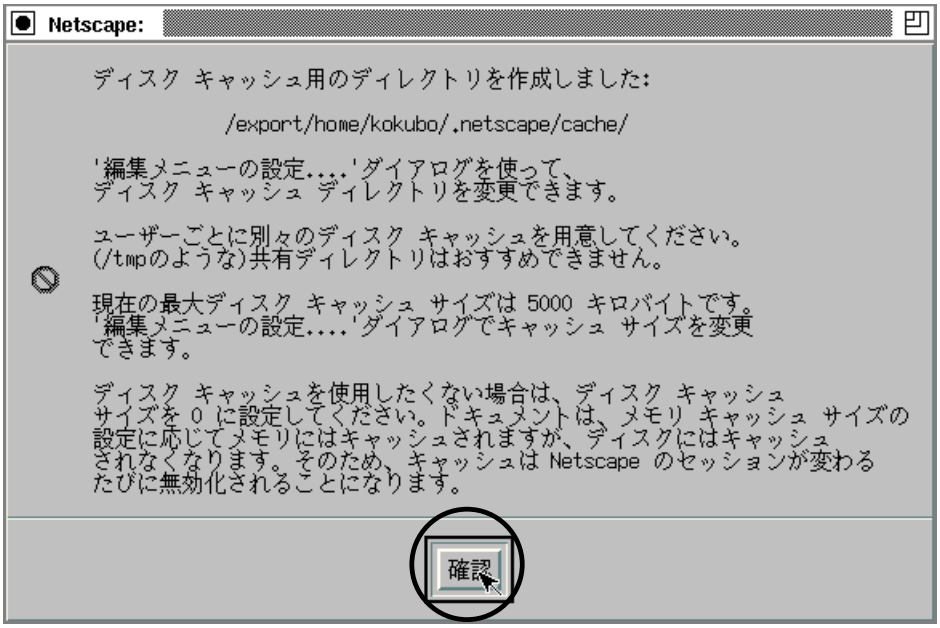
13.2.3 初回起動時メッセージ

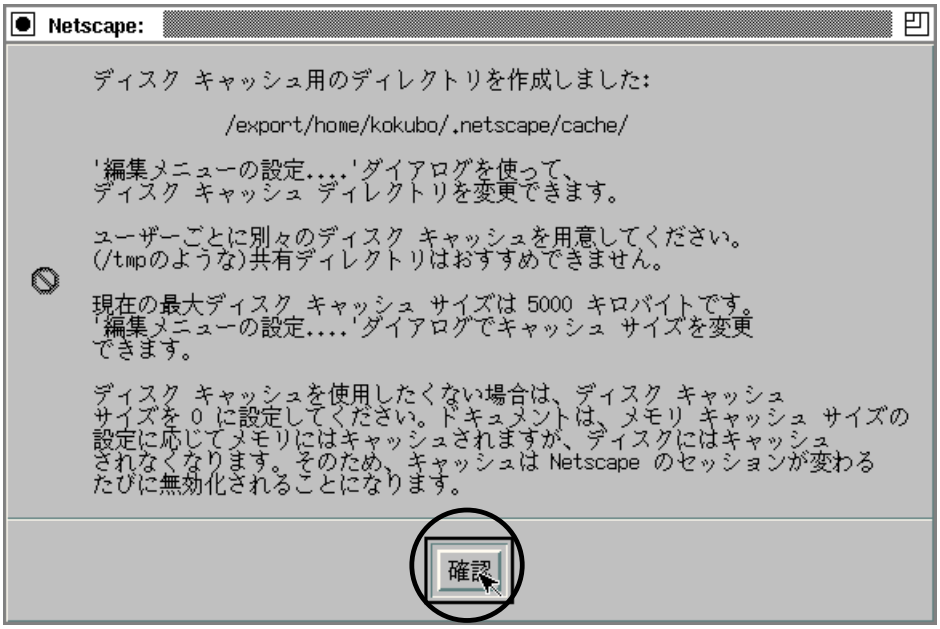
メッセージがいくつかでて、 Netscape が起動する。



しばらくすると、 Netscape 社のページに接続しようとして動きはじめる。しかし、設定が済んでいないうちは接続できないので、「停止」ボタンを押して止める。

キャッシュを作成した等のメッセージは 確認 をクリックして閉じる。





13.2.4 設定

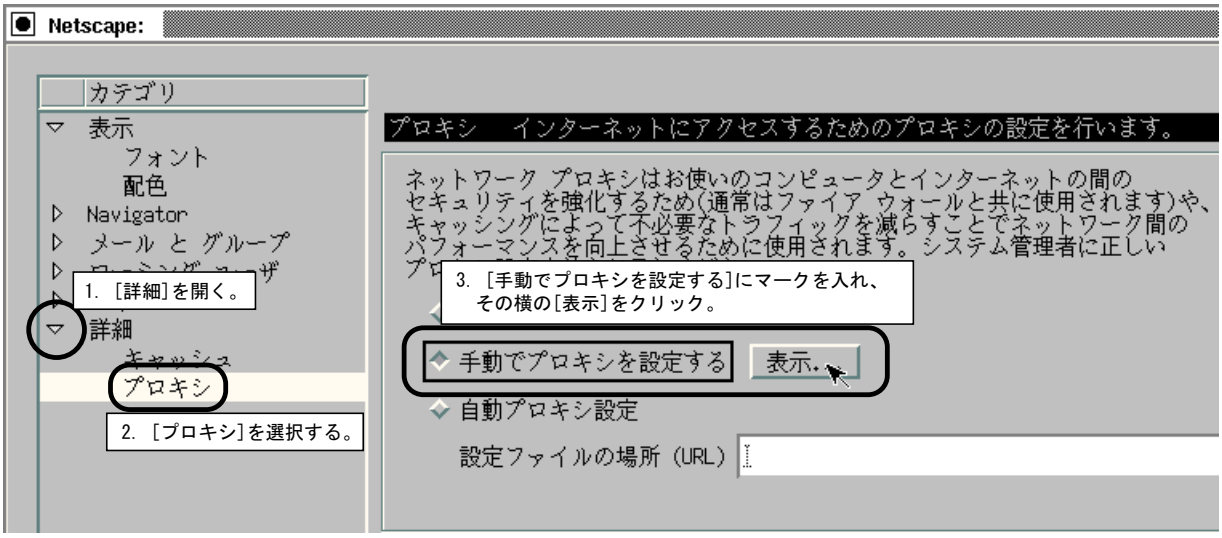
Netscape の設定を行う。まず、メニュー・バーの「編集」から「設定」を選ぶ。

プロキシの設定 (必須)

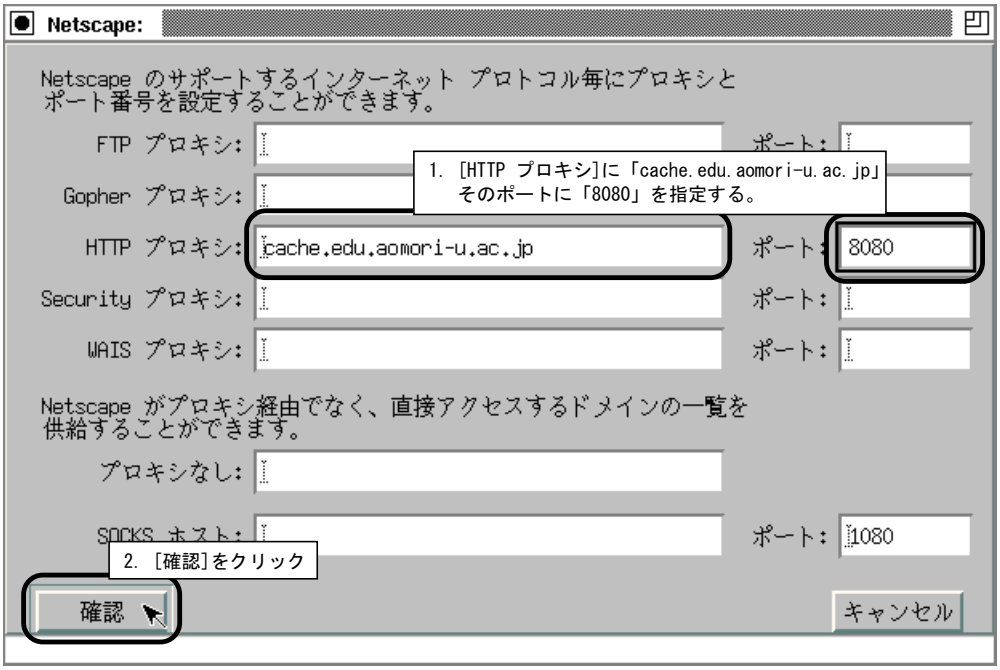
最初にプロキシの設定を行う。まず、画面左側に出ている「カテゴリ」の中の「詳細」を開く。これを開くには、左側にある「△」のマークをクリックすればよい。

次に現われたサブ・メニューの中から「プロキシ」を選択する。

画面右側に移って「手動でプロキシを設定する」の左側のチェック・ボックスをクリックしてチェックする。その後、すぐ右側にある「表示」をクリックする。

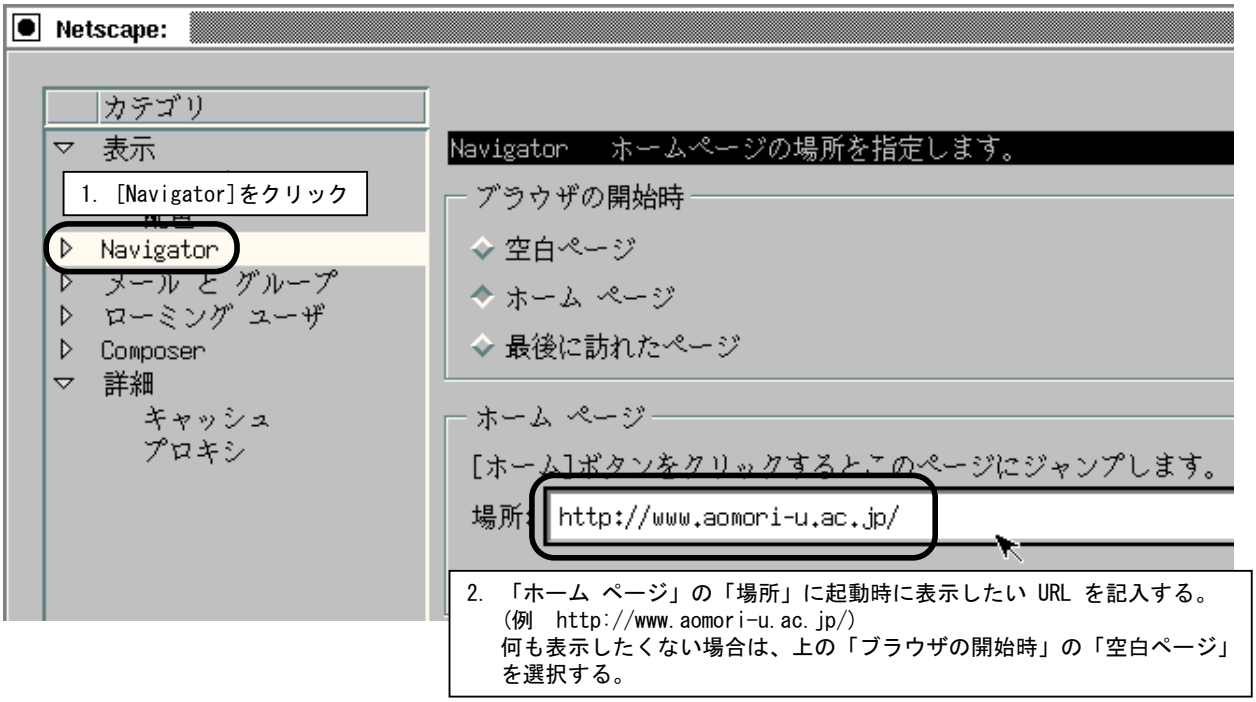


現われたウィンドウの「HTTP プロキシ」に「cache.edu.aomori-u.ac.jp」を、そのポートに「8080」を指定する。その後で「確認」をクリック。



起動時に表示するページの設定

画面左側の [Navigator] メニューをクリックする。次に、画面右側の「ホーム ページ」の「場所」の部分に、起動時に表示したいページの URL を指定する。たとえば「http://www.aomori-u.ac.jp/」など。もしも、起動時に空白ページを表示したい場合には、「ブラウザの開始時」の「空白ページ」を選択しておく。

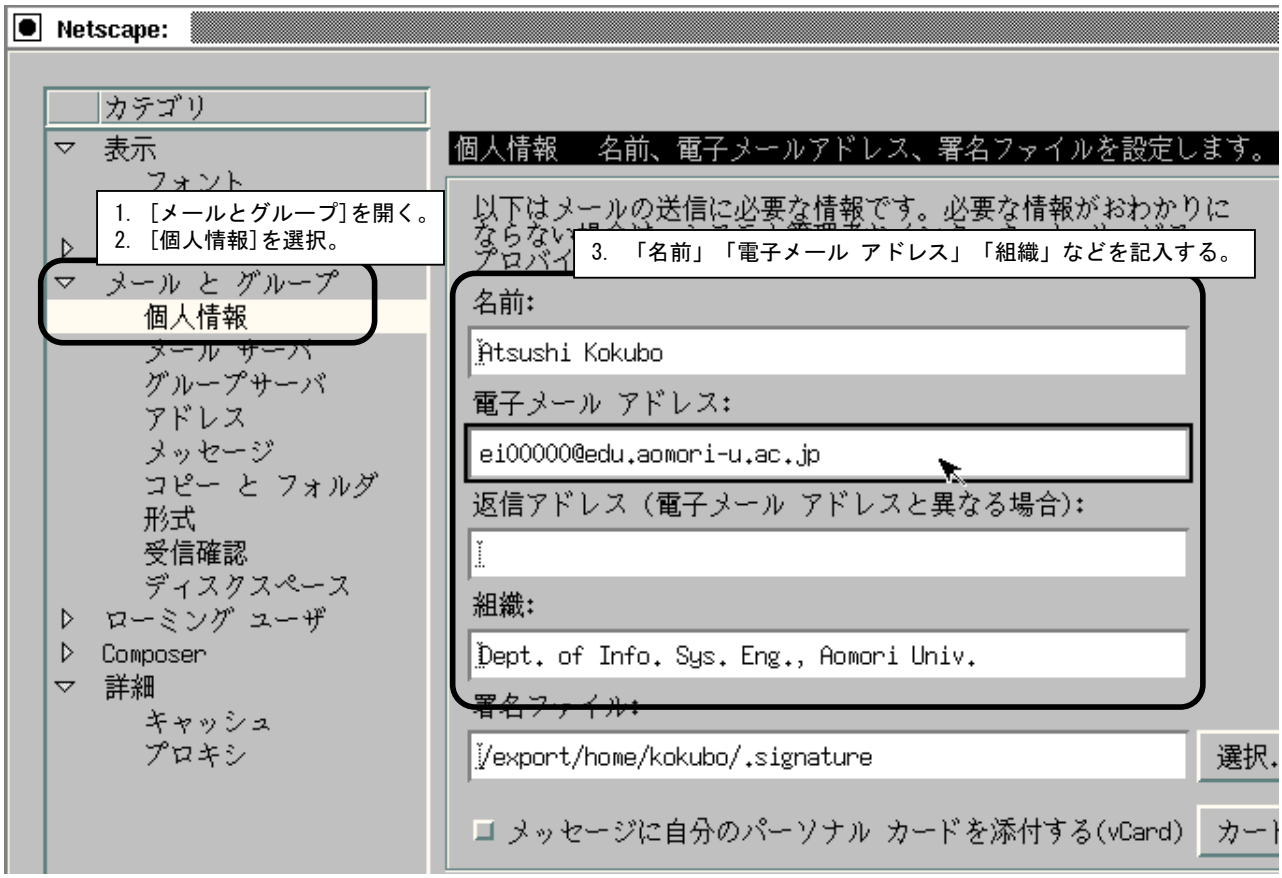


メールの設定

個人情報

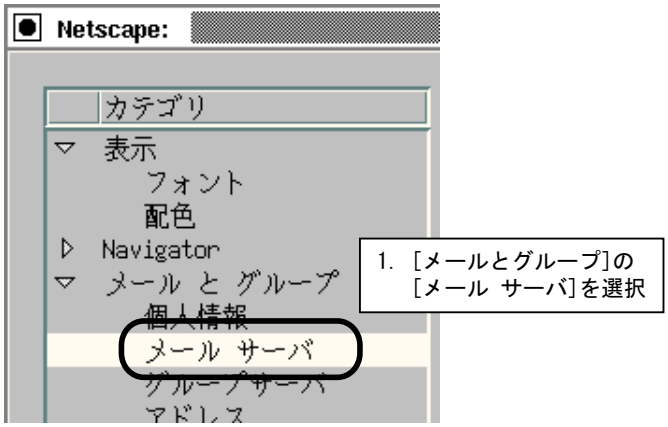
画面左側の [メール と ニュース] の左の△をクリックして開き、その中の [個人情報] を選択する。

画面右側に現われたダイアログに「名前」をローマ字などで、「電子メール アドレス」には「ei0X0XX@edu.aomori-u.ac.jp」など自分のものを、「組織」にはたとえば「Dept. Info. Sys. Eng., Aomori Univ.(青森大学情報システム工学科)」などを入力する。

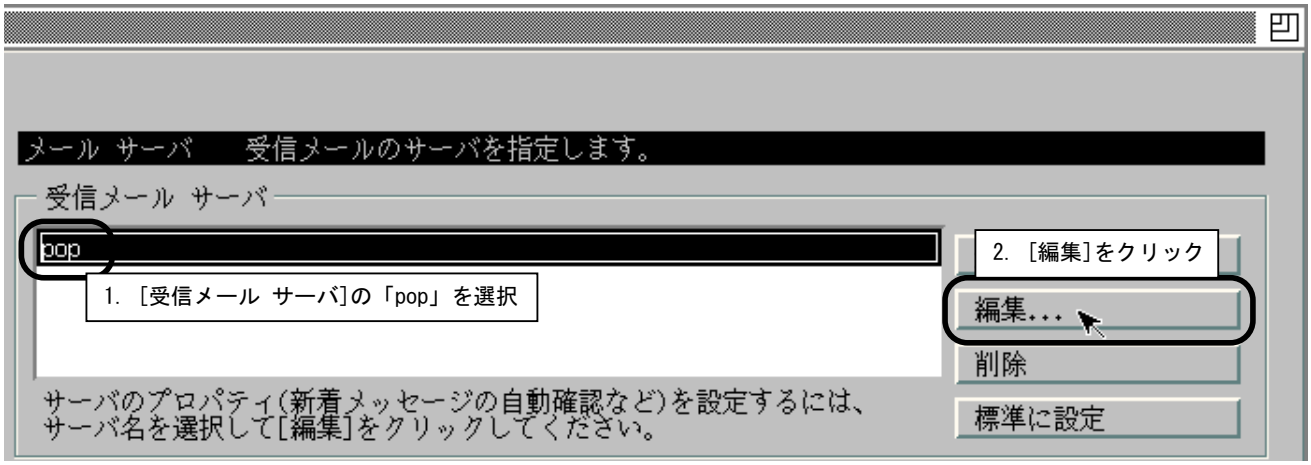


メール サーバ

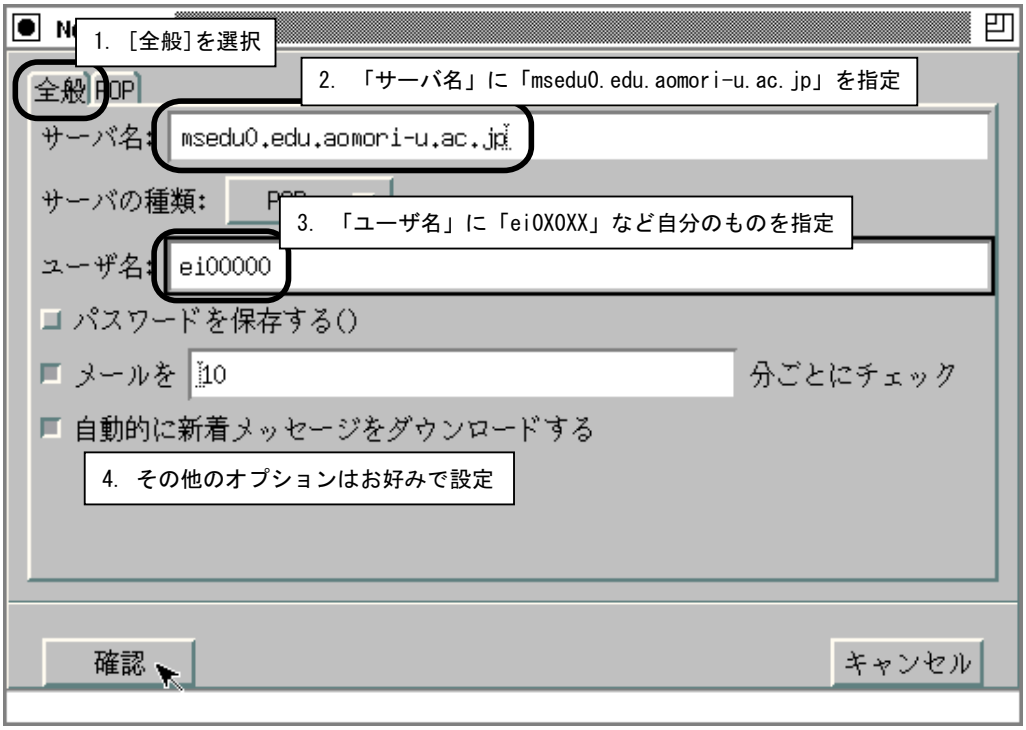
画面左側の [メール と グループ] の [メール サーバ] を選択する。



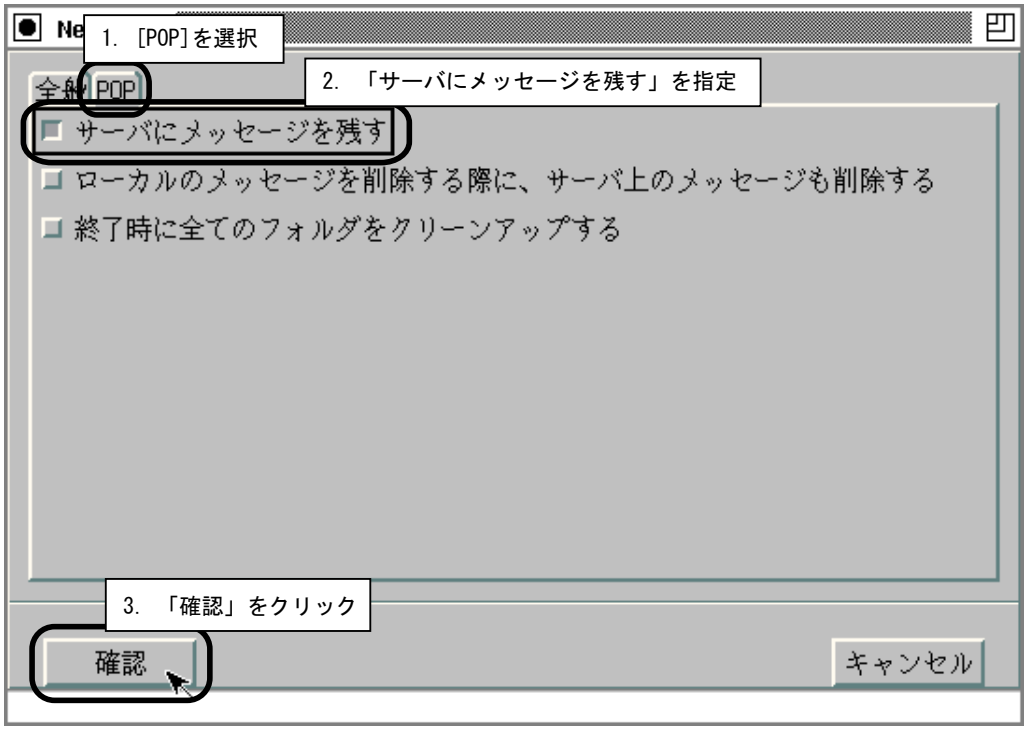
画面右側の「受信メール サーバ」から「pop」を選択し、「編集」をクリックする。



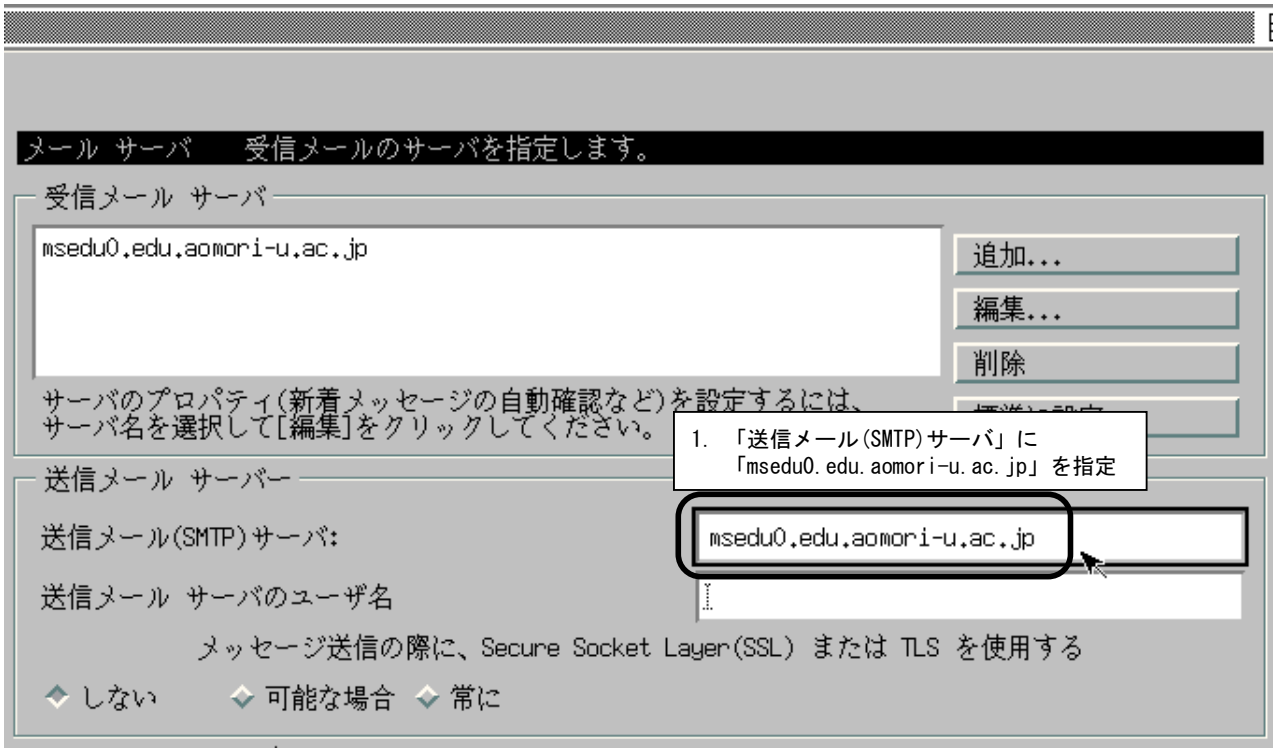
現われたダイアログの「全般」タブを選択する。「サーバ名」に「msedu0.edu.aomori-u.ac.jp」を指定する。「ユーザ名」に「ei0X0XX」など、自分のものを指定する。ほかのオプションも好みで設定していいだろう。



今後、基本的には常に FreeBSD 側でメールを読もうという人以外[1]は、サーバにメールを残すように設定をしておこう。まず、「POP」タブを選択する。現われたダイアログの「サーバにメッセージを残す」にチェックを入れる。後は、「確認」をクリックする。



「送信メール サーバ」に「msedu0.edu.aomori-u.ac.jp」を指定する。

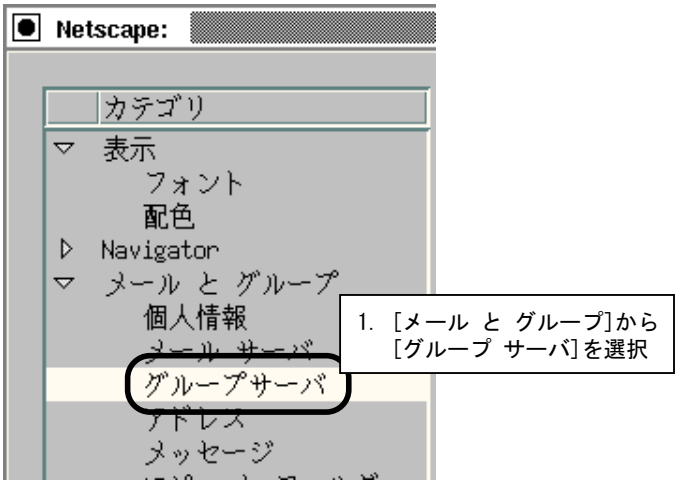


[1]たとえば、基本的には Windows でメールを読む。FreeBSD 側はサブで使うという人

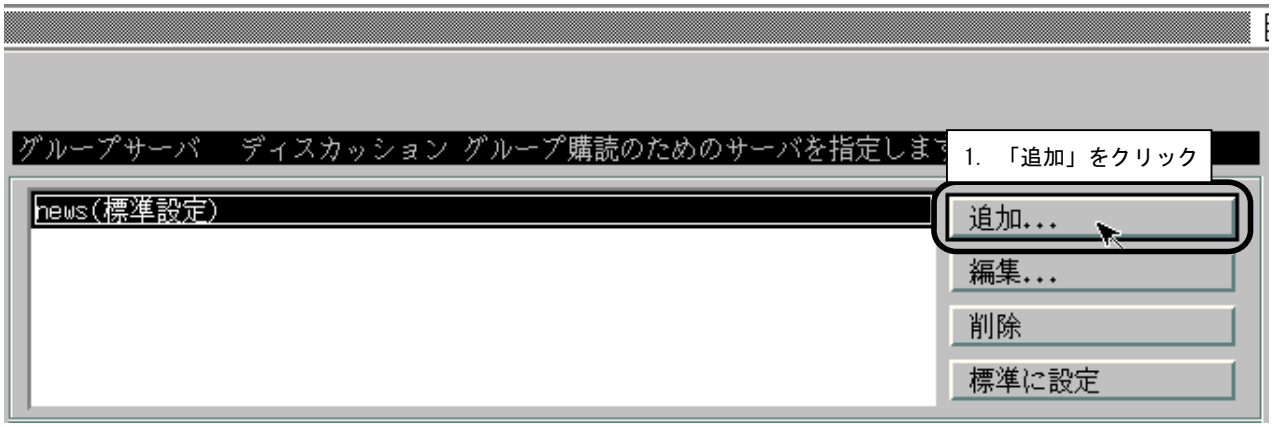
ネットニューズの設定

Netscape でネットニューズを読もうという人だけ以下の設定を行う。

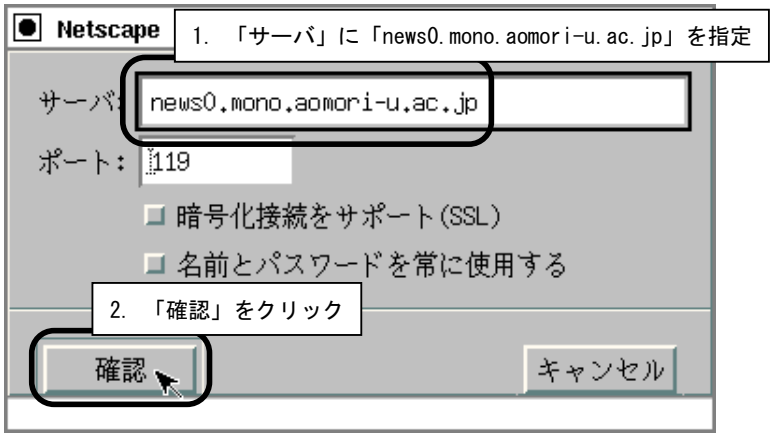
まず、画面左側の [メール と グループ] の [グループ サーバ] を選択する。



次に画面右側から「追加」をクリックする。



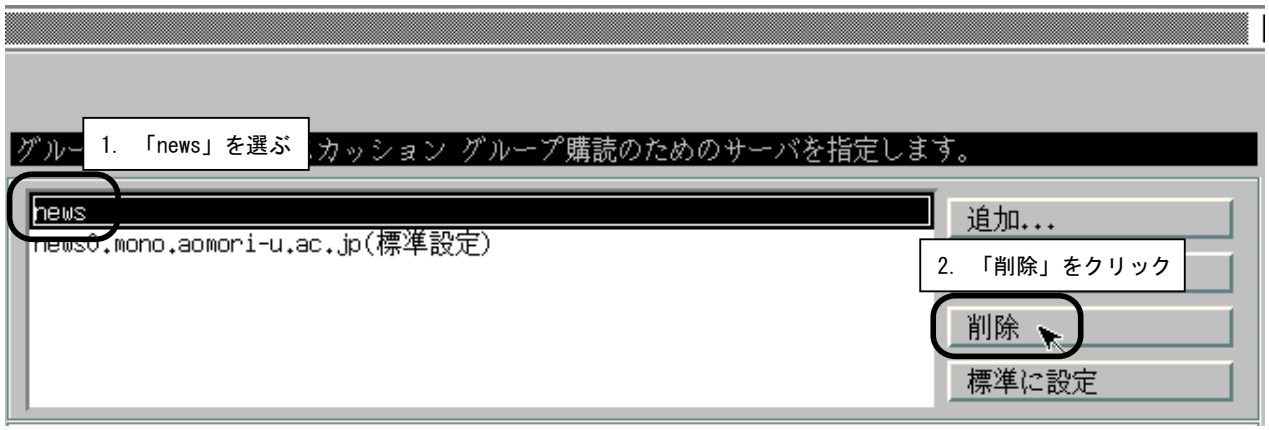
「サーバ」に「news0.mono.aomori-u.ac.jp」を指定し、「確認」をクリック。



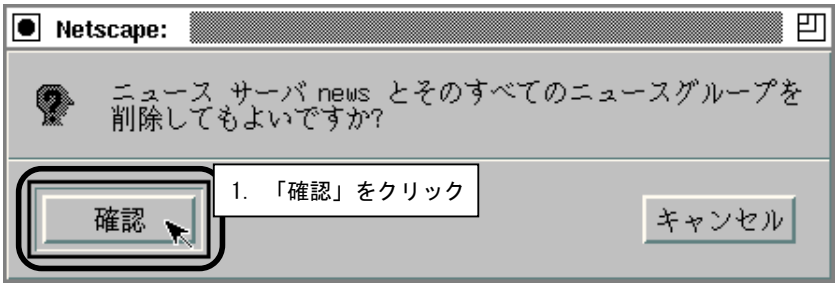
すると、新たに「news0.mono.aomori-u.ac.jp」が現われているのでこれを選択し、「標準に設定」をクリック。

「news0.mono.aomori-u.ac.jp」が「標準設定」になった。

次には、「news」を選択し、「削除」をクリックする。

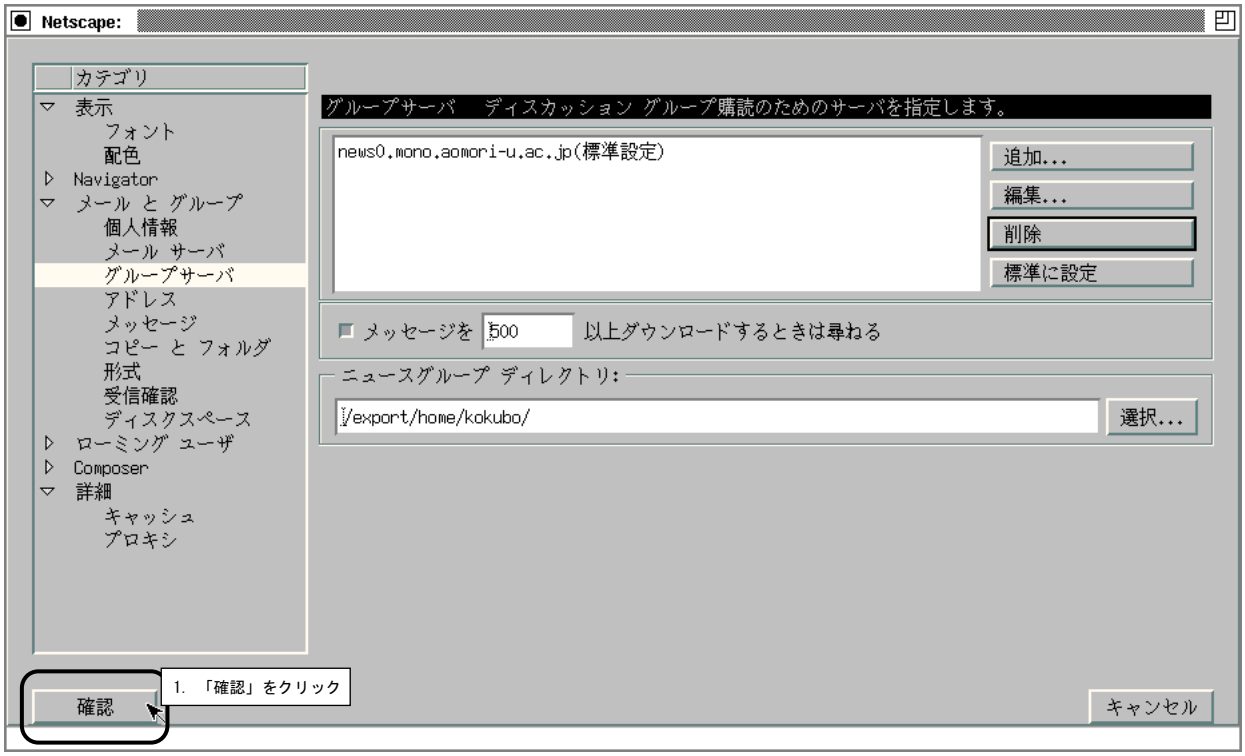


「ニュース サーバ news とそのすべてのニュースグループを削除してもよいですか？」と聞かれるが、「確認」をクリックして削除する。



設定終了

すべての設定が終了したら、画面左下の「確認」をクリックする。



13.2.5 ホーム ページの表示

「ホーム」をクリックすると、「ホーム ページ」に指定した URL が表示される。

13.2.6 Netscape の操作

Netscape の操作は、基本的にメニューのボタンと「場所」ダイアログの 2 つで行える。

まず、メニューの方は、以下のようになっていて、非常に直感的に構成されている。ほとんど説明する必要はないと思われるので、割愛する。

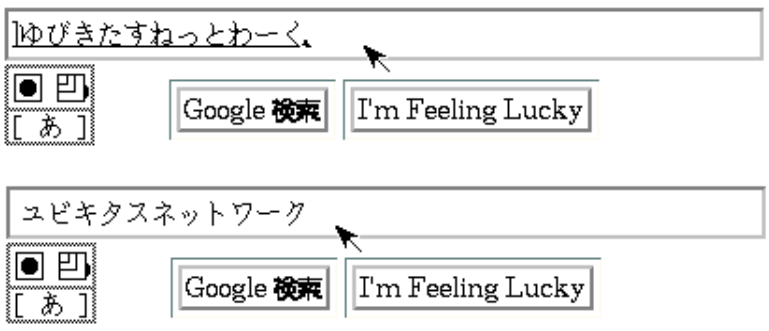
それから、「場所」ダイアログ・ボックスには、直接 URL を入力することができ、入力して Enter することでページを表示させられる。

13.2.7 Netscape で日本語入力

たとえば、検索エンジンなどのページで、ダイアログ・ボックスに日本語を入力し、検索を実行したくなることがある。Netscape では[2]、XIM プロトコルを通じて `kinput2` というソフトを使って日本語入力を使用することができる。

`kinput2` による日本語入力の on / off は、Shift+スペース である。日本語変換には Mule と同様に Canna が使われている。on / off が、この場合は Shift+スペース、Mule の場合は「`c-o`」ということ以外はほとんど同じ操作である。

日本語変換操作はほぼ Mule で Canna を使用しているときと同様



[2]正確には Netscape に限らず、X Windows System のアプリケーション一般も同様

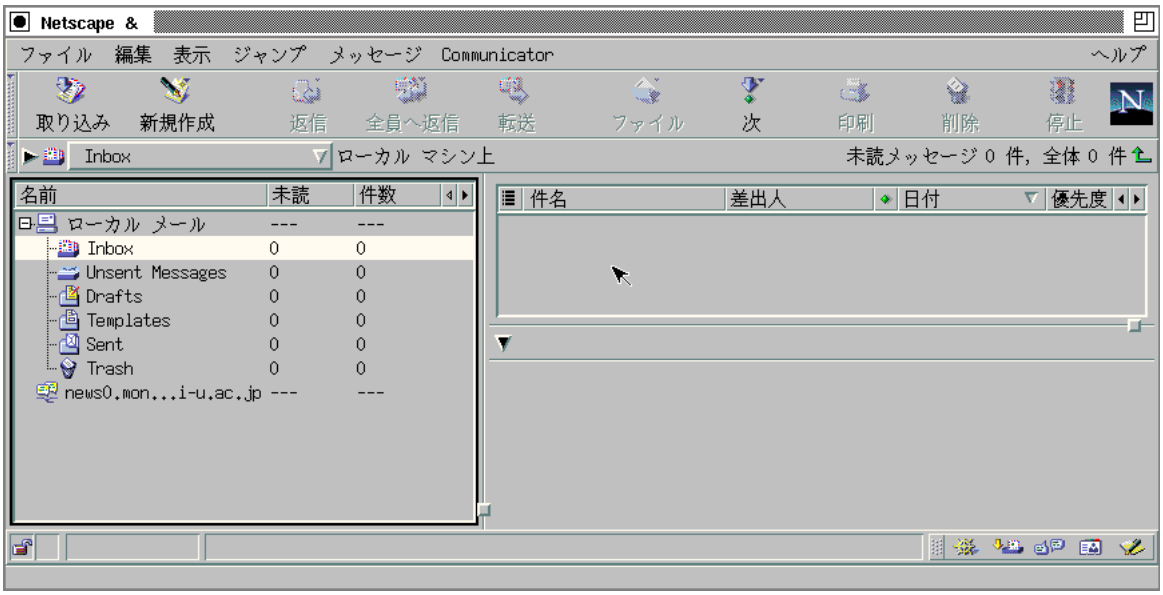
13.2.8 Netscape でメールを使う

Netscape Messenger の起動

Netscape でメールを使うには、メニューバーの「Communicator」から「Messenger メールボックス」を選択する。

また、「`netscape &`」の代わりに「`netscape -mail &`」として起動すると、最初からメール・モードで立ち上がる。

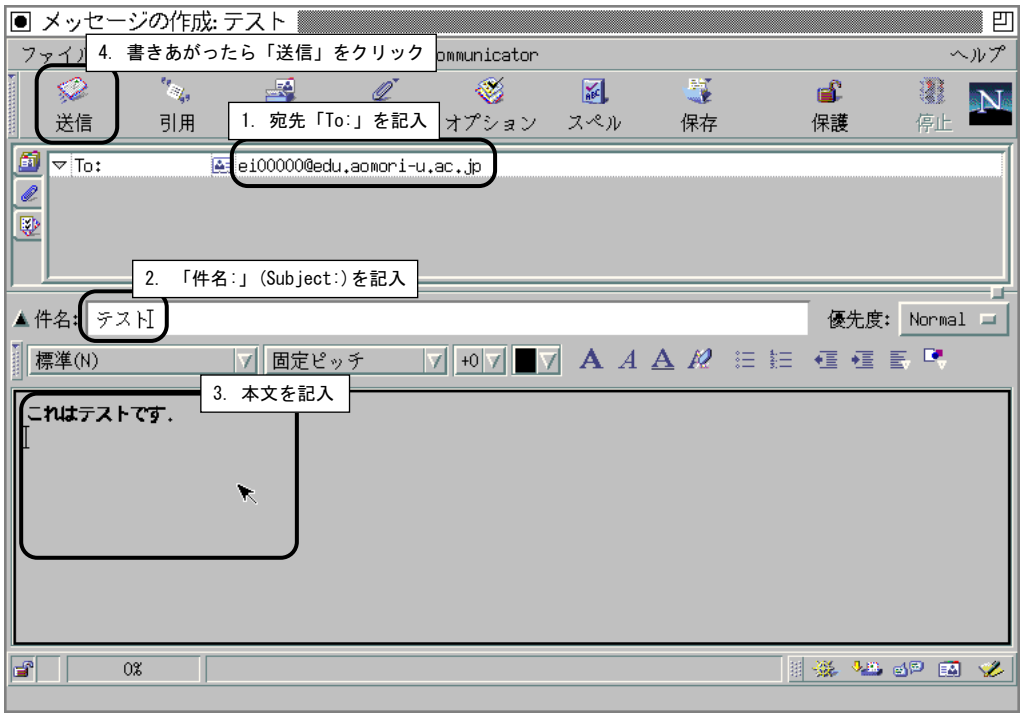
Netscape Messenger が起動する。



メールを書く

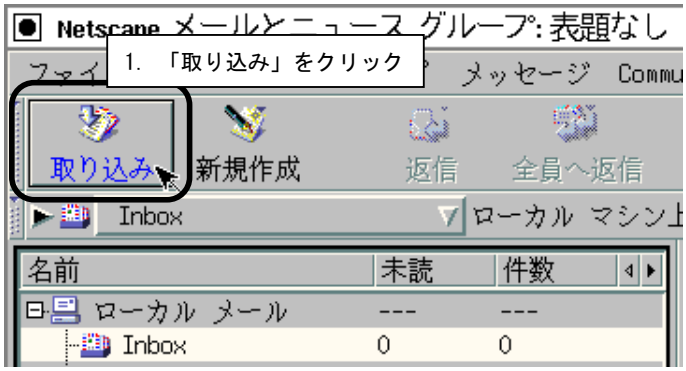
メールを書くには「新規作成」をクリックする。

「メッセージの作成」ウィンドウが開く。宛先「To:」にメール・アドレスを、「件名:」(Subject:)に題名を、そして本文を記入する。メールが書きあがったら「送信」ボタンをクリックすると送信される。なお、日本語入力は、 kinput2 によるもので、Shift+スペース で on / off である。

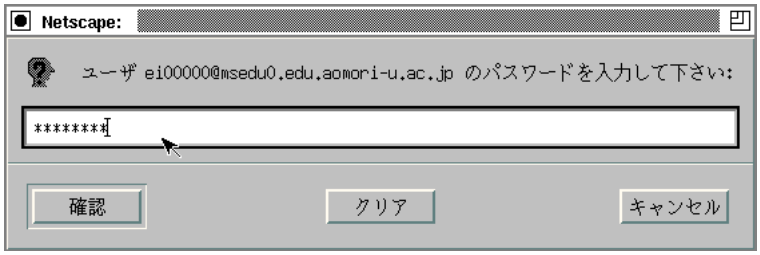


メールを読む

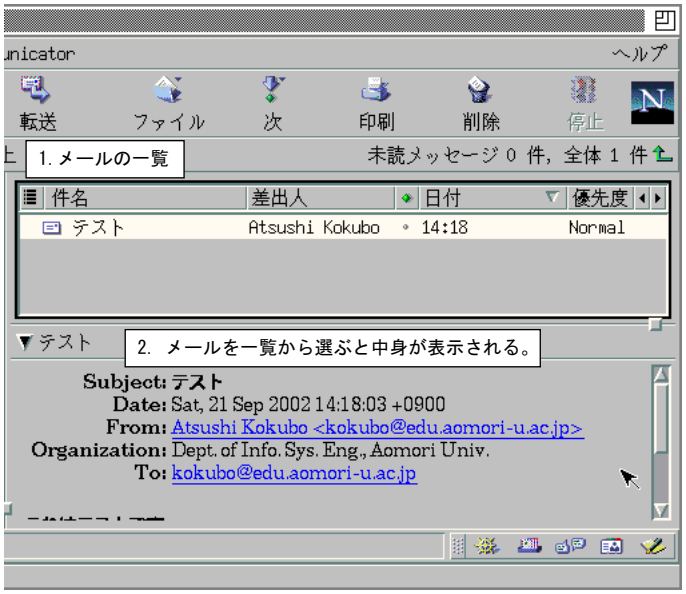
メールを読むには「取り込み」をクリックする。



すると、メールのパスワードを聞かれるので、ダイアログ・ボックスに入力して、「確認」をクリックする。



メールが届いていると、メールをサーバから読み出し、一覧が画面の右側に表示される。この一覧からメールを選ぶと、すぐ下の部分に表示される。



13.3 テキスト・ブラウザ w3m

w3m は伊藤彰則さんの作られたテキスト・ブラウザだ。このブラウザは、最近のバージョンは画像も表示可能だが、元々は基本的に文字だけで Web ページを表示するものだった。文字の表示だけだと、非常に高速に Web ページを表示することが可能になり、ストレスなく Web ブラウジングが可能になる。

起動するには、次のようにする。

```
% w3m -v
```

すると、次のような画面になって、 w3m が起動する。使い方は、基本的に `jless` などのページャーと同様である。

```
                        Welcome to w3m!

                This is w3m version w3m/0.2.3.2+cvs-1.196-img-1.17
                        Written by Akinori Ito




≪↑ ↓ Viewing <W3M startup page>
```

Web ページを見るには次のようにする。まず、「`U`」と打つ。すると、画面一番下に URL を入力できるようになる。

```
Goto URL:
```

ここにたとえば、 `http://www.aomori-u.ac.jp/` などの URL を指定して Enter する。

```
Goto URL: http://www.aomori-u.ac.jp/
```

すると、下の図のように、画像まで読み込んでページを表示してくれる。

図 13.10: w3m の画面

カーソルの移動やコマンドは、次のようになっている。

表 13.1: カーソルの移動

コマンド	動作
↑	ひとつ前の行へ
↓	ひとつ次の行へ
→	一文字前に
←	一文字後ろに
TAB	次のリンクの位置へ
スペース	下の画面へ
b	上の画面へ
<	左の画面へ
>	右の画面へ
J	一行上にスクロール
K	一行上にスクロール
/「文字」Enter	前方サーチ
?「文字」Enter	後方サーチ

表 13.2: さまざまなコマンド

コマンド	動作
q	終了
リンクの上で Enter	リンク先へ飛ぶ
B	前のページにもどる
U	URL を指定する
v	ページのソースを表示 (表示を戻すには B)
M-b	ブックマークの表示
M-a	ブックマークへの追加
o	オプションの設定
H	ヘルプの表示

w3m では、画像がそのまま表示されるが、これはむしろ表示しない方がブラウザの動作はスムーズだ。もしも、表示しないように設定したければ、「o」と入力してオプションを表示してみよう。

```
                    Option Setting Panel
            (w3m version w3m/0.2.3.2+cvs-1.196-img-1.17)

外部ビューアの編集

表示関係

タブ幅                                  [8                    ]
文字幅 (4.0...32.0)                     [7                    ]
一行の高さ (4.0...64.0)                 [14                   ]
表示用漢字コード                        [EUC-JP                                ]
文書の文字コード                        [auto detect      ]
システムの文字コード                    [EUC-JP   ]
フレームの自動表示                      ( )ON  (*)OFF
target が未指定の場合に_self を使用する  ( )ON  (*)OFF
リンク先の自動表示                      ( )ON  (*)OFF
ディレクトリリストに外部コマンドを使う  (*)ON  ( )OFF
ディレクトリリスト用コマンド            [file:///$LIB/dirlist]
ファイル名のマルチカラム表示            ( )ON  (*)OFF
エンティティを ASCII の代替表現で表す   ( )ON  (*)OFF
TEXTAREA の行を折り返して表示           ( )ON  (*)OFF
空の IMG ALT 属性の時にリンク名を表示する (*)ON  ( )OFF
背景画像等へのリンクを作る              ( )ON  (*)OFF
≪↑ ↓ Viewing <Option Setting Panel>
```

この画面を下にスクロールすると、「インライン画像を表示」というオプションがある。「OFF」の前の「( )」で Enter を入力し、チェックを入れる。

```
インライン画像を表示                    ( )ON  (*)OFF
```

そして「[OK]」の上で Enter する。今度は、さきほど見ていたページが文字データとして表示される。

```
青森大学・青森短期大学
[t]                          * カット
    *                          大学院
    What's New                 経営学部
    *                          * 経営学科／産業デザイン学科
    メッセージ                 社会学部
    *                          * 社会学科／社会福祉学科
    入試情報                   工学部
    *                          * 電子システム工学科／情報システム工学科／生物工学
    大学案内                   青森短期大学
    *                          * 商経科第一部／商経科第二部
    キャンパスニュース
    *                          トピックス
    サイトマップ               *
    *                          ◆ 青森大学工学部創立 10 周年記念講演会 (第 11 回青森大
    リンク集                     月 12 日 (土)13:30 より、青森大学メモリアルホールで
                                 場は無料です。
                               ◆ 出張講義のページをリニューアルしました。随時受付
                                 ご利用ください。
                               ◆ 青森大学・青森短期大学オープンキャンパスへ参加し
                                 ３回オープンキャンパスは９月 21 日 (土) です。
                               ◆ 本学図書館新館 2 階展示資料室にて「没後 90 年石川啄
                                 催中です (〜9/20 迄)。
≪↑ ↓ Viewing <青森大学・青森短期大学>
```

この状態だと、いろいろなページを高速に表示することができて非常に便利である。このように、絵を見たい場合、文字を高速に表示したい場合、それぞれに合わせて適切なブラウザを選択しよう。

この章で紹介したコマンド

WWW 関連コマンド他

`netscape` : Netscape Communicator

使い方: `netscape &` Netscape Communicator 起動

`-mail` オプションをつけるとメール・モードで起動

`w3m` : テキスト・ベース Web ブラウザ

使い方: `w3m -v` w3m 起動

第14章　ネットワーク機能

UNIX システムのネットワーク機能

これまで、 WWW へのアクセス方法や電子メールなど、いくつかのネットワーク関係のアプリケーションを紹介してきた。今回は、UNIX システムのその他のネットワークに関係した機能を紹介しよう。

14.1　ユーザ情報の表示

UNIX システムでは、一台のマシンに他のマシンから、同時に何人も login して使うことができる。そのことに関係したコマンドを紹介しよう。

14.1.1　誰が何をしているのか?: w

現在、誰が login しているのかを調べるコマンドはいくつかある。たとえば、以前紹介した who コマンドもその一つである。

今回は w を使ってみよう。w には「**w**ho and **w**hat」という意味がこめられていて、誰が何をしているのかが表示される。では「w」と入力してみよう。

```
% w
11:51AM  up 3 mins, 3 users, load averages: 0.01, 0.01, 0.00
USER             TTY      FROM              LOGIN@  IDLE WHAT
ei00000          p0       :0.0             11:50AM     - tcsh
ei00000          p1       :0.0             11:48AM     - w
ei00000          p2       :0.0             11:50AM     - tcsh
%
```

この表示からわかることは、 ei00000 さんが 3 つ端末を開いて login していること。その端末番号は「p0」〜「p2」であること。また、いずれもマシンのコンソール (:0.0) で端末をひらいていること。そして現在開いている端末を開いたのは 11:48AM 以降であること。また現在 w コマンドを実行しているということなどである。なお、 IDLE というのは、最後にキーを叩いてから何秒たっているかを表す。

who を実行して比較してみると、w の方がずっと詳しいことがわかるだろう。

```
% who
ei00000          ttyp0     9/23 11:50   (:0.0)
ei00000          ttyp1     9/23 11:48   (:0.0)
ei00000          ttyp2     9/23 11:50   (:0.0)
%
```

14.1.2 より詳しいユーザ情報を調べる: finger

ここまでのところは、単に今 login している人の情報を表示するだけだったが、finger を使うと、それ以外のユーザの情報も表示することができる。まずは使って見よう。

```
% finger
Login            Name                TTY  Idle  Login Time   Office  Phone
ei00000                              p0         木     11:50
ei00000                              p1         木     11:48
ei00000                              p2         木     11:50
%
```

これでは、特に w とあまり変わらないような気がする。

しかし、 finger は、今 login していないユーザに関しても調べることができる。「finger 「ユーザ ID」」で現在そのマシンに login していない適当なユーザを指定してみよう。

```
% finger ei00000
Login: ei00000                          Name:
Directory: /home/ei00/ei00000           Shell: /bin/tcsh
Last login Thu Nov 15 03:18 (JST) on ttyp2 from :0.0
No Mail.
No Plan.
%
```

と、このように、名前 (登録されてい場合のみ)、ホーム・ディレクトリのパス、使っているシェル、最後に login した日時まで表示されてしまう。

ちなみに、現在ほどネットワーク犯罪が盛んでなかった頃は、他のマシンの情報も「finger 「ユーザ ID」@「ホスト名」」で調べることもできた[1]。

[1]更には「finger @「ホスト名」」で全ユーザの情報も調べられることもあった。あらかじめこのような方法でユーザ情報を集めてから、マシンにアタックをかけて侵入しようとする人が現われるようになり、これらのコマンドを実行することができなくなったのである

14.2 ホスト名と IP アドレス

ネットワークに接続されたマシンは、「ホスト名」と「IP アドレス」によって識別されている。これらを調べるコマンドを紹介しよう。

14.2.1 ホスト名: hostname

現在使っているマシンのホスト名は hostname コマンドで表示できる。

```
% hostname
pcux00.edu3.aomori-u.ac.jp
%
```

B 演習室には、コンピュータにラベルが貼ってあるが、この番号を使って「pcux「番号」.edu3.aomori-u.ac.jp」というのが、「ホスト名」になっている。また、「ホスト名」の後ろの方の「edu3.aomori-u.ac.jp」の部分を「ドメイン名」という。

14.2.2 IP アドレスとホスト名の対応: nslookup

マシンには、ホスト名だけでなく IP アドレスというものも割り振られている。ホスト名と IP アドレスの対応は nslookup コマンドで調べることができる。

まず、ホスト名から IP アドレスを調べるには次のようにする。

```
% nslookup pcux01.edu3.aomori-u.ac.jp
Server:  msedu0.edu.aomori-u.ac.jp
Address:  172.31.66.9

Name:    pcux00.edu3.aomori-u.ac.jp
Address:  172.31.67.65
%
```

この最後の「172.31.67.65」というのが、ホスト名「pcux01.edu3.aomori-u.ac.jp」の IP アドレスである。なお、学外のマシンの IP アドレスに関しても同様に調べることができる。

次は IP アドレスからホスト名を調べてみよう。これも全く同様で、ホスト名の代わりに IP アドレスを打てばいい。

```
% nslookup 172.31.67.65
Server:  msedu0.edu.aomori-u.ac.jp
Address:  172.31.66.9

Name:    pcux00.edu3.aomori-u.ac.jp
Address:  172.31.67.65
%
```

14.3 他のマシンに入る: ssh

SSH (Secure SHell) を用いると、通信データを暗号化した状態で他のマシンにネットワークごしにログインすることができる。SSH の認証方式には何通りかある[2]が、ここではパスワード認証の方式を紹介しよう。

SSH で他のマシンに接続するには、次のようにする。

ssh 「ユーザ ID」@「ホスト名」

では、実際にやってみよう。たとえば ei00000 さんが pcux01.edu3.aomori-u.ac.jp というホストに接続しようとしたところが以下の図である。すると、初回にそのホストに接続するときだけ、『「RSA1 key fingerprint(RSA1 鍵の指紋)」が「9d:c4:27:e3:1d:13:15:f9:10:3a:93:a5:07:c7:0b:7d(これはホストによって異なる)」だが、これを受け入れてリストに追加してよいか?』と英語で聞かれるので、これに「yes」と答える。次にはパスワードを聞かれ、これを答えると login することができる。

```
% ssh ei000000@pcux01.edu3.aomori-u.ac.jp
The authenticity of host 'pcux01.edu3.aomori-u.ac.jp (172.31.67.65)' can't be e
stablished.
RSA1 key fingerprint is 9d:c4:27:e3:1d:13:15:f9:10:3a:93:a5:07:c7:0b:7d.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'pcux01.edu3.aomori-u.ac.jp' (RSA1) to the list of kn
own hosts.
ei00000@pcux01.edu3.aomori-u.ac.jp's password: パスワードを打つ (見えない)
Last login: Mon Sep 23 11:54:49 2002 from console
Copyright (c) 1980, 1983, 1986, 1988, 1990, 1991, 1993, 1994
        The Regents of the University of California.  All rights reserved.
FreeBSD 4.5-RELEASE (TRI_UX) #0: Tue Apr 23 19:49:03 JST 2002

⋮

[中略]
⋮


ei00000@pcux61%
```

[2]公開鍵認証、ホストベース認証、S/key 認証、パスワード認証、Kerberos 認証、Rhosts 認証など。一般的には公開鍵認証がいいと思われるが、 B 演習室のようにホームが NFS というしくみで共有されている場合は、必ずしもこの限りではなく、ケース・バイ・ケースである。

これで login することができた。次のようにウィンドウを開かないようなコマンドは普通に実行できる[3]。

```
% w
 5:58PM  up 6 mins, 2 users, load averages: 0.16, 0.05, 0.01
USER     TTY FROM              LOGIN@  IDLE WHAT
ei00000  p0  pcux51            5:58PM     - w
kokubo   p1  :0.0              5:53PM     4 -csh (tcsh)
%
```

```
% finger
Login    Name                 TTY  Idle  Login Time   Office     Office Phone
ei00000                      *p0         Wed    17:58
kokubo   A.Kokubo             p1      4  Wed    17:53
%
```

このマシンから抜けるには「exit」コマンドを打てばよい。

```
% exit
logout
Connection closed by foreign host.
%
```

なお、 SSH は Windows[4]や MacOS[5]などでも使用することができ、これらのマシンからもネットワークを使って UNIX システムに login することができる。

[3]「ssh -X 「ユーザ ID」@「ホスト名」」という具合に「-X」オプションをつけて SSH を起動すると、ウィンドウを開くような X Window System のアプリケーションも実行可能である。

[4]Cygwin、TTSSH、PortForwarder などの対応したアプリケーションをインストールする必要がある。

[5]MacOS X 以降では最初から付属している。それ以前の場合は MacSSH などが必要。

14.4　複数の人と会話: phone

phone というコマンドがあり、複数の人と会話することができる。phone の使い方は次の通りである。

phone「ユーザ ID」@「ホスト名」

なお、自分のホーム・ディレクトリに .phonerc というファイルを用意して、次のように書いておくと打った文字を自動的にひらがなに変換してくれる。

.phonerc
```
set code euc
```

では、実際に使ってみよう。今回は初めてなので、あらかじめ 2 人で打ち合せて、片方から phone をかけてみよう。

たとえば、次のような状況だとしよう。まず、電話をかける側が ei00000 さんで「pcux00.edu3.aomori-u.ac.jp」を使っている。そして、電話を受ける側が ei00001 さんで「pcux01.edu3.aomori-u.ac.jp」を使っている。

この場合、まず ei00000 さんが次のように入力する。

```
% phone ei00001@pcux01.edu3.aomori-u.ac.jp
```

すると、相手の画面に次のようなメッセージが現われる。

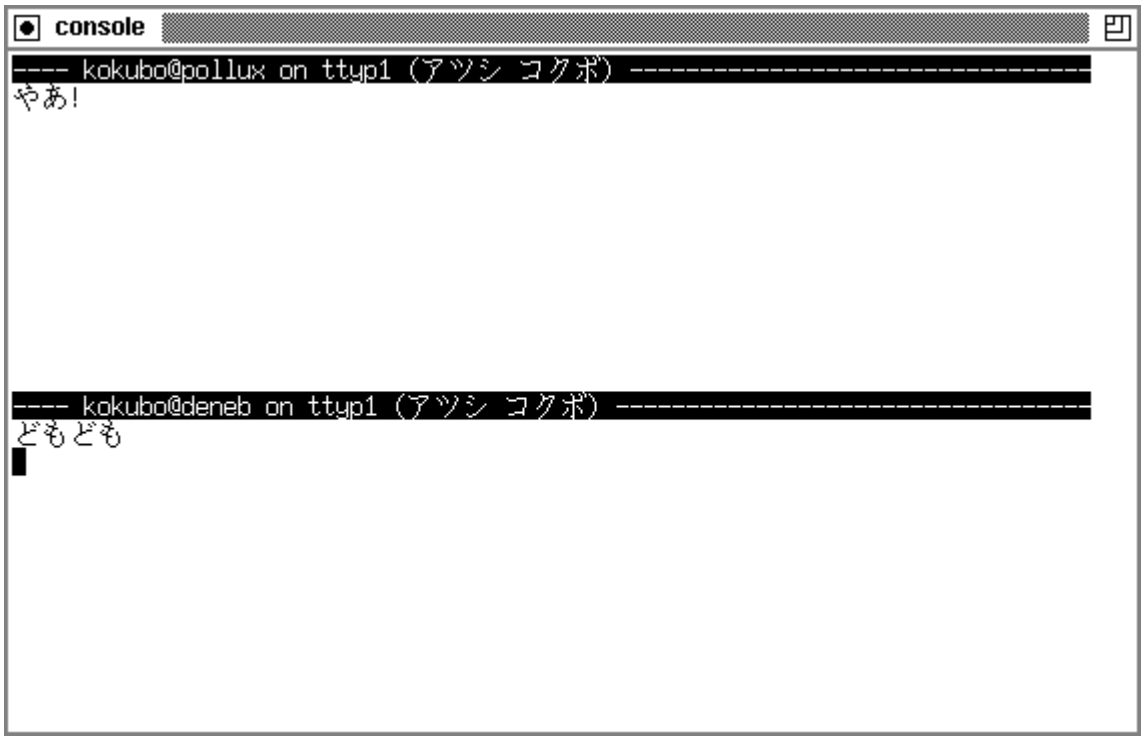
```
Message from the Telephone_Operator@pcux01.edu3.aomori-u.ac.jp at 11:13 ...
phone: connection requested by ei00000@pcux00.edu3.aomori-u.ac.jp
phone: respond with "phone ei00000@pcux00.edu3.aomori-u.ac.jp"
```

この英語は要するに「返事をするなら、「phone ei00000@pcux00.edu3.aomori-u.ac.jp」と入力しなさい」という意味である。

そこで今度は ei00001 の方が次のように入力する。

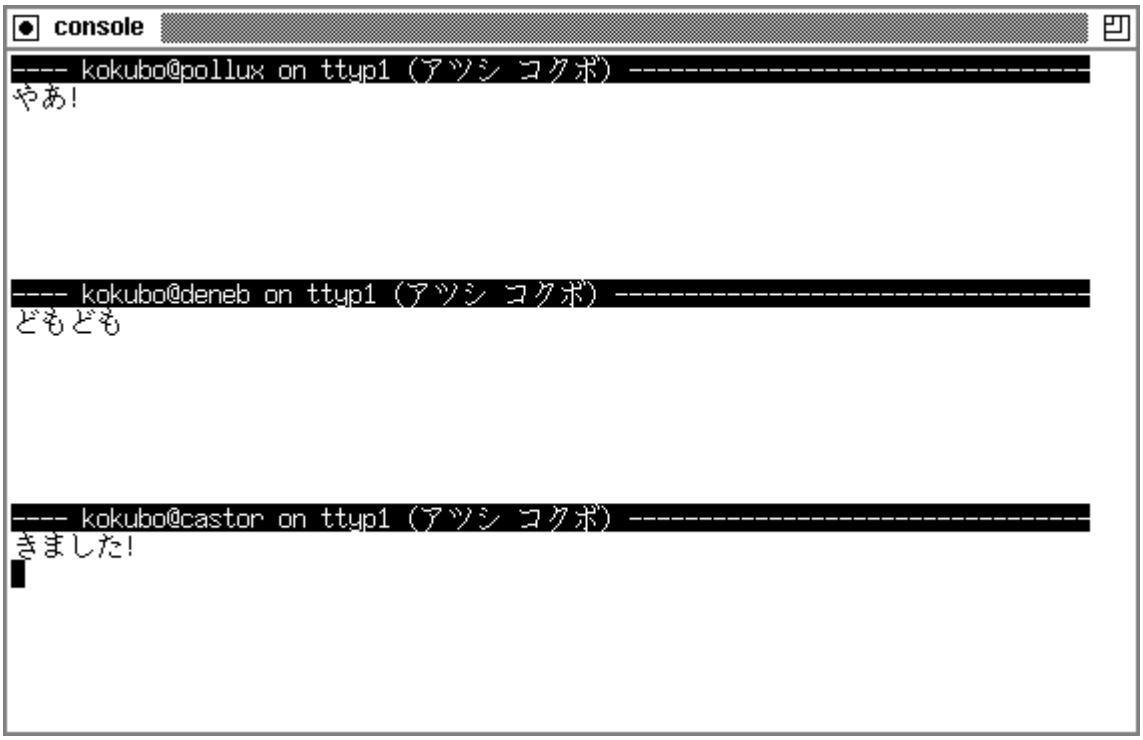
```
% phone ei00000@pcux00.edu3.aomori-u.ac.jp
```

すると次のような画面になり、打った文字が双方の画面に表示される。

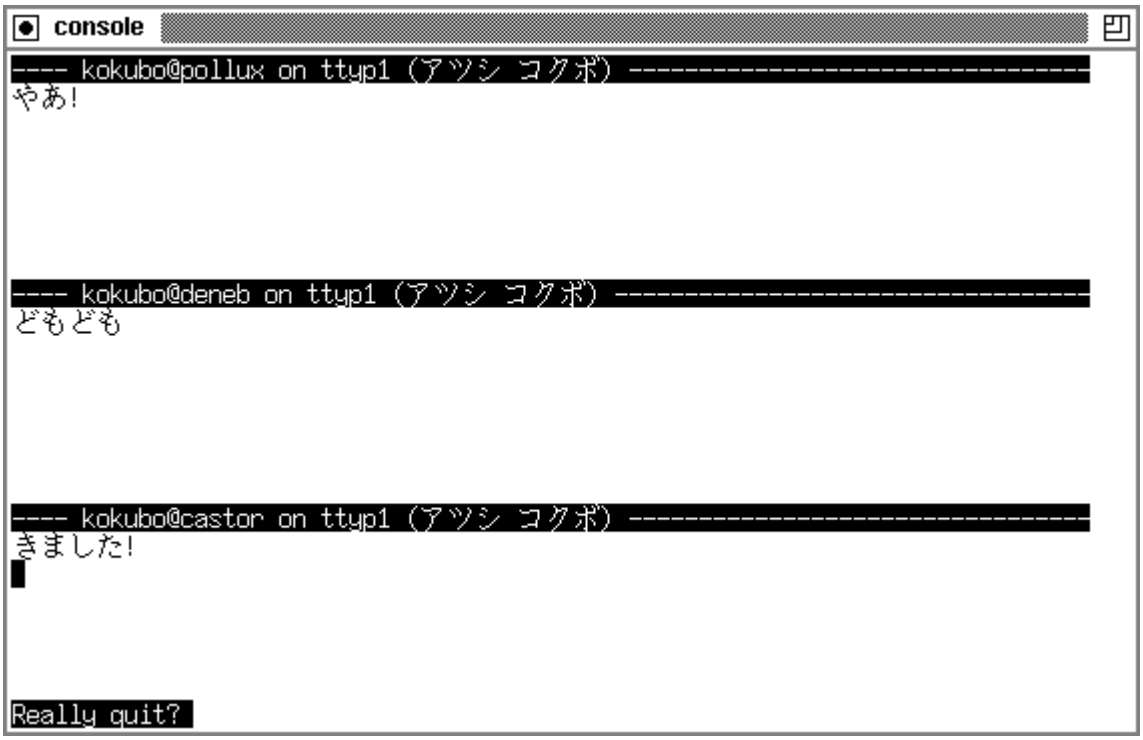


また 3 人目を呼ぶには、Esc を押して、コマンドモードにし、「call 「ユーザ ID」@「ホスト名」」とコマンドを打つ。

3 人目が同様に答えると、次のようになって、会話することができる。



終了するには「Ctrl+c」と入力する。すると、下のように本当に抜けるか? と英語で聞かれるので、「y」と入力して終了する。



この章で紹介したコマンド

ネットワーク関連コマンド

`w` : login している人と、現在実行中のコマンドの一覧

`who` : login している人の一覧

`finger` : ユーザ情報の表示

使い方: `finger` 「ユーザ ID」
または、 `finger` 「ユーザ ID」@「ホスト名」

`hostname` : ホスト名の表示

`nslookup` : ホスト名と IP アドレスの対照使い方: `nslookup` 「ホスト名」
あるいは「IP アドレス」

`ssh` : 他のマシンへのログイン

使い方: `ssh` 「ユーザ ID」@「ホスト名」

`phone` : 他のユーザと会話

使い方: `phone` 「ユーザ ID」@「ホスト名」

著者について

1968 年	2 月	埼玉県に生まれる
1990 年	3 月	東京理科大学 理学部 物理学科 卒業
1992 年	3 月	東北大学 大学院 理学研究科 原子核理学専攻 博士課程前期 2 年の課程 終了
1996 年	3 月	東北大学 大学院 理学研究科 原子核理学専攻 博士課程後期 3 年の課程 終了
1996 年	12 月	郵政省 認可法人 通信・放送機構 国内招へい研究者として、東北大学 加齢医学研究所に勤務
1997 年	4 月	青森大学 工学部 情報システム工学科 助手
現在		同上

UNIX にはじめて触れたのは、22 才のとき、仙台のソフトウェア開発会社 (株) コムテック (`http://www.comtec.co.jp/`) でのアルバイトのこと。そこには、 UNIX System V のマニュアルが一揃い完備していて、とても勉強になった。

しばらくは、 System V 系のシステムとの日々が続くが、後に SunOS で BSD 系のシステムにも触れることになる。SunOS では、 make 三昧の日々が続き、すっかり UNIX 系のシステムにはまることになる。現在では、 FreeBSD 以外に SPARCStation で OpenBSD を走らせる日々。

趣味は、本を読むことと、絵を描くこと、本を作ることなど。絵を描く時間が取れないのが、最近の悩み。なお、この本は ASCII の日本語 pLaTeX を FreeBSD や Cygwin の上で使用して作成した。ちなみにある種の本を作るときも、全く同様の環境を使っているらしい。

はじめての BSD rev.

2002 年　9 月 25 日　初版

著者: 小久保 温 (こくぼ あつし)

〒 030-0943 青森市 幸畑 2-3-1 青森大学 工学部 情報システム工学科

TEL: 017-738-2001（青森大学代表） / FAX: 017-738-2030（青森大学代表）

e-mail: kokubo@aomori-u.ac.jp

web page: http://www.aomori-u.ac.jp/staff/inform/kokubo/

http://www.dma.aoba.sendai.jp/~acchan/

発行: 青森大学出版局

印刷: 青森コロニー印刷

〒 030-0943 青森市 幸畑 字松元 62-3　TEL: 017-738-2021 / FAX: 017-738-6753

A 1st Step of BSD rev. by Atsushi Kokubo

A 1st Step of BSD rev.

by Atsushi Kokubo
Aomori Univ. Press