



小久保温著
青森大学出版局発行



はじめに

この本は、情報システム工学科の1年生の情報工学演習で使用するテキストとして書いたものです。既に、Windows NTで学んだことについては、「使ってみよう! Windows NT 4.0 '99」を参照して下さい。

この本では、FreeBSDを使って、UNIX系のOSと、その上で動作するアプリケーションの使い方を紹介しています。ログイン、ログアウトからはじめ、ファイル操作、Muleの基本的な使い方、プロセス、インターネット・アプリケーションなどを取り上げました。

UNIXの世界は奥が深く、知れば知るほど、簡単な操作で高度なことができます。また、LATEXや、GIMPなどの便利なツールもそろっています。本来は、これらの使い方も紹介した方がよかったですかもしれません。内容の取捨選択には大変苦労しました。

折からの多忙な状態の中、執筆や修正作業を行なっていたため、この本の出版はとても遅れてしまいました。担当して下さった、青森大学出版局の方々には大変感謝いたします。また、本の出版が遅れて、1999年度の情報システム工学科の1年生のみなさんには、申し訳ありません。それから、この本の出版にさまざまな期待を寄せていた方たちに感謝いたします。大いに力づけられました。

そして、次々と起こる数々の出来事を岩のように転がりつつ、時をわかった人に最大の感謝を込めて。

1999年寒空に 小久保 温

商標、登録商標について

本文中に現れる、会社名、商品名は、各社の商標、または登録商標です。なお、本文中では、
TM や R マークは表示しておりません。

目 次

| | |
|---|-----------|
| 第 1 章 UNIX システムとは何か | 1 |
| 1.1 OS とは何か | 1 |
| 1.1.1 どんな OS があるのか? | 1 |
| 1.1.2 OS の役割とは | 1 |
| 1.2 FreeBSD とはどういう OS か | 2 |
| 1.2.1 FreeBSD と Windows の違い | 2 |
| 1.2.2 FreeBSD の利点とは | 4 |
| 第 2 章 UNIX システムを使ってみよう | 5 |
| 2.1 システムの起動、終了、ログイン、ログアウト | 5 |
| 2.1.1 システムの起動 | 5 |
| 2.1.2 ログイン | 6 |
| 2.1.3 ログアウト | 7 |
| 2.1.4 システムの終了 | 8 |
| 2.2 パスワードの変更 | 9 |
| 2.2.1 パスワードの条件 | 9 |
| 2.2.2 どんなパスワードがいいか | 9 |
| 2.2.3 パスワードの変更 | 9 |
| 2.3 X Window System 入門 | 11 |
| 2.3.1 マウスのボタンの呼び方 | 11 |
| 2.3.2 画面内の各部の名称 | 12 |
| 2.3.3 ウィンドウを開く | 13 |
| 2.3.4 ウィンドウを閉じる | 15 |
| 2.3.5 ウィンドウのアイコン化 | 16 |
| 2.3.6 ウィンドウの移動 | 17 |
| 2.3.7 ウィンドウの大きさの変更 | 17 |
| 2.3.8 重なったウィンドウを一番上に移動 | 18 |
| 第 3 章 UNIX コマンド入門 | 21 |
| 3.1 カレンダーの表示: cal コマンド | 21 |
| 3.1.1 cal コマンドを使ってみよう | 21 |
| 3.1.2 コマンドを打ったけどうまくいかない?? | 22 |
| 3.1.3 cal コマンドの引き数 | 22 |
| 3.2 その他の簡単なコマンド | 24 |
| 3.2.1 日時の表示: date コマンド | 24 |
| 3.2.2 login している人のリスト: who コマンド | 24 |
| 3.2.3 今日は何の日?: today コマンド | 25 |

| | |
|--|-----------|
| 3.2.4 おみくじ: <code>fortune</code> コマンド | 25 |
| 3.3 エラーからの回復方法 | 26 |
| 第4章 ファイル操作コマンド | 29 |
| 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 | 29 |
| 4.1.1 ファイル名について | 30 |
| 4.2 ファイルのリストを表示する: <code>ls</code> | 30 |
| 4.3 ファイルの中身を表示: <code>cat</code> | 30 |
| 4.3.1 ファイルをジャンジャン作ってみよう | 31 |
| 4.4 ファイルのコピー: <code>cp</code> | 32 |
| 4.5 ファイルの消去: <code>rm</code> | 33 |
| 4.6 ファイルの名前変更: <code>mv</code> | 33 |
| 第5章 ページャーとマニュアル | 35 |
| 5.1 1画面ずつ表示する: ページャー <code>jless</code> | 35 |
| 5.1.1 ページャーって何? | 35 |
| 5.1.2 <code>jless</code> を使ってファイルを表示しよう | 35 |
| 5.1.3 コマンドの実行結果も1ページずつ: <code>jless</code> とパイプ | 38 |
| 5.2 マニュアルを読む: <code>man</code> と <code>jman</code> | 39 |
| 5.2.1 マニュアルを読んでみよう | 39 |
| 5.2.2 オンライン・マニュアルはどんなときに読むか? | 40 |
| 5.2.3 オンライン・マニュアルの構成 | 41 |
| 第6章 エディタ Mule と Canna による日本語入力 | 43 |
| 6.1 エディタ | 43 |
| 6.2 Mule の使い方: 基本前編 | 43 |
| 6.2.1 Mule の起動 | 43 |
| 6.2.2 Mule の各部名称 | 45 |
| 6.2.3 文字を打ってみよう | 46 |
| 6.2.4 カーソルの移動 | 46 |
| 6.3 Canna による日本語入力 | 48 |
| 6.4 Mule の使い方: 基本後編 | 52 |
| 6.4.1 ファイルの保存 | 52 |
| 6.4.2 Mule の終了 | 53 |
| 6.4.3 [File] メニューの項目 | 54 |
| 6.4.4 Mule のつくり出すファイル | 54 |
| 6.5 Mule の使い方: 応用編 | 55 |
| 6.5.1 部分を選択してコピーして貼り付け | 55 |
| 6.5.2 部分を選択して切り取り | 57 |
| 6.5.3 上下に画面分割 | 58 |
| 6.5.4 左右に画面分割 | 60 |
| 6.5.5 GNU Emacs 系エディタの機能 | 62 |

| | |
|--|------------|
| 第 7 章 印刷 | 63 |
| 7.1 テキスト・ファイルの PostScript への変換: <code>a2ps-j</code> | 63 |
| 7.2 PostScript ファイルの印刷: <code>lpr</code> | 64 |
| 7.3 テキスト・ファイルから直接印刷 | 65 |
| 7.4 印刷状況の確認と取消: <code>lpq</code> と <code>lprm</code> | 65 |
| 7.5 PostScript ファイルの表示: <code>ghostview</code> | 66 |
| 7.6 PostScript ってどんなものか? | 67 |
| 第 8 章 ディレクトリとパス | 69 |
| 8.1 UNIX システムのディレクトリ構造 | 69 |
| 8.1.1 ディレクトリの全体像 | 69 |
| 8.1.2 B 演習室の実際 | 72 |
| 8.2 パス | 72 |
| 8.2.1 根っこからの通り道: 絶対パス | 73 |
| 8.2.2 自分がいるところが中心: 相対パス | 74 |
| 8.2.3 絶対パスと相対パス、どっちがお特? | 76 |
| 8.3 ディレクトリ操作コマンド | 77 |
| 8.3.1 現在いるディレクトリは?: <code>pwd</code> | 77 |
| 8.3.2 ディレクトリを移動: <code>cd</code> | 78 |
| 8.3.3 ディレクトリを作る: <code>mkdir</code> | 81 |
| 8.3.4 ディレクトリを消す: <code>rmdir</code> | 82 |
| 8.3.5 ファイルの移動、コピー | 82 |
| 第 9 章 プロセスとジョブ | 85 |
| 9.1 プロセス | 85 |
| 9.1.1 プロセスの表示: <code>ps</code> | 85 |
| 9.1.2 CPU をよく使っているプロセス: <code>top</code> | 88 |
| 9.1.3 プロセスを止める: <code>kill</code> | 89 |
| 9.2 ジョブのコントロール | 90 |
| 9.2.1 フォアグラウンドとバックグラウンドのジョブ | 90 |
| 第 10 章 パーミッション | 93 |
| 10.1 超入門: C のプログラミング | 93 |
| 10.1.1 ソース・コードの作成 | 93 |
| 10.1.2 コンパイル | 94 |
| 10.1.3 コマンドの実行 | 95 |
| 10.2 パーミッション | 96 |
| 10.2.1 パーミッションの表示: <code>ls -l</code> | 96 |
| 10.2.2 パーミッションの見方 | 96 |
| 10.2.3 パーミッションの変更: <code>chmod</code> | 98 |
| 第 11 章 シェル | 105 |
| 11.1 ユーザ・インターフェース | 105 |
| 11.1.1 GUI の特徴 | 105 |
| 11.1.2 コマンド・ライン・ベースのインターフェースの特徴 | 106 |

| | |
|---|---------|
| 11.1.3 GUI の欠点 | 106 |
| 11.1.4 コマンド・ライン・ベースの欠点を補完するシェル | 107 |
| 11.2 シェルの役割 | 108 |
| 11.3 いろいろな貝殻 | 108 |
| 11.4 シェルの機能 | 110 |
| 11.4.1 ヒストリー機能 | 110 |
| 11.4.2 コマンド・ライン編集機能 | 112 |
| 11.4.3 コマンド別名 | 112 |
| 11.4.4 名前の補完 | 113 |
| 11.5 シェル・スクリプト | 114 |
| 11.6 ワイルドカード | 115 |
| 11.7 シェルの初期設定をカスタマイズ | 116 |
| 11.8 シェルの変更 | 117 |
| 第 12 章 WWW ヘアクセスしてみよう | 119 |
| 12.1 Chimera | 119 |
| 12.2 Netscape Communicator の設定の仕方 | 120 |
| 12.2.1 起動 | 121 |
| 12.2.2 使用許諾条件 | 121 |
| 12.2.3 初回起動時メッセージ | 122 |
| 12.2.4 設定 | 124 |
| 12.2.5 Netscape で日本語入力 | 134 |
| 12.3 テキスト・ブラウザ lynx | 135 |
| 12.4 jweblint | 136 |
| 第 13 章 電子メールとニュース | 137 |
| 13.1 mnews の起動と終了 | 137 |
| 13.2 mnews でニュースを読む | 138 |
| 13.2.1 ニュースを読む | 138 |
| 13.2.2 ニュースに記事を投稿するには | 141 |
| 13.2.3 ニュースに記事にフォローするには | 144 |
| 13.3 mnews で電子メールを使う | 145 |
| 13.3.1 メールを送る | 146 |
| 13.3.2 メールに返事を書く | 150 |
| 第 14 章 ネットワーク機能 | 153 |
| 14.1 ユーザ情報の表示 | 153 |
| 14.1.1 誰が login しているのか? | 153 |
| 14.1.2 より詳しいユーザ情報を調べる: finger | 154 |
| 14.2 他のマシンに入る: telnet | 156 |
| 14.3 他のマシンの他のユーザとおしゃべり | 158 |
| 14.3.1 talk | 158 |
| 14.3.2 3 人以上で会話: phone | 161 |

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。この本では、 FreeBSD(フリー・ビーエスディ) という OS を使って、 UNIX システムの使い方を紹介する。と言っても、よくわからないと思うので、この章では UNIX システム、 OS 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

OS とは、 Operating System(オペレーティング・システム) の略で、コンピュータを使うための、基本となるシステムのプログラムである。

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 シリーズ、 Apple 社の MacOS、 Be 社の BeOS、 IBM 社の OS/2、東大の坂村教授の TRON、 DOS と呼ばれるいくつかの OS、そして 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、その内の Linux や FreeBSD などが最近流行している。

1.1.2 OS の役割とは

例えば、フロッピー・ディスクを使いたいとしよう。

Windows では、デスクトップの [マイコンピュータ] の中に、中に、 [3.5 インチ FD] というアイコンがある。このアイコンがフロッピーを表してして、クリックすると中にあるファイルの一覧を出したりできる。また、ハードディスクのファイルをコピーするには、ファイルのアイコンを持って、このアイコンに重ねてやればいい。

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。

こんなに複雑を自分で全部やっていては、とてもフロッピー 1 枚でさえ、まともに使うことはできないだろう。

OS の役割というのは、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、ユーザはアイコンをクリックしたりするだけで、簡単にコンピュータのシステムを使えるようにすることなのだ。

要するに、OS のおかげで、コンピュータのシステムの細かいところまで一々操作しなくても、簡単に使うことができるというわけだ。

1.2 FreeBSD とはどういう OS か

FreeBSD は、UNIX 系の OS の一つで、主にパソコンと呼ばれているコンピュータで動く。なお、同じような OS に Linux(リナックス) がある。

なお、UNIX 系の OS の操作は、だいたいどの OS でも一緒なので、どれかをマスターすれば他の OS も使うことができるので、とても便利である。

1.2.1 FreeBSD と Windows の違い

FreeBSD では、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たり、WWW にアクセスしたり、絵を描いたりできる¹。つまり、Windows でできたことは、細かいやり方は異なるものの、一通りできる。

では、この 2 つの OS ではどこが違うのか、細かく見てみよう。

1. コマンド・ライン・ベースと GUI

FreeBSD 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。なお、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。

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は GUI(ジーユーアイ) のシステムと呼ぶ。なお、GUI は、Graphical User Interface(グラフィカル・ユーザ・インターフェース) の略である。

なお、FreeBSD などの UNIX 系のシステムでも、X Window System(エックス・ウィンドウ・システム) という GUI を使うことができる。この X を使うことで、Windows にも全く劣らない使い勝手を手にいれることができる。しかも、X はユーザ好みによって、大幅に環境を変えることができる。特に、画面をコントロールしているプログラムを Wi

¹ ちなみに、このテキストも、多くの部分は FreeBSD で作成している。

ンドウ・マネージャと言うが、これを切替えると、全く異なったシステムのように見える。

2. フリーと商品

FreeBSD は、世界中の好意のある人々が協力して作られたシステムで、フリー・ソフトとして公開されている。

ソフトが無料で手にいれることができるだけでなく、コピーを誰に配ってもいい。また、C 言語などで書かれた元のプログラムも全部公開されていて、中を読むこともできるし、プログラムを好きにいじりまわすこともできる。

なお、<http://www.jp.freebsd.org/> が、日本の FreeBSD のサイトで、ここからフルセットをダウンロードできる他、いろいろな情報を読むことができる。また、B 演習室のコンピュータには、FreeBSD の基本部分の元のプログラムが /usr/src 以下におさめられていて、自由に見ることができる。

基本となる部分のプログラムだけでなく、それ以外の様々なプログラムの多くもフリーである。B 演習室のマシンには、C、FORTRAN、Pascal、LISP、Perl、Java、BASIC、Tcl/Tk などのプログラミング言語や、日本語変換システム、ワープロ、メール、ニュース、WWW、ゲームなど、非常にたくさんのソフトが入れられているが、これらも全部無料だった。ただし、これらのソフトを使ってトラブルが発生しても、誰も何の保証もしてくれない。ユーザは自分の責任でこれらを使うことになっている。

一方、Windows は商品である。Windows を使うためには、マシンの台数に応じてパッケージを購入する必要がある。また、よく使われているワープロ、表計算、電子メールなどのソフトも商品が多い。特に、グラフィック、サウンド、ゲーム関係などは、きちんとソフトを購入すれば、値段に応じて素晴らしいものが手に入ることが多い。

3. マルチ・ユーザとシングル・ユーザ

FreeBSD などの UNIX 系のシステムでは、ネットワークにつながっている他のマシンから、入って使うこともできる。また、同じマシンを同時に何人もで使うことができるし、1 人で二重、三重に入って使うこともできる。

例えば、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が、これらは何人もで同時に使ってファイルを置いたりすることができます。

一方、Windows は基本的には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、自由自在にプログラムを動かしたりするには特殊なソフトを使わなければならない。

1.2.2 FreeBSD の利点とは

FreeBSD の利点は、まず、お金をかけずにプログラムを作れることだ。C 言語などが、ソフトを購入しなくても最初から付いていて、使うことができる。マニュアルも、コンピュータの中に入っていて、オンラインで読むことができる（ただし、英語の場合がある）。もちろん、プログラミングの入門書くらいは買った方がいいと思うが、それ以外は心配しなくていい。また、様々な言語を使えることも魅力だろう。なお、後のプログラミング演習のいくつかや、いくつかの卒業研究などでは、UNIX を使うので、マスターしておいて損はない。

それから、前のところにも書いたが、UNIX 系のシステムは、どれでもだいたい操作が共通である。だから、FreeBSD をマスターしておけば、他の UNIX 系のシステムも同様に使うことができる。特に、プロバイダなどのマシンには UNIX 系のものが使われていることが多いし、インターネットに関係した事業を行なっている会社でも UNIX 系の OS を使っていることが多い。実際、青森大学でも、ネットワークで重要な役割を担っているマシンは、すべて UNIX 系の OS が使われている。このように、将来ネットワークに関する技術を身に付けようという場合には、UNIX の知識は必須である。

それから、UNIX 系のシステムでは、自分の好きなように環境を大幅にいじれることが特徴である。たとえば、X Window System は、ウィンドウ・マネージャを切替えることで、全く異なったシステムに見える。また、キー操作なども自分で変更することができ、様々なショートカット・キーを自作できる。

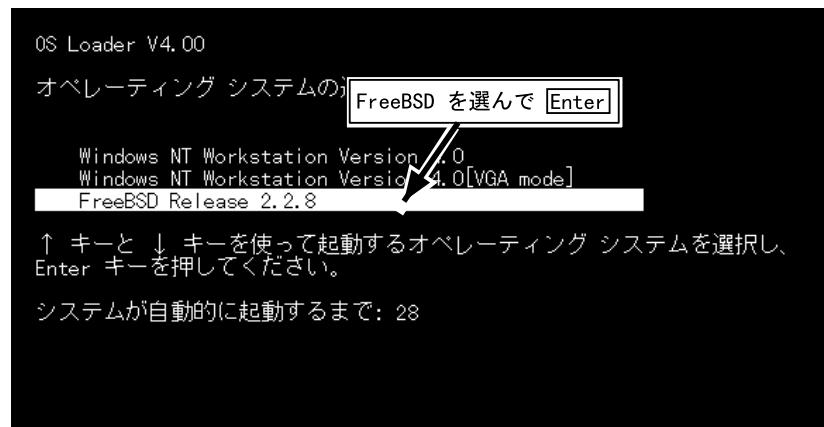
第2章 UNIX システムを使ってみよう

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 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。



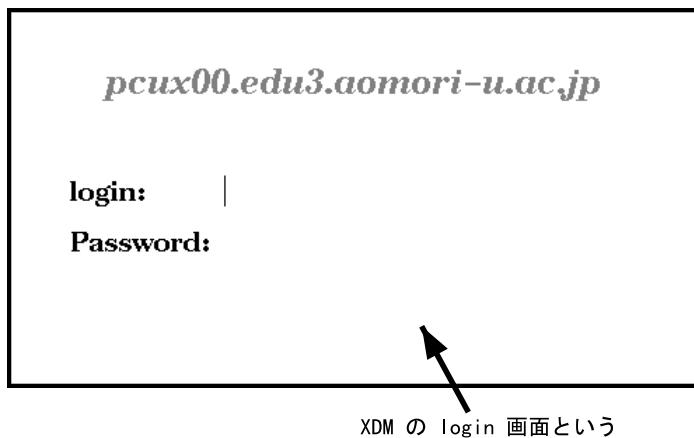
3. boot: と表示されるので、[Enter] を押す(放っておいてもいい)。

```

>> FreeBSD BOOT @ 0x10000: 639/64640 k of memory, internal console
Boot default: 0:wd(0,a)kernel
Usage: bios_drive:interface(unit,partition)kernel_name options
      bios_drive  0, 1, ...
      interface    fd, wd or sd
      :
      中略
      :
hardware parameters (c), and print verbose messages (v)
boot: [Enter]

```

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 XDM¹ の login 画面になる。



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

まず、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だ。

ID は UNIX システムを使うときのユーザの名前である。ID 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、ei99001、0198011、tz01a92 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。

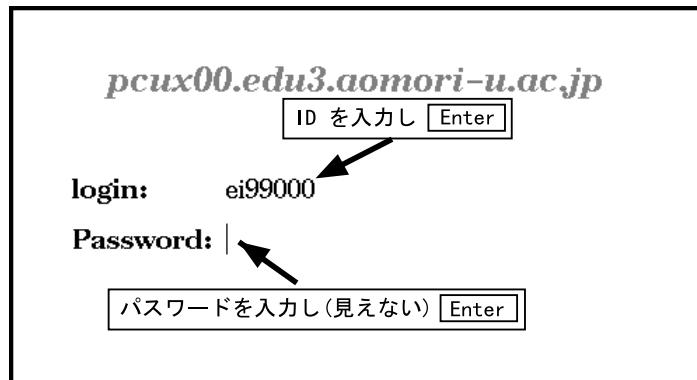
パスワードは、たぶん、システムの管理者が用意してくれたものを最初に渡される。これをユーザは、自分で好きなものに変更して使う。変更の仕方は、後の方で説明しよう。

XDM からのログイン

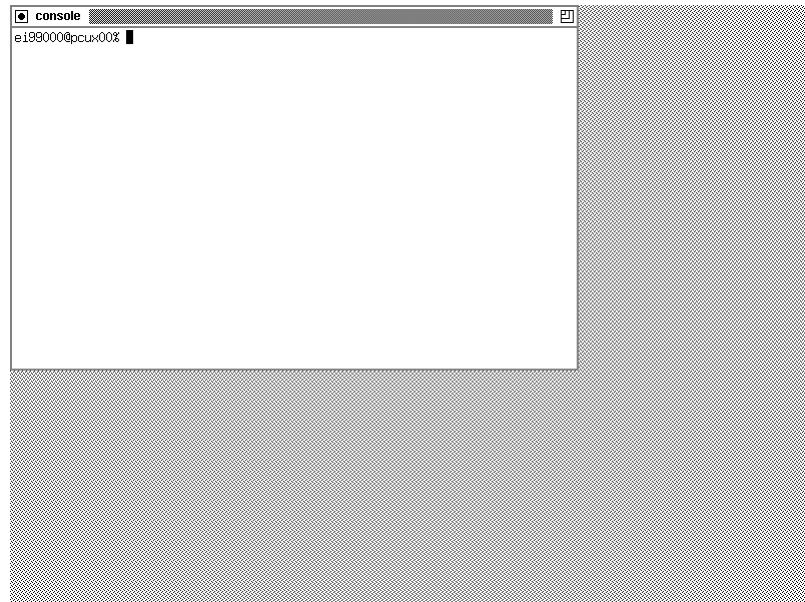
演習室 B のマシンは、UNIX システムが起動すると、XDM のログイン画面ができる。この画面からログインするには、次のようにする。

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]。

¹ X Display Manager



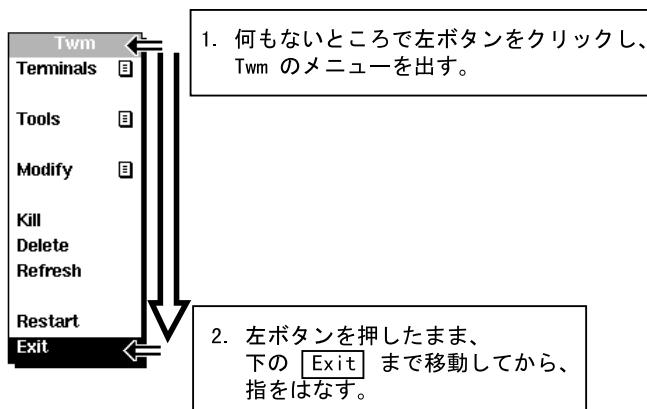
3. login に成功すると、次のような画面になる。失敗した場合は、赤字で Login incorrect と表示されるので、やりなおす。



2.1.3 ログアウト

UNIX システムからのログアウトの方法を説明しよう。

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。

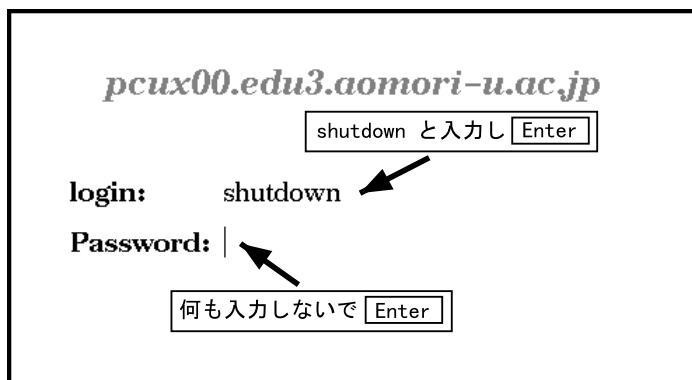
2.1.4 システムの終了

通常、UNIX システムは、電源を切らないで 24 時間動かしちゃ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終了できる。

しかし、演習室 B のマシンは、Windows NT 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、演習室 B 以外のマシン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 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 haled. (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. 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、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、数字だけ、記号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。

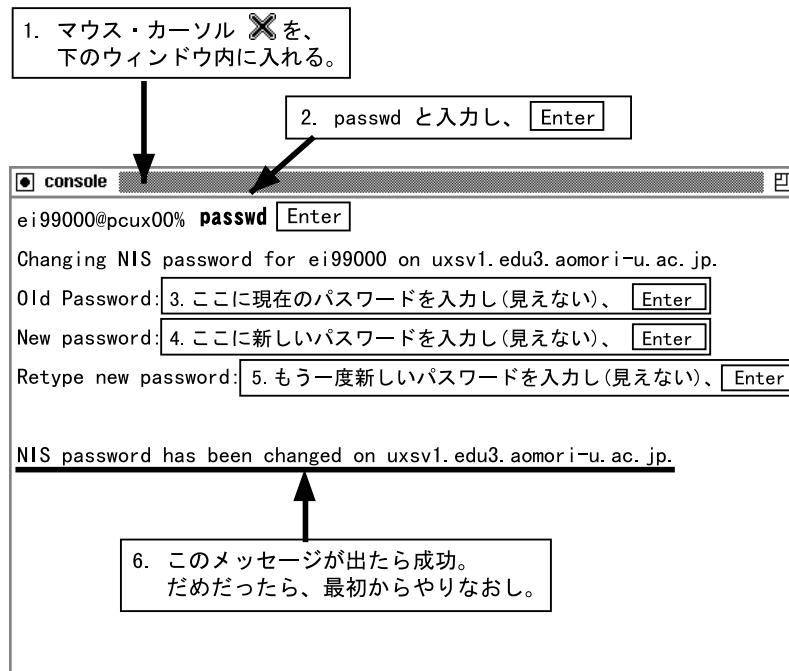
1. 辞書にそのまま載っているような単語。
2. 自分の名前、電話番号、住所など、他人から簡単に想像できるものを そのまま 使う。
3. あまりに長かったり、自分でも覚えられないもの。

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、
3. Old Password: に対して、現在のパスワードを入力し、
4. New password: に対して、新しいパスワードを入力し、

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 characters 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打った、1回目と2回目のものが異なっていた。

2.3 X Window System 入門

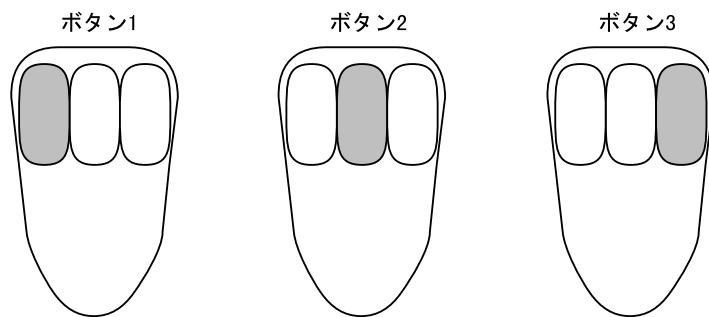
ここでは、X Window Systemについて、簡単に紹介する。詳しい使い方は、後の章で行なう。

2.3.1 マウスのボタンの呼び方

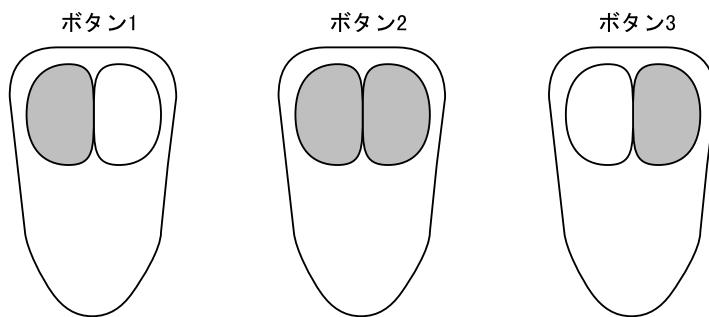
まず、最初にマウスのボタンの呼び方を紹介しよう。X Window Systemを使うUNIXでは、普通、3つボタン・マウスを使う。それぞれのボタンは、左から順番に、ボタン1、2、3と呼ぶ。

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ことで、真中のボタン2の代わりになる。

3つボタン・マウスの場合

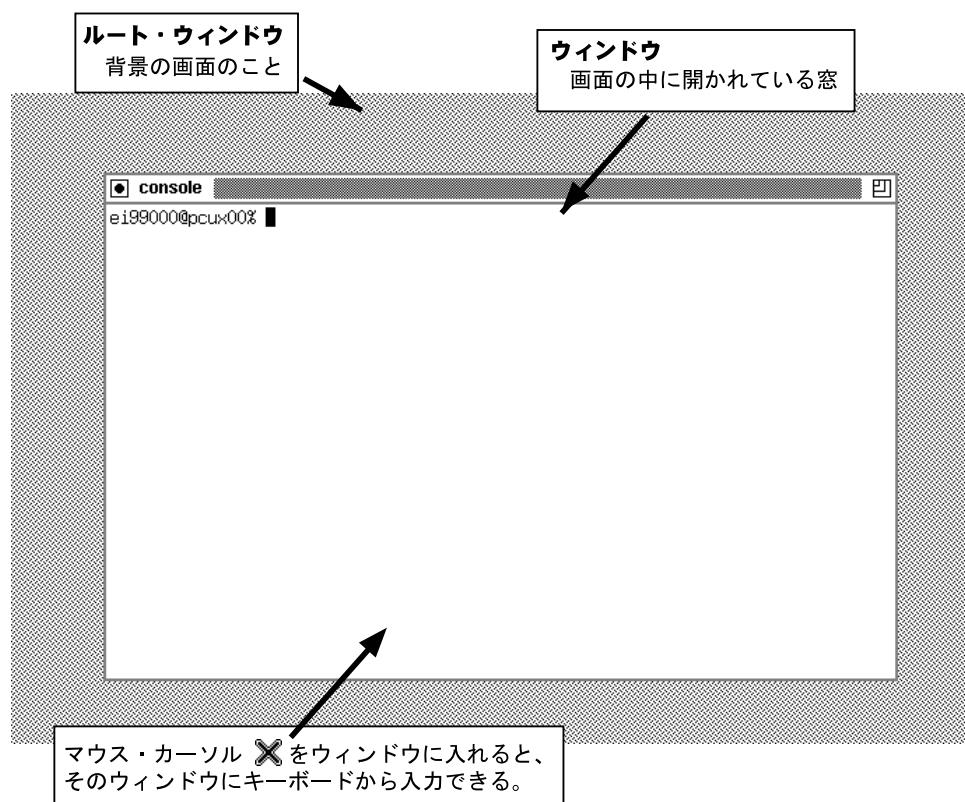


2つボタン・マウスの場合

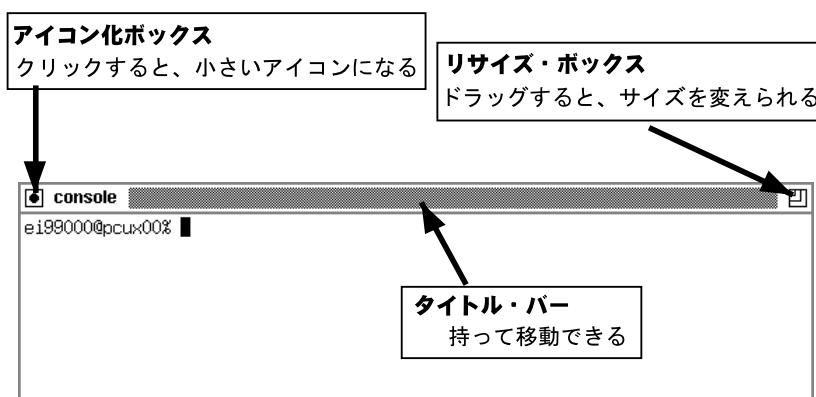


2.3.2 画面内の各部の名称

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」、画面の中を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 & [Enter]」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 & [Enter]
- xeyes & [Enter]
- xcalc & [Enter]
- xclock & [Enter]
- emiclock & [Enter]
- xengine & [Enter]
- tksol & [Enter]
- xmine & [Enter]
- xbill & [Enter]
- oneko & [Enter]
- xearth & [Enter]

なお、コマンドに「&」を付けずに実行すると、コマンドを実行した元のウィンドウにコマンド打っても、効かなくなる。

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

このとき、[Ctrl]+Z を入力すると、「^Z」と表示され、再びコマンドが打てるようになる。

```
% kterm [Enter]  
^Z  
[1]+ Stopped kterm  
% (ここにコマンドが打てるようになった)
```

こうやって、再びコマンドが打てるようになったら、「bg」というコマンドを打っておくとよい。

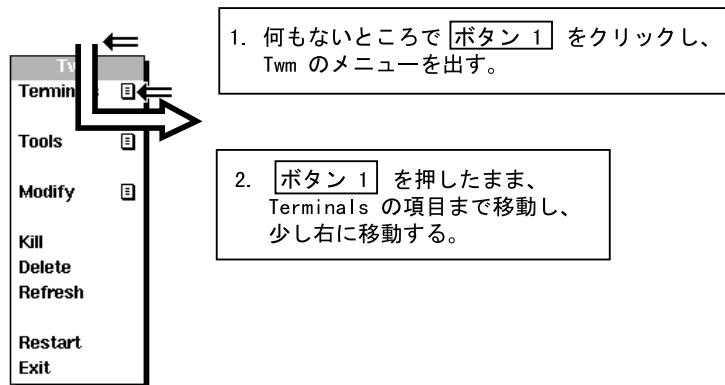
```
% kterm [Enter]  
^Z  
[1]+ Stopped kterm  
% bg [Enter]  
[1]+ kterm &
```

ウィンドウ・マネージャのメニューから選択

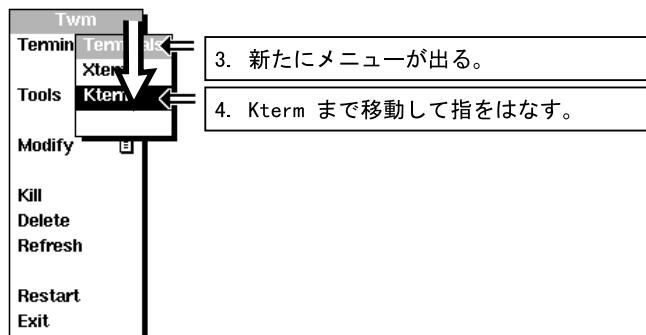
ルート・ウィンドウでマウスのボタン1をクリックするとメニューが出る。このメニューからウィンドウを開くこともできる。

では実際に、次のようにしてみよう。

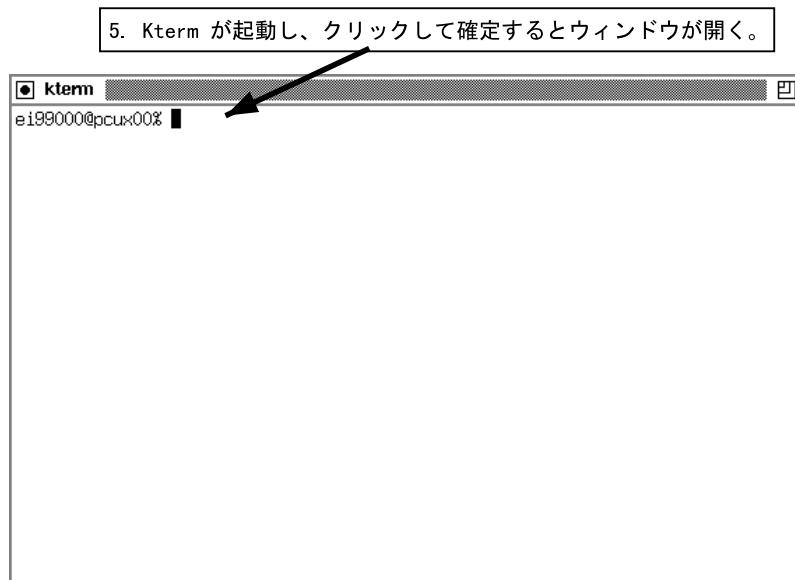
1. ボタン1を何もないところでクリックし、Twmのメニューを出す。
2. ボタン1を押したまま、Terminalsまで移動し、少し右に移動する。



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がある。

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。

ei99000@pcux00% exit

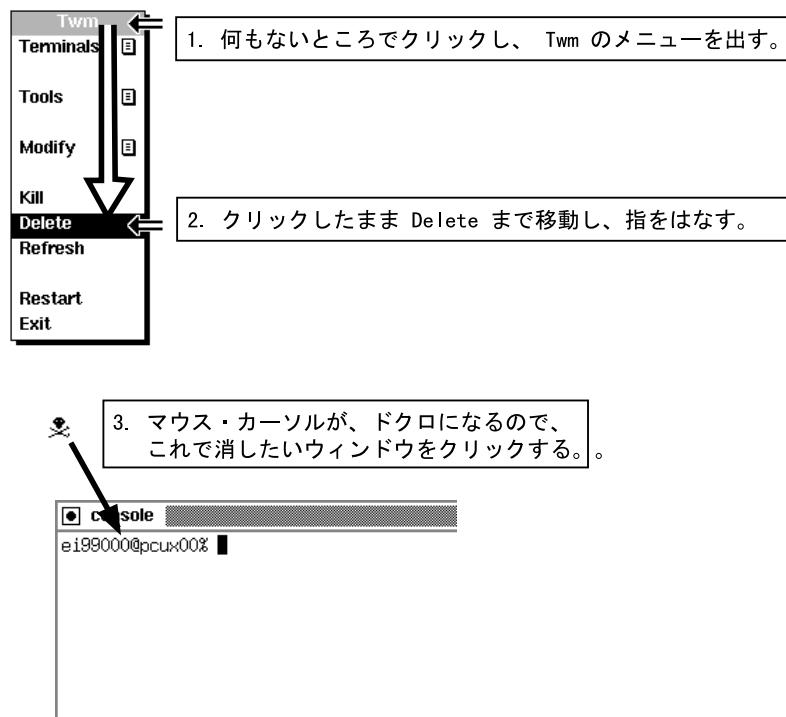
ウィンドウのメニューから exit などを選ぶ

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

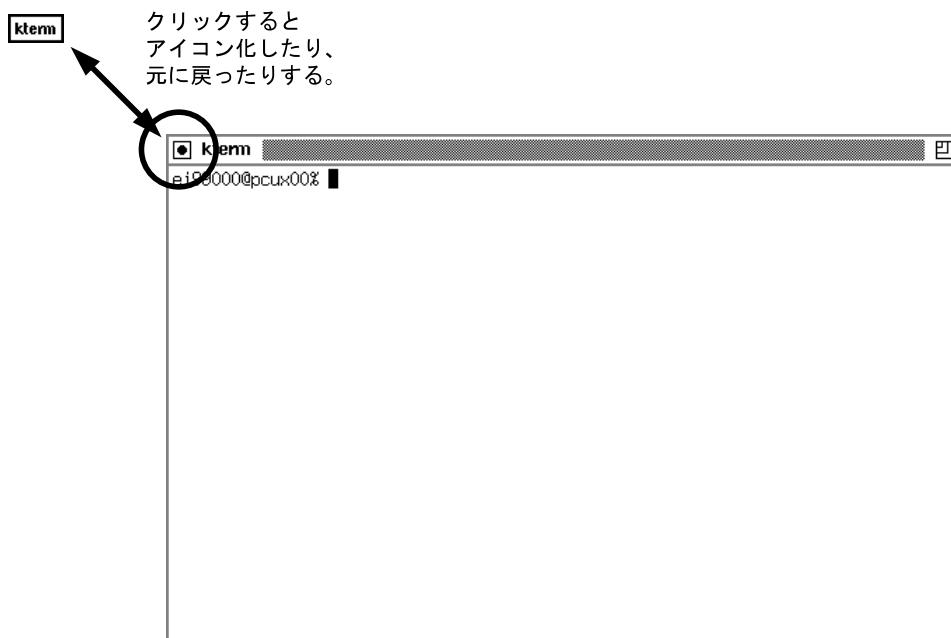
上の 2 つの方法で終了できない場合は、Twm のメニューから Delete を選んで終了させられる。なお、Kill を選んだ場合は、「強制終了」になる。

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。



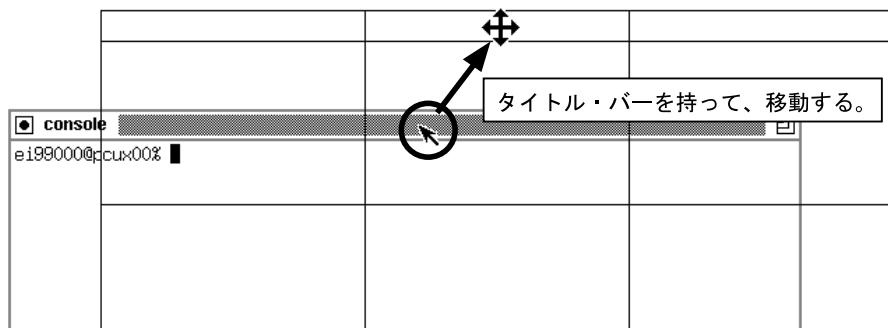
2.3.5 ウィンドウのアイコン化

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・バーを持って、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

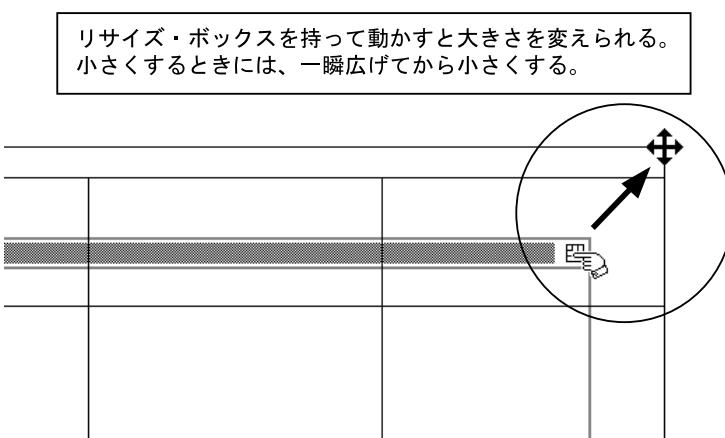
ウィンドウの大きさは、「リサイズ・ボックス」を持って、動かすことで変更できる。

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。

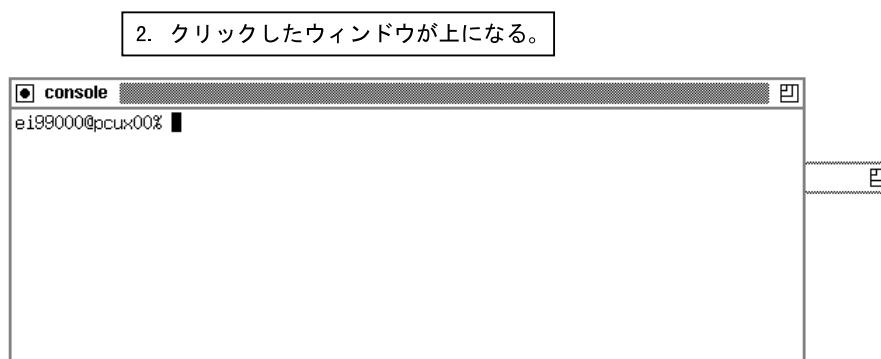
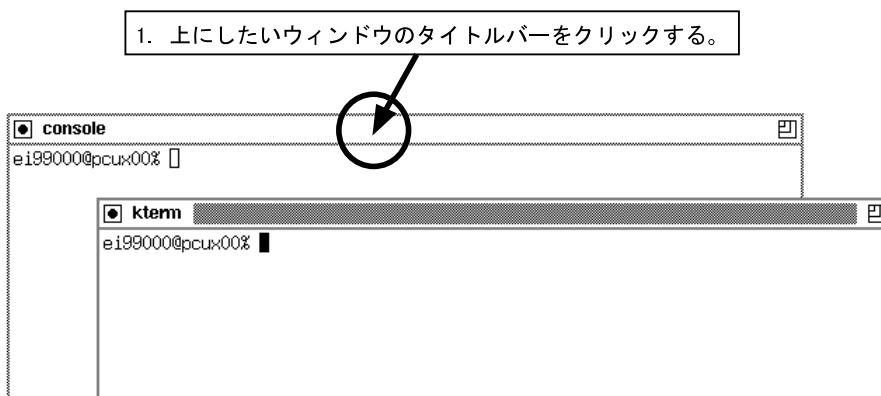


2.3.8 重なったウィンドウを一番上に移動

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

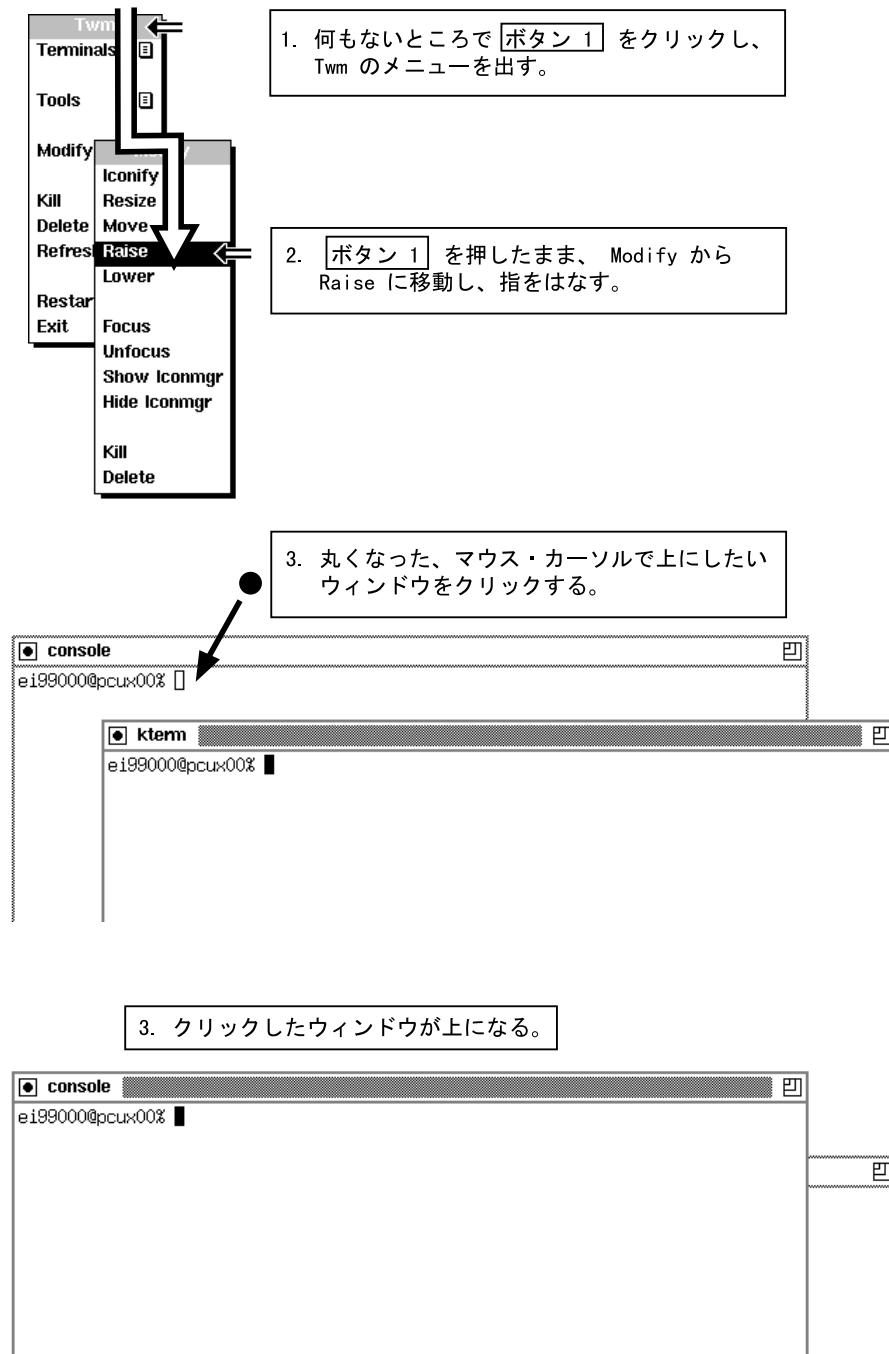
上にしたいウィンドウのタイトル・バーをクリック

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 Raise 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、プログラムを走らせることが多い。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。英語とは言っても、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だ。また、C 言語でやさしいプログラムを作つて実行するくらいだったら、極端な話 10 個くらいのコマンドを知つ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、どの UNIX システムでも、ほとんど同じように使うことができるので、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。

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

では、ためしにプロンプトに向かって、次のように `cal`¹と打ち込んでみよう。なお、UNIX システムでは、大文字と小文字は区別される。このコマンドは全部小文字で打つこと。

```
% cal
```

すると、次のように、今月¹のカレンダーが表示される。さつき、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つたが、`cal` は `calendar` (カレンダー) を省略したものだ。

```
% cal
 10 月 1999
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
%
```

¹この本を書いたのは 1999 年の 10 月でした。

3.1.2 コマンドを打ったけどうまくいかない??

- 打ち間違えた文字を消すには?

打ち間違えた文字は、**[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きる。

```
% cak[Delete]
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・タイプしたときに運が悪いと、プログラムが暴走して、プロンプトが出て来なくなることがある。そんなときは、**[Ctrl]+C** (**[Ctrl]** を押しながら **[c]**) と打てば、たいてい止められる。

3.1.3 `cal` コマンドの引き数

もう少しつっこんだ `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 1999[Delete]` と月と年を指定して、実行してみよう。

```
% cal 4 1999[Delete]
        4月 1999
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、1999年4月のカレンダーが表示される。

では、今度は、月を省略して、`cal 1999`と打ってみよう。

```
% cal 1999
```

1999

| 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 | 2 | 3 | 4 | 5 | 6 | | 1 | 2 | 3 | 4 | 5 | 6 | |
| 3 | 4 | 5 | 6 | 7 | 8 | 9 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| 10 | 11 | 12 | 13 | 14 | 15 | 16 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 17 | 18 | 19 | 20 | 21 | 22 | 23 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 21 | 22 | 23 | 24 | 25 | 26 | 27 |
| 24 | 25 | 26 | 27 | 28 | 29 | 30 | 28 | | | | | | | 28 | 29 | 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 | | 1 | 2 | 3 | 4 | | | |
| 3 | 4 | 5 | 6 | 7 | 8 | 9 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 10 | 11 | 12 | 13 | 14 | 15 | 16 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| 17 | 18 | 19 | 20 | 21 | 22 | 23 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |
| 24 | 25 | 26 | 27 | 28 | 29 | 30 | 28 | 29 | 30 | | | | | 26 | 27 | 28 | 29 | 30 | 31 | |
| 31 | | | | | | | | | | | | | | | | | | | | |

%

このように 1999 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、後ろの部分だけが表示される²。

ひとまず、ここまでのことろをまとめよう。`cal` コマンドは次のように使う。

cal 「月」 「年」

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、「^ひ ^{すう}argument」と言う。これはプログラミングの本や、マニュアルなどでよく使う用語なので、おぼえておこう。

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。



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 56 年は、西暦 1981 年。昭和の年数に 25 を足すと、西暦の下 2 ケタになる。

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  
1999年 10月 01日 金曜日 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、別のマシンから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  tttyp0    9/01 01:06  (:0.0)  
kokubo  tttyp1    9/01 01:05  (:0.0)  
tomoda  tttyp2    9/01 02:31  (172.31.1.1)  
%
```

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 と打った人の ID を表示する。

```
% whoami  
kokubo  
%
```

3.2.3 今日は何の日?: today コマンド

today 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³。

```
% today
```

こんばんは。

きょうは、平成 11 年 10 月 1 日（金曜日）です。

旧暦では、平成 11 年 8 月 22 日（大安）[小潮] です。

干支では、己卯（つちのとう）の年、丙戌（ひのえいぬ）の日です。

九曜星で、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 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⁴。

```
% 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このコマンドもオプションなので、入っていないもある。

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、**[Delete]** 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** (**[Ctrl]** を押しながら **[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

X Window System でウィンドウを開くコマンドに「&」を付けずに実行すると、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** (**[Ctrl]** を押しながら **[z]**) と打って、次に **bg** というコマンドを打てばよい。

また、ある種の端末では、**[Ctrl]+S** (**[Ctrl]** を押しながら **[s]**) で、入力がロックされる。これは、**[Ctrl]+Q** (**[Ctrl]** を押しながら **[q]**) でキャンセルできるので、思い当たるふしがある場合には、試してみるといいだろう。

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」などのコマンドを入力すると、止められることがある。

例えば、**bc** という電卓のプログラムがある。これは、「**bc**」で起動できる。

```
% 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止めるには、**[Ctrl]+D** である。

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。

なお、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

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、コピーしたりという、ファイルの操作は、コンピュータの操作の基本中の基本だ。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

`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 1999
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、`ThisMonth` (今月) というファイルに入れてみよう。

```
% cal > ThisMonth
%
```

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 `ThisMonth` という名前のファイルに入ったのだ。

このように、

「コマンド」 > 「ファイル名」

としてやると、ファイルにコマンドの実行結果を入れることができる。

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。

4.1.1 ファイル名について

ところで、この例では、ファイル名を ThisMonth にしたが、別にこれは何でもいい。たとえば、KonGetsu とかでも大丈夫だ。

ただし、ファイル名には、「*」や、「?」などの特殊な記号は使わない方がいい²。また、ファイル名の最初の文字を「-」にするのはやめた方がいい³。

よくあ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。

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。

4.2 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 - list」という意味だ。

では、ls コマンドを実行してみよう。

```
% ls
ThisMonth
%
```

さっしき作った ThisMonth があることがわかる。

このように、

と単に打つと、ファイルのリストを表示することができる。

4.3 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。次のように実行してみよう。

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ので。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。

```
% cat ThisMonth  
10月 1999  
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 – concatnete」から来ている。このように、

```
cat 「ファイル名」
```

としてやると、ファイルの中身を画面に表示することができる。

ちなみに、どの辺が「連結表示」か説明しよう。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」… という順番に中身が(連結して) 表示されるのである。

4.3.1 ファイルをジャンジャン作ってみよう

1999 年のカレンダーを表示するには、cal 1999 だ。これを ThisYear というファイルに入れるには、次のようにする。実際に実行してみよう。

```
% cal 1999 > ThisYear  
%
```

ls コマンドで、ThisYear ができたか、確かめてみよう。

```
% ls  
ThisMonth      ThisYear  
%
```

確かにできている。次は、1999 年 4 月のカレンダーも、同じように LastApr というファイルに入れてみよう。

```
% cal 4 1999 > LastApr  
%
```

`ls` と打つと、確かに `LastApr` もできていることがわかる。

```
% ls
LastApr      ThisMonth      ThisYear
%
```



練習

1. 自分が生まれた月のカレンダーを、`MyBirth` 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。

4.4 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「copy - コピー」という意味だ。`ThisMonth` をコピーして、`KonGetsu` というファイルを作るには、次のように打つ。

```
% cp ThisMonth KonGetsu
%
```

`ls` と `cat` で確かめてみよう。

```
% ls
KonGetsu      LastApr      MyBirth      ThisMonth      ThisYear
% cat KonGetsu
10月 1999
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、`KonGetsu` というファイルができていて、`ThisMonth` 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」
```

4.5 ファイルの消去: rm

ファイルを消すには、 rm を使う。 rm は、「remove – 消去」を短くしたものだ。

次のようにして、 KonGetsu を消してみよう。

```
% rm KonGetsu  
%
```

ls で消えたか確認してみよう。

```
% ls  
LastApr      MyBirth      ThisMonth      ThisYear  
%
```

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」
```

4.6 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、「move – 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、 ThisMonth を KonGetsu というファイル名に変えてみよう。

```
% mv ThisMonth KonGetsu  
%
```

ls で名前が変わっているか、確認してみよう。

```
% ls  
KonGetsu      LastApr      MyBirth      ThisYear  
%
```

ThisMonth がなくなって、 KonGetsu 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」

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、もちろんできる。



練習

1. `ThisYear` をコピーして、`Kotoshi` というファイルを作つてみよう。
2. `Kotoshi` ができているか、`ls` で確かめてみよう。
3. `Kotoshi` を消してみよう。
4. `Kotoshi` が消えたか、`ls` で確かめてみよう。
5. `ThisYear` の名前を、`1999nen` 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。

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。

ところで、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 ページャーって何？

前の章で説明したように `cal 1999 > ThisYear` とすると、1999 年のカレンダーを `ThisYear` 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。なお、「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

最初にでかいファイルを用意する。まず、次のようにして、1998 年から 2000 年までの 3 年分のカレンダーを、`1998nen ~ 2000nen` というファイルにそれぞれいれてみよう。

```
% cal 1998 > 1998nen
% cal 1999 > 1999nen
% cal 2000 > 2000nen
%
```

そして、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、`Years` という一つのファイルに突っ込む。

```
% cat 1998nen 1999nen 2000nen > Years
%
```

こうして作った `Years` は 102 行くらいの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読んでみる。

`jless` の使い方は、

jless 「ファイル名」

である。`jless Years` と入力してみよう。

```
% jless Years
```

すると、下のように `Years` の最初の1ページ分が表示される。

ここで、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] を押す。

| | | |
|---|---|---|
| 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 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 | 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 | 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 |
| 1998 | | |
| 4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 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 | 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 | 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 |
| 5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 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 | 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 | 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 |
| 6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 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 | 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 | 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 |
| 7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 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 | 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 | 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 |
| 8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 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 | 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 | 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 |
| 9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 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 | 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 | 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 |

Years

jless の基本操作は次のようになっていて、これだけ知っていれば、一応使うことができる。

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 を抜ける | [q] (クイット quit) |

また、上のもの以外に、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を打って [Enter] |
| 後方の文字をサーチ | ? に続けてサーチしたい文字を打って [Enter] |

なお、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たまにある。そんなときは、画面の一番下を見よう。jless 使用中には、「Years」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くと表示されるメッセージである。

練習

1. 上に書いたように Years というファイルを作って、jless でのぞいてみよう。
2. 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¹ 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。この場合も、jless の使い方は、さっきファイルの中を見たときと全く同じだ。

こんにちは。

きょうは、平成11年10月1日（金曜日）です。

旧暦では、平成11年8月22日（大安）〔小潮〕です。

干支では、己卯（つちのとう）の年、丙戌（ひのえいぬ）の日です。

九曜星で、二黒です。

***** きょうは何の日かな? *****

昨日は週刊 Bing 発売でした。

昨日は週刊 DODA 発売でした。

昨日は隔週 ぴあ中部版 発売でした。

昨日は週刊 コミックモーニング 発売でした。

昨日は週刊 ヤングジャンプ 発売でした。

昨日は月刊 たのしい幼稚園 発売でした。

昨日は月刊 おともだち 発売でした。

昨日は月刊 ペンギンクラブ 発売でした。

昨日は月刊 Do Book 発売でした。

昨日は月刊 消費者 発売でした。

昨日は月刊 DRAGON 発売でした。

昨日は月刊 まんが シャレダ!! 発売でした。

昨日は月刊 GAME 遊 発売でした。

昨日は月例 ビッグ・コミックス 発売でした。

昨日は月刊 マル勝 PC エンジン 発売でした。

コーヒーの日です。

:■

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使うことができる。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。

¹これはオプションなので、入っていないこともある。そのときは別のコマンドでためしてみよう。

5.2 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出ることがある。なお、マニュアルは、オプションなので、インストールされていない場合もある。

5.2.1 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。なお、この表示は、`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 | <code>cal, ncal</code> - カレンダおよびイースターの日付を表示する | |
| 書式 | <code>cal [-jy] [[month] year]</code> <code>ncal [-jJpwy] [-s country_code] [[month] year]</code> <code>ncal [-Jeo] [year]</code> | |
| 解説 | <code>cal</code> は簡単なカレンダを表示します。また <code>ncal</code> 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、 cal のマニュアルを読んでみよう。
2. who のマニュアルを読んでみよう。
3. ls のマニュアルを読んでみよう。

5.2.2 オンライン・マニュアルはどんなときに読むか?

実際問題として、 UNIX を使い始めたばかりのときに、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。複雑なことのできる強力なコマンドの場合、長いマニュアルが表示されて、特にそう感じことだろう。

そこで、オンライン・マニュアルの活用の仕方を紹介しておこう。

まず、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。こういうのは、初めて勉強しようという人にとっては、ちょっとつらいことになる。

では、どういうときに役に立つかと言うと、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忘れたときや、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。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。実は、 C 言語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見ると、意外に便利だったりする。

2. キーワード検索したい。

実はマニュアルは、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事だ。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不便なので、前の節で紹介した jless の検索機能を使って、キーワードでサーチして読むといい。

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ある。

5.2.3 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が、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 のシステムで使う関数など |
| 3 ライブラリ | C 言語などの関数など |
| 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 |

なお、これ以外にも、セクションには `n` や `l` があり、例えば `n` には Tcl/Tk などマニュアルが入っている。

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には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」、「コマンド名」 | jless

スペース で 1 画面進み、 b で 1 画面戻る。終了するには q

jman : 日本語マニュアルの表示

使い方: jman 「コマンド名」など

第6章 エディタ Mule と Canna による日本語入力

自由にファイルを作る

ここまでのことでは、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。自分で自由に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、プログラムを書きはじめるための準備もほぼ終わりだ。それから、UNIX システムでは、環境設定は基本的に隠しファイルで行なわれているが、このファイルもエディタで書き換えることができ、自分の好きなように環境を設定することも可能になる。

また、この章では、日本語入力の仕方も説明する。

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」、Macintosh では「Simple Text」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 グニュー イーマックス 系と ヴィジュアル vi 系の GNU Emacs 系と vi 系のエディタである。

ここでは、GNU Emacs 系のエディタである ミュール Mule を紹介する。Mule は GNU Emacs の多言語拡張版で、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、実際にいろいろな言語を編集することができる。

今回紹介する Mule も含めて GNU Emacs 系のエディタは、実に様々な機能を持っているが、ここでは基本的なことだけを紹介する。

6.2 Mule の使い方： 基本前編

6.2.1 Mule の起動

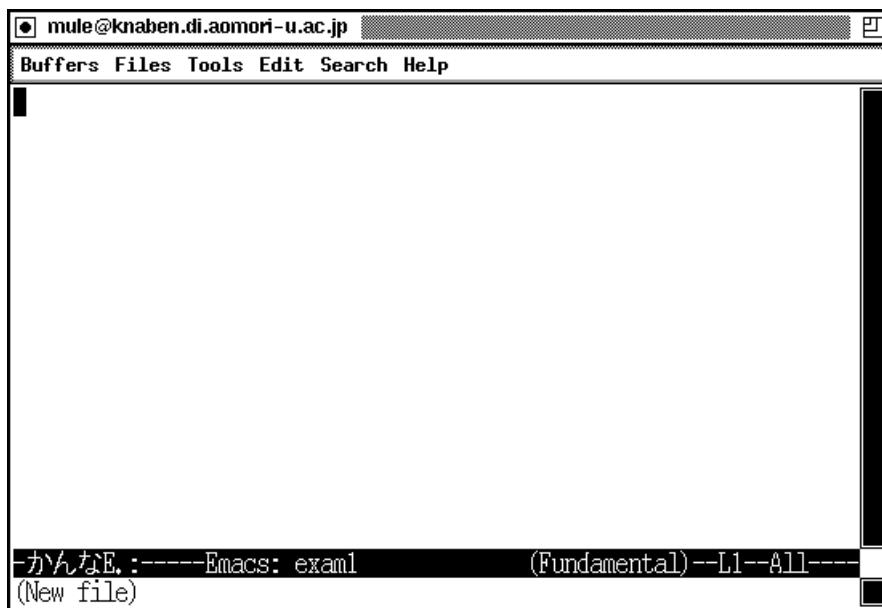
X Window System を使っているとき、Mule は次のようにして起動する。

`mule 「ファイル名」 &`

では、`mule exam1 &`と入力して、`exam1`というファイルを編集してみよう。

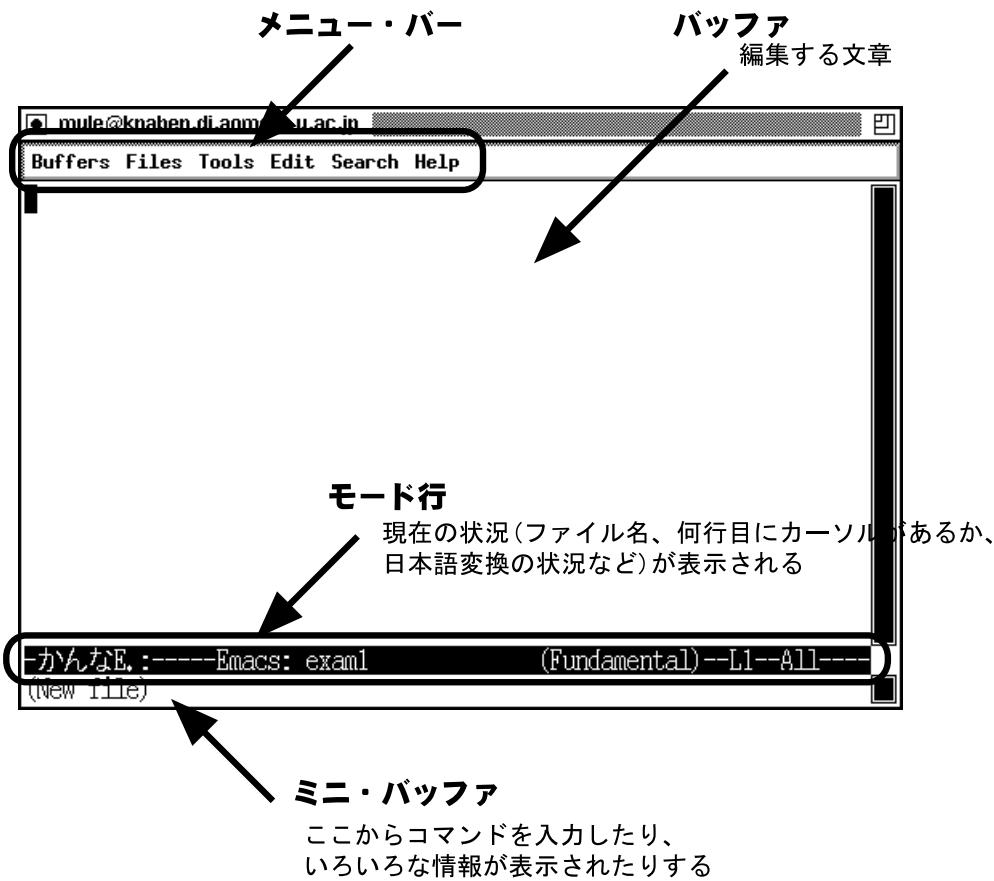
```
% mule exam1 &
%
```

すると、下のようなウィンドウが開く。



6.2.2 Mule の各部名称

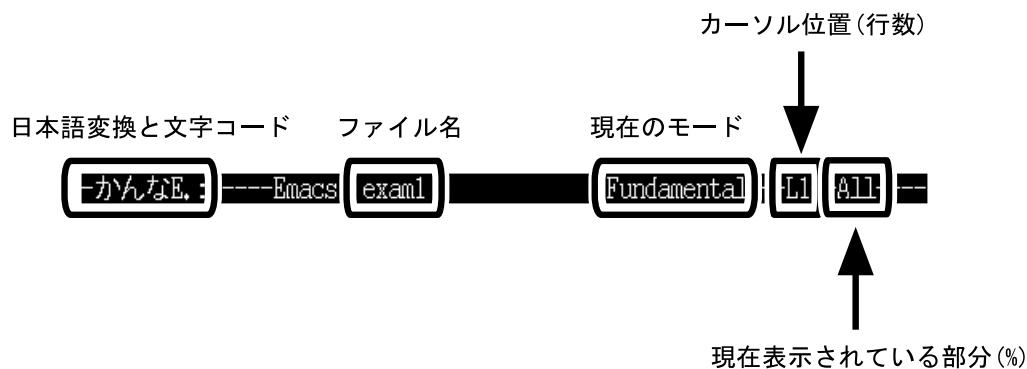
各部の名前を紹介しよう。



一番上が「メニュー・バー」。これを使うと、簡単に Mule を操作できる。

真ん中の広い部分が「バッファ」で、ここで文章を作る。

黒いラインが「モード行」。ここにファイルの漢字コードや、今見えている部分がどこかという情報や、漢字変換の状況などが表示される。



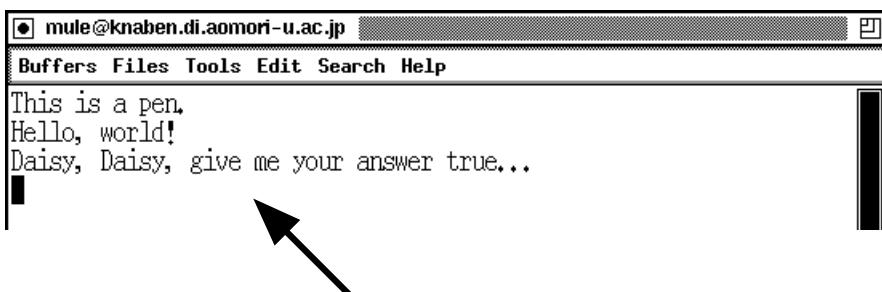
一番下が「ミニ・バッファ」で、コマンドを入れる部分。

右の黒い部分は「スクロール・バー」。長いファイルを使っているときには、これをマウスの

[ボタン 3]でつかんで移動することができる。

6.2.3 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。



マウス・カーソルをウィンドウに入れ、
文字を打つと、入力される。
文字を消去するには [Delete]

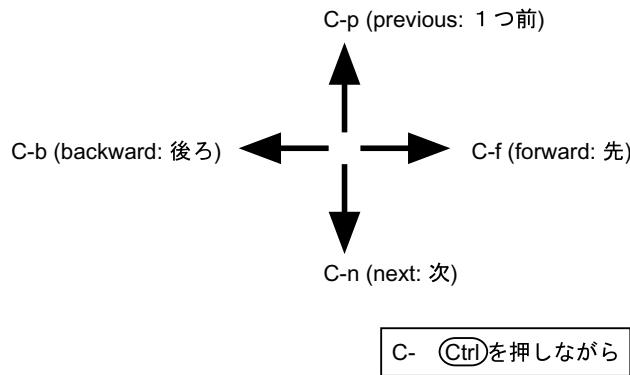
6.2.4 カーソルの移動

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。

しかし慣れてくると、手をマウスに置いたり、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 | ひとつ前へ previous |
| | C-n | 次へ next |
| | C-f | 前に forward |
| | C-b | 後ろに backward |



ちなみに Mule では、**[Ctrl]+[p]** のことを C-p と書くので、注意しよう。

また、これ以外に M-x とかいうのもある。M は「**[Esc]** を一瞬押すこと」を意味している。

[Esc] は、**[Ctrl]** のように押しっぱなしにしてはいけない。

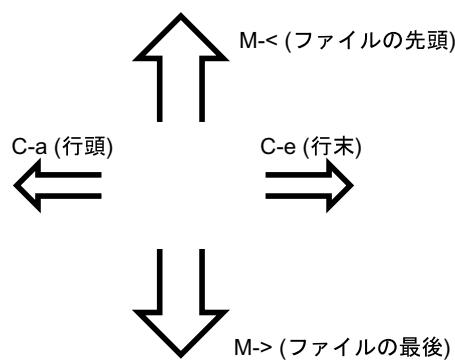
つまり M-x と書いたら、「一瞬 **[Esc]** を押して、x を押す」という意味だ。

なお、、 Mule の中のコマンドのキャンセルは C-g だ。 変なったりわけわからなくなったら、とりあえず C-g してみよう。

上のコマンド以外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 <input type="text"/> 行番号 <input type="text"/> |



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「かんな (Canna)」というシステムを使う。

まず、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

かんなで、日本語モードと英語モードの切り替えは C-o だ。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。

ちなみに、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) \leftrightarrow (英語モード) が切り替わる

-[あ]E:--***-Emacs: exam1



日本語入力モードでは、
[あ] と表示される。

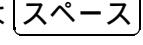
日本語の入力

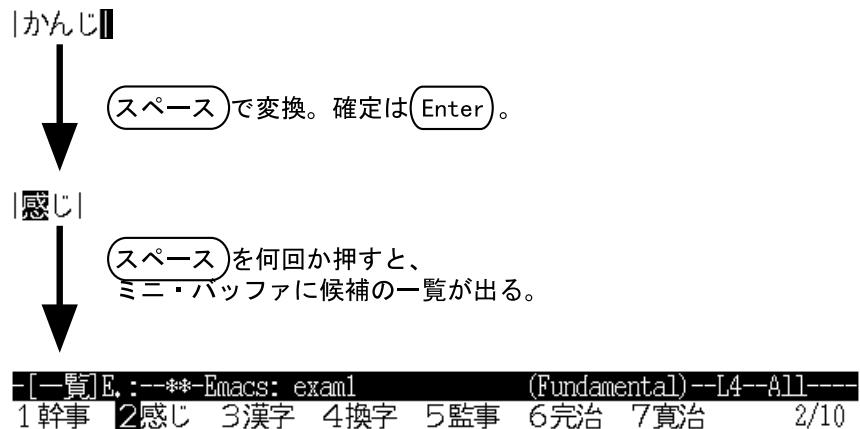
単語 1 個をいろいろ変換

まず、日本語モードにしよう。ここでは、ローマ字入力で、日本語が打てる。

ちなみに「ん」は、「nn」か「n」か「mn」だ。また、「しゃ」は、「sya」や「sha」など。「い」は、「xi」でそれぞれ入力できる。

試しに「かんじ」と打ってみると、「|かんじ|」と表示される。ここで  すると、ひらがなになる。

漢字に変換するには  を押す。 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、変換をやめる場合には、C-g である。



なお、カタカナや(全角の)英語に変換するには、C-p を何回か入力する。ちなみに、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

表 6.3: 変換のコマンド

| 動作 | コマンド |
|------------|------|
| 漢字へ変換 | スペース |
| カタカナや英数に変換 | C-p |
| 変換の確定 | ➡ |

文を変換

文を日本語変換してみる。

試しに「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 6.4: 文節操作

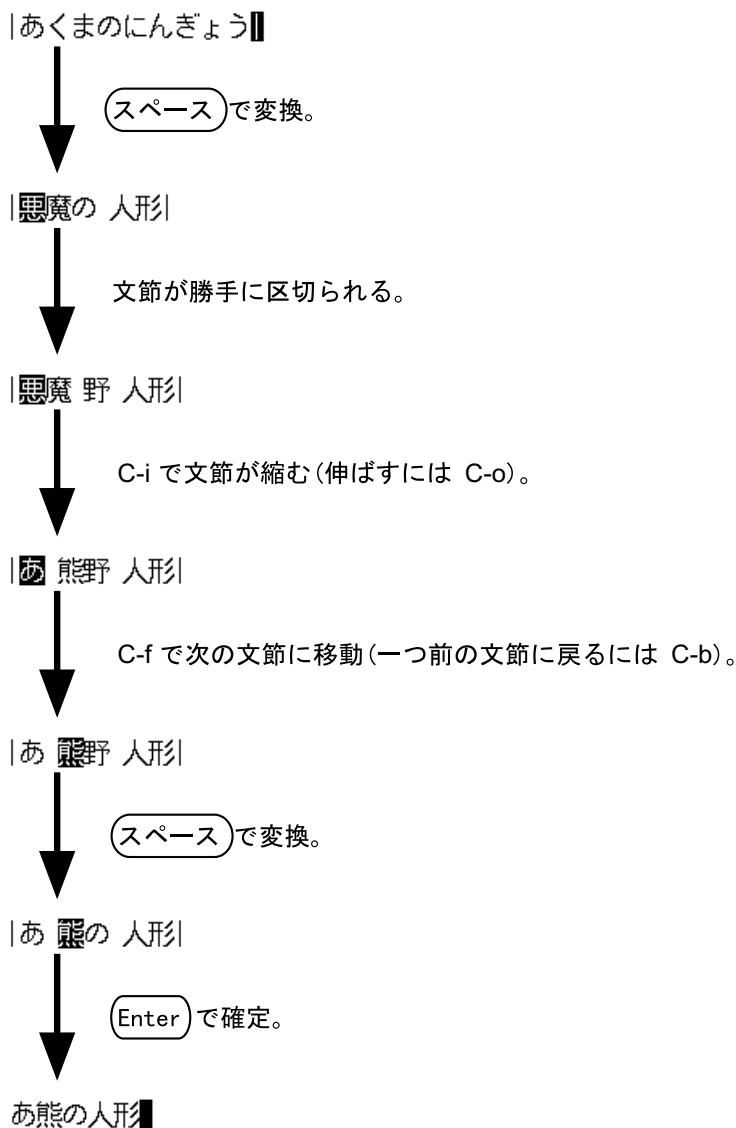
| 動作 | コマンド |
|------------------|------|
| 文節を縮める | C-i |
| 文節を伸ばす | C-o |
| 一つ前の文節にカーソルを移動する | C-b |
| 次の文節にカーソルを移動する | C-f |

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。



その他の入力

まず、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。

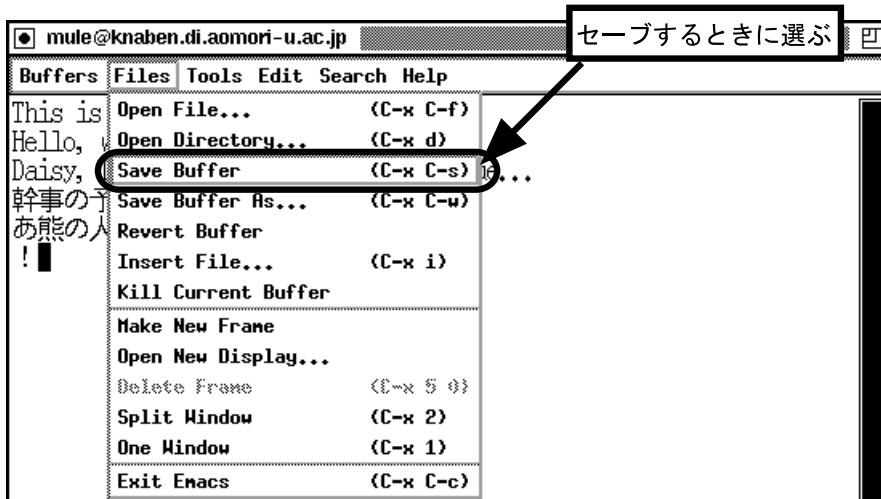
```
[記号]E:--**-Emacs: examl (Fundamental)--L5--All---  
[212a] 、 。 、 . . : ; ? ! " " \\
```

これ以外にも部首索引は [F.3]、コード入力が [F.2] と、いろいろなモードがある。

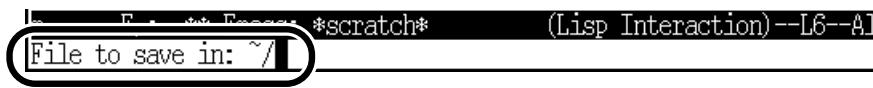
6.4 Mule の使い方: 基本後編

6.4.1 ファイルの保存

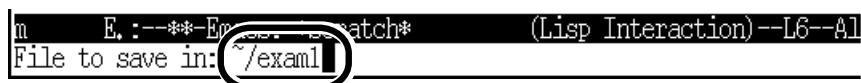
ファイルを保存するには、メニュー・バーの [File] から [Save Buffer] を選択する。または、**C-x C-s** と入力してもよい。



なお、ファイル名を指定しないで Mule を起動している場合、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、ファイル名入力して、**Enter** する。



ファイル名を指定しないで、Mule を起動した場合

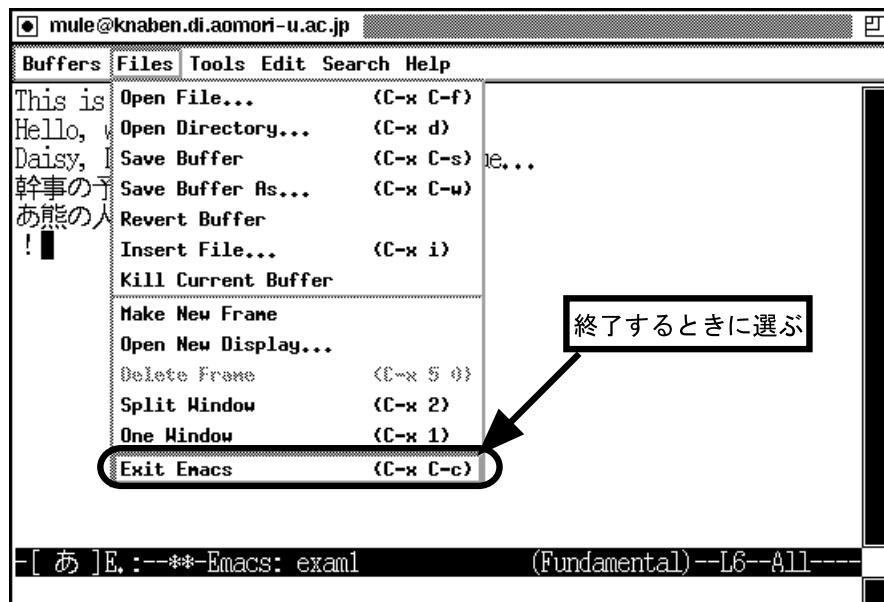


ファイル名を入力して、**Enter**

6.4.2 Mule の終了

[File] 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。



もしも前回セーブしたとき以降、何か変更を加えていると、このときウィンドウが開いて、セーブ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、セーブしないけど本当に終わっていいかと聞かれ、[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あるので、y か n で答える。



- [Yes] でセーブして終了
- [No] でセーブしない。
なお、[No] を選んだ場合には、「本当にセーブしないで終了していいか」と聞かれるので、それには [Yes] と答える。

6.4.3 [File] メニューの項目

ここで、[File] メニューの主な項目を紹介しておこう。

表 6.5: [File] 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 | 今、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 は、ファイルを編集すると、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、1回前の状態のファイルだけは、自動的に残っている。もしも、編集しそこねた場合は、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。

また、ファイルの編集中には、元のファイル名の最初と最後に「#」の付いたファイルを作る。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、セーブしていない場合はそのまま残る。

6.5 Mule の使い方: 応用編

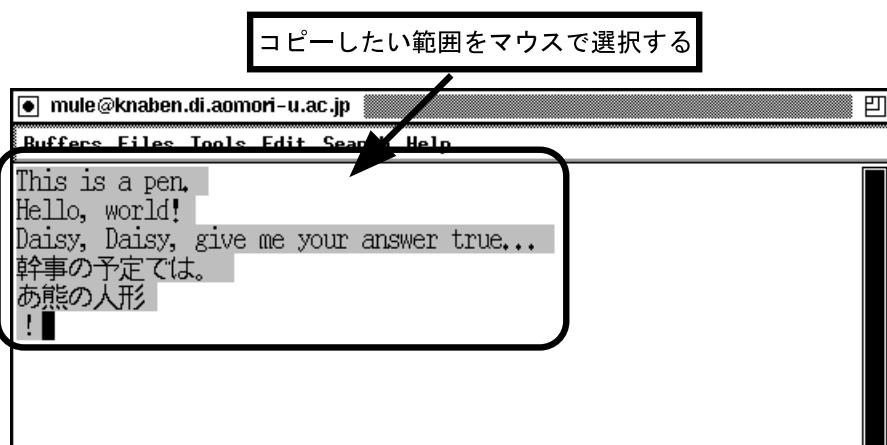
ここでは、知っていると便利な機能を紹介しよう。

6.5.1 部分を選択してコピーして貼り付け

普通、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ある。Mule の場合を紹介しよう。

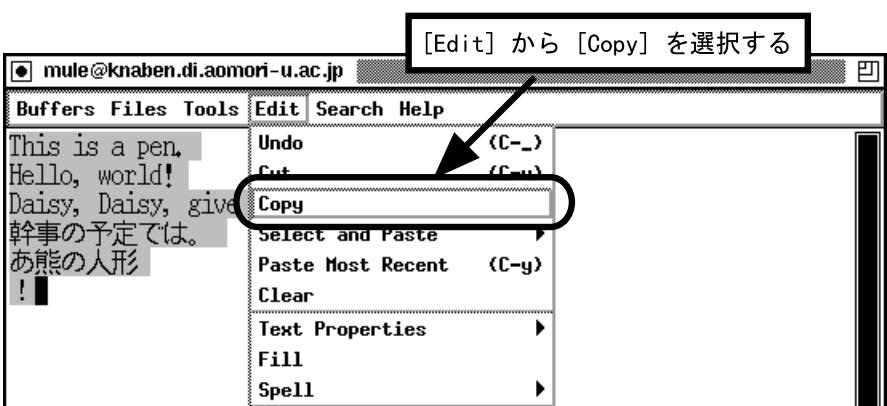
まず、部分を選択するには、次の図のようにマウスでドラッグしてやればよい。

なお、選択する最初の部分にカーソルを移動し、そこで $C-\text{c}$ ないしは $C-$ [スペース] し、選択する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。

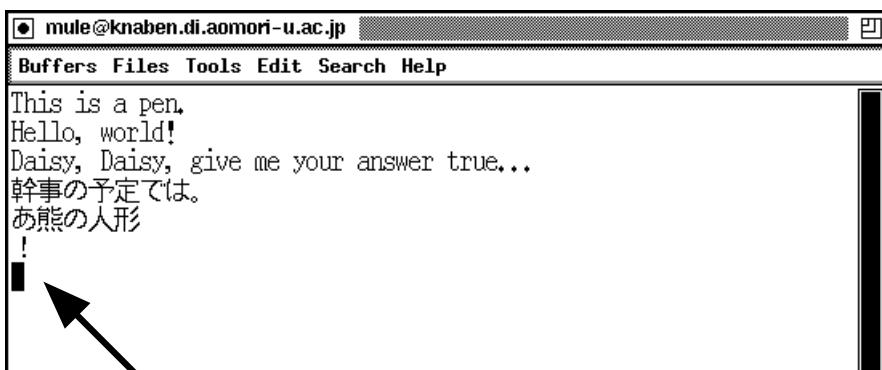


この部分をコピーする(コンピュータに記憶させる)には、メニュー・バーの [Edit] から [Copy] を選ぶ。

または、 $M-w$ と入力してもよい。



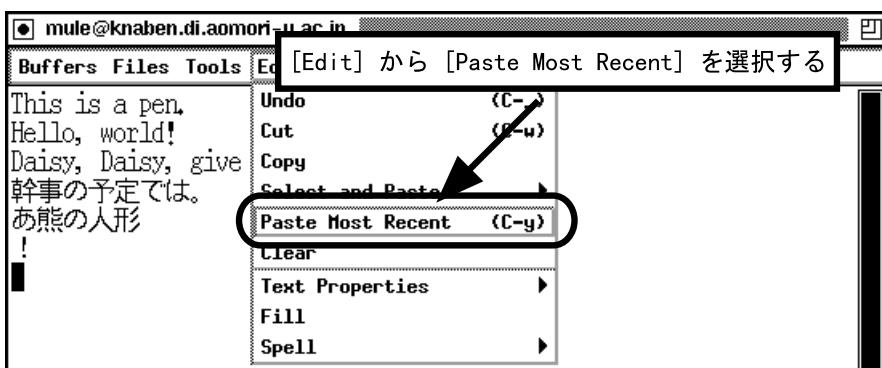
次に、今、記憶させたものを、貼り付けたいところにカーソルを移動する。



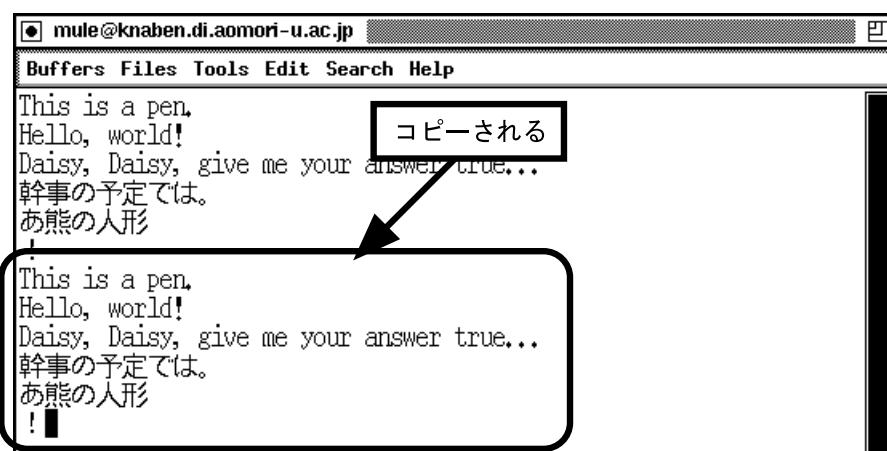
貼り付けたい位置にカーソルを移動

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。



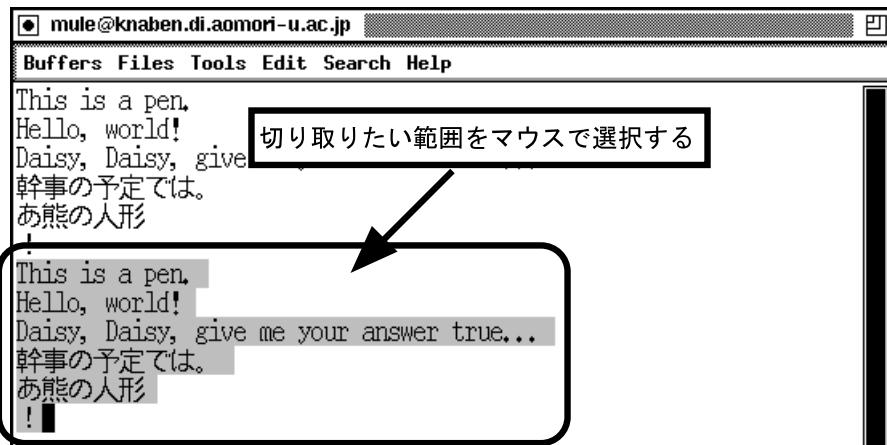
記憶された内容が、貼り付けられる。



6.5.2 部分を選択して切り取り

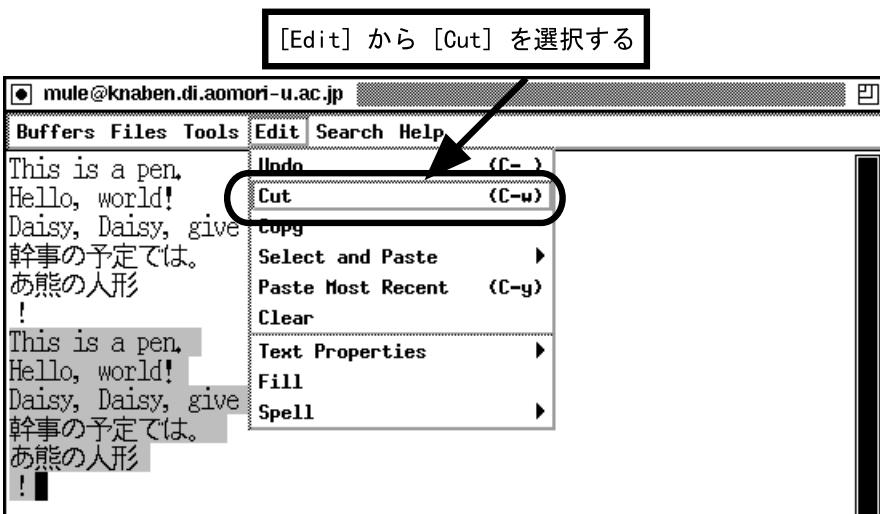
さきほどと同様に、切り取りたい部分を、マウスでドラッグして選択する。

なお、選択する最初の部分にカーソルを移動し、そこで $C-\text{e}$ ないしは $C-$ [スペース] し、選択する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。

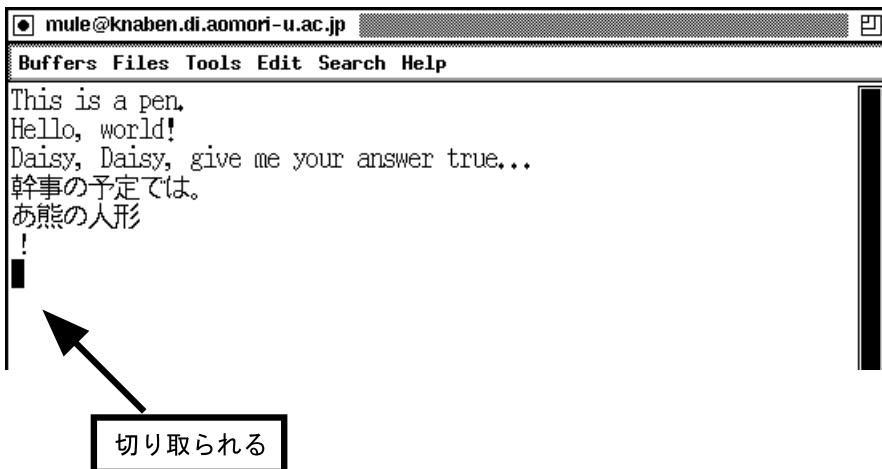


この部分を切り取るには、メニュー・バーの [Edit] から [Cut] を選ぶ。

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。



6.5.3 上下に画面分割

複数のファイルを同時に編集したい場合がある。この場合、いくつも Mule を起動してもよいが、画面を分割して操作することもできる。

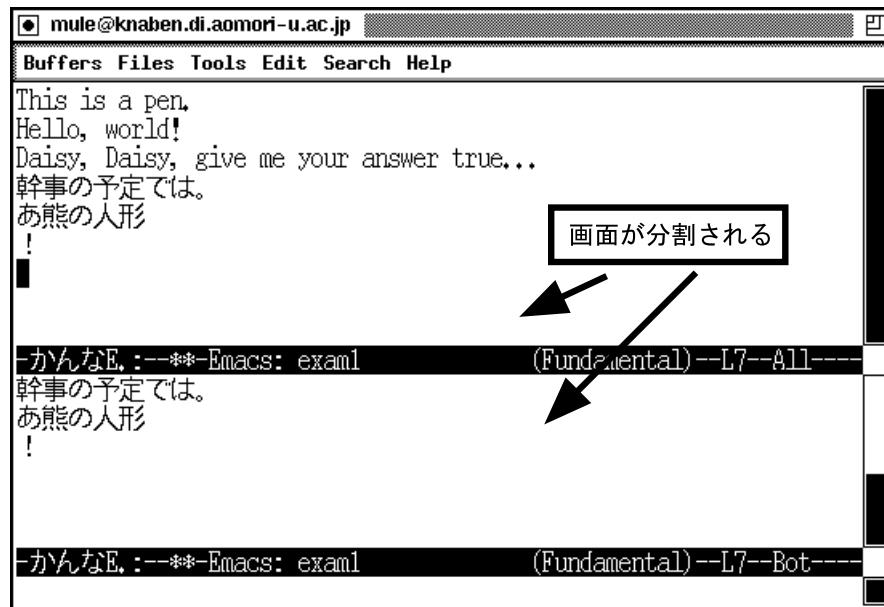
画面を上下に分割するには、スクロール・バーの分割したい位置で、C-ボタン 2 を実行する。または、C-x 2 と入力しても同様の効果がある。



すると、画面が上下に分割される。

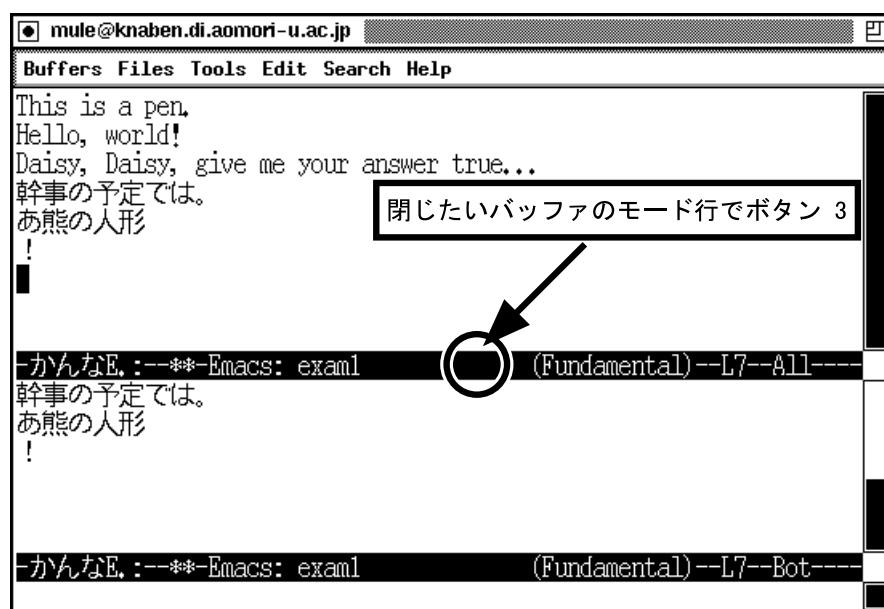
上下の画面は、マウスでクリックすれば、カーソルを行き来させられる。これは、C-o と入力することでも、同様の効果がある。

この画面はそれぞれ、独立に別のファイルを読み込んで編集することもできる。

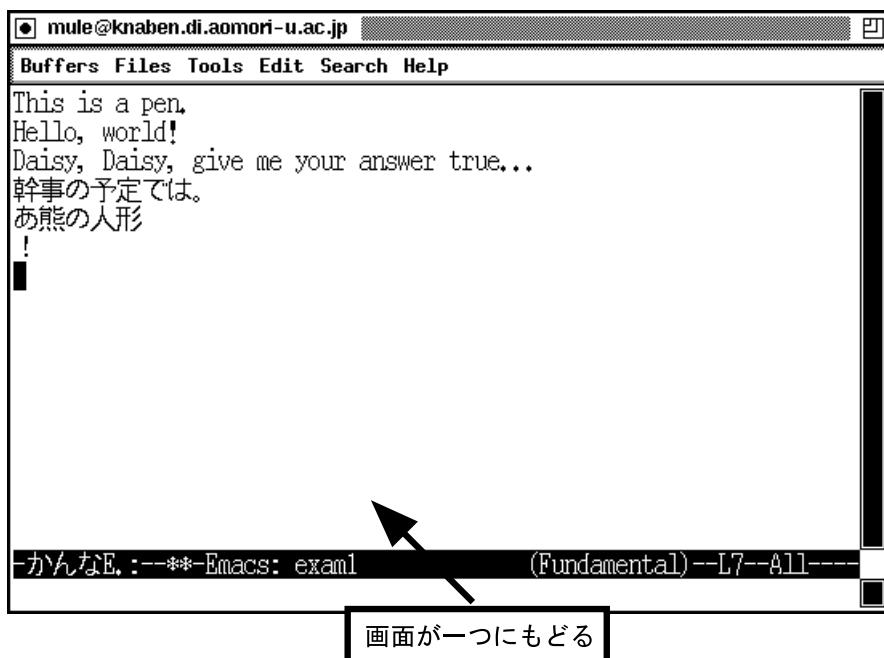


画面を消したいときには、消したい方のバッファのモード行でボタン 3 を押す。

または、残したい方の画面にカーソルを入れて、 C-x 1 と入力しても同様の効果がある。



すると、画面が一つにもどる。

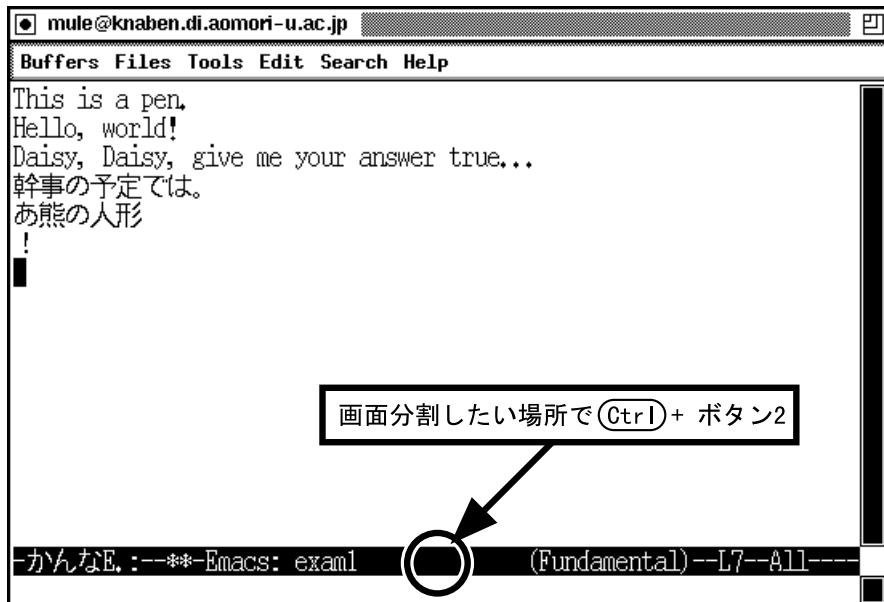


6.5.4 左右に画面分割

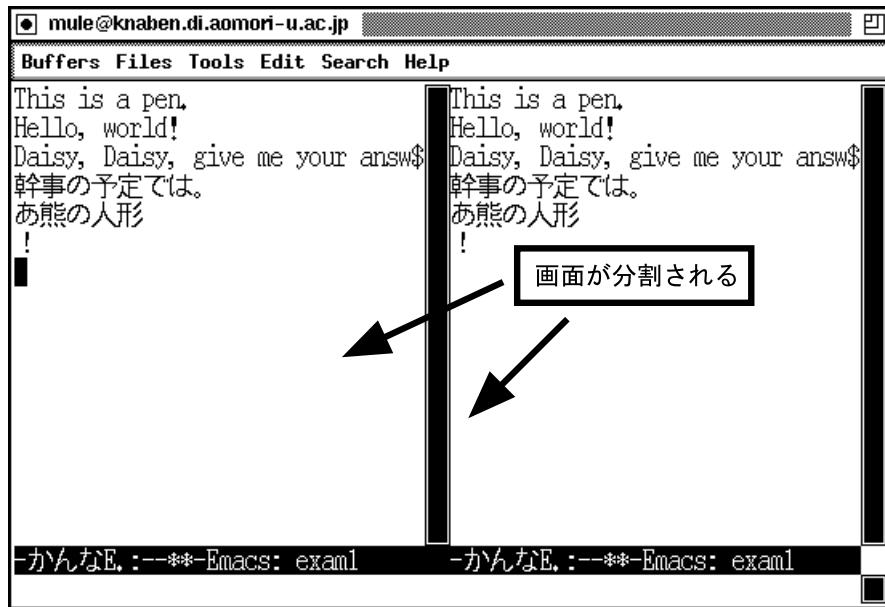
同様に左右に分割することもできる。

左右に分割するには、モード行の分割したい位置で、C-ボタン 2 を実行する。

または、C-x 3 と入力しても同様の効果がある。



左右のカーソルの行き来は、マウスができる。または、C-o でも同様である。

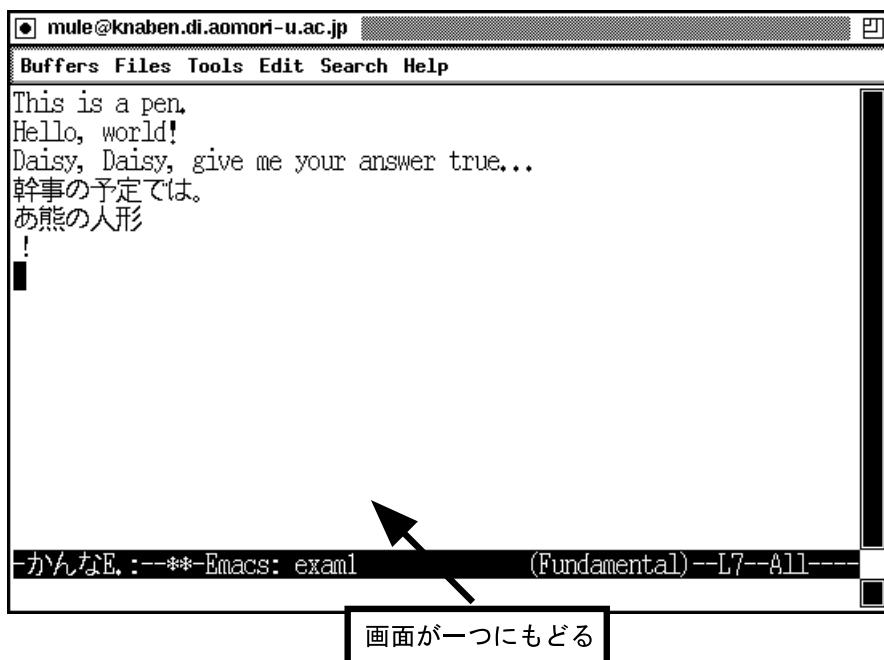


さきほどと同様、画面を消したいときには、消したい方のバッファのモード行でボタン 3 を押す。

または、残したい方の画面にカーソルを入れて、C-x 1 と入力しても同様の効果がある。



すると、画面が一つにもどる。



6.5.5 GNU Emacs 系エディタの機能

今回紹介した Mule も含めて、GNU Emacs 系のエディタは実に様々な機能を持っている¹。その凄さと言えば、極端な話、UNIX システムでは、login 直後に GNU Emacs を起動しておけば、後は GNU Emacs を一度も終了しないで、すべての作業を GNU Emacs の中から行なうことが可能なくらいである。

最後に簡単にここで紹介しなかった機能のうちで、主なものを挙げておこう。

1. 各種プログラミング言語用のモード

例えば、C 言語のファイルを編集するときには、自動的に「C 言語モード」になり、プログラミングをサポートする機能が使える。

2. 強力なマクロ機能

GNU Emacs の中で GNU Emacs の動作そのものがプログラミング可能。こみいといった作業も、使い方次第では簡単な操作で実行することができる。また、自分専用にコマンド登録もできる。

3. 数値計算、電子メール、ニュース、WWW、UNIX のコマンドの実行

GNU Emacs から、コマンドを実行できるだけでなく、専用の便利なツールが多数開発されている。

¹ GNU Emacs が高機能なのは、Emacs Lisp という専用のインタプリタ言語の上で動いているプログラムだからである。

第7章 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ます。

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ます。

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

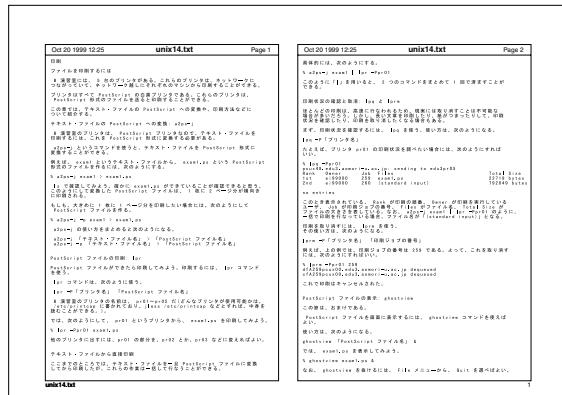
B 演習室のプリンタは、PostScript プリンタなので、テキスト・ファイルを印刷するには、これを PostScript 形式に変換する必要がある。

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 で確認してみよう。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 exam1 > exam1.ps
```

```
Oct 20 1999 12:25          unix14.txt          Page 1
日割
ファイルを印刷するには:
PostScriptには、多くのプリンタがある。これらのプリンタは、ネットワークに
つながる場合に、そのデータを直接送り、そのままのタイミングで印刷することができます。
プリンタはすべてPostScriptの命令プリンタである。これらのプリンタは、
PostScript形式のファイルを送ると自動的に印刷することができます。
PostScript形式のファイルをPostScriptへの変換、印字方法などに
詳しくは、PostScriptへの変換、印字方法などを詳しく見てください。
$ a2ps -j -p exam1 > exam1.ps
1で印刷してみよう。確かにexam1.psができていることが確認できると思う。
この段落はPostScript形式で記述したPostScriptファイルは、1枚に2ページ分が複数書き
出されることがある。
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ファイルを作成。
$ a2ps -j -p exam1 > exam1.ps
exam1.psの使い方をまとめた次のようになる。
exam1.ps → PostScript ファイル名
exam1.ps → 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 ファイルができると印刷してみよう。印刷するには、lpr コマンドを使おう。
lpr コマンドは、次のようだ。
lpr -P「プリンタ名」 「PostScript ファイル名」
PostScriptのプリンタの名前を指定してlpr -P「pr01」 「exam1.ps」 とすると、PostScriptを読み取る
プリンタからexam1.psが印刷される。ただし、PostScriptを読み取るプリンタが存在しない場合は、
lpr -P「pr01」 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・ファイルなら直接印刷
ここまでこの手順では、テキスト・ファイルを一旦PostScriptに変換して行なうことができる。
```

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」
```

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、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」
```

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、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。

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。

7.3 テキスト・ファイルから直接印刷

ここまでのことでは、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 exam1 | lpr -Ppr01
```

このように「|」を用いると、2つのコマンドをまとめて1回で済ますことができる。

7.4 印刷状況の確認と取消: lpq と lprm

ほとんどの印刷は、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、紙がつまつたりして、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。

まず、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   e199000    259  exam1.ps       22719 bytes
2nd   e199000    260  (standard input) 792049 bytes

no entries
```

このとき表示されている、Rank が印刷の順番、Owner が印刷を実行しているユーザ、Job が印刷ジョブの番号、Files がファイル名、Total Size がファイルの大きさを表している。なお、a2ps-j exam1 | lpr -Ppr01 のように、一括で印刷を行なっている場合、ファイル名が「(standard input)」となる。

印刷を取り消すには、lprmを使う。その使い方は、次のようになる。

```
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。よって、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た。

7.5 PostScript ファイルの表示: ghostview

この節は、おまけである。

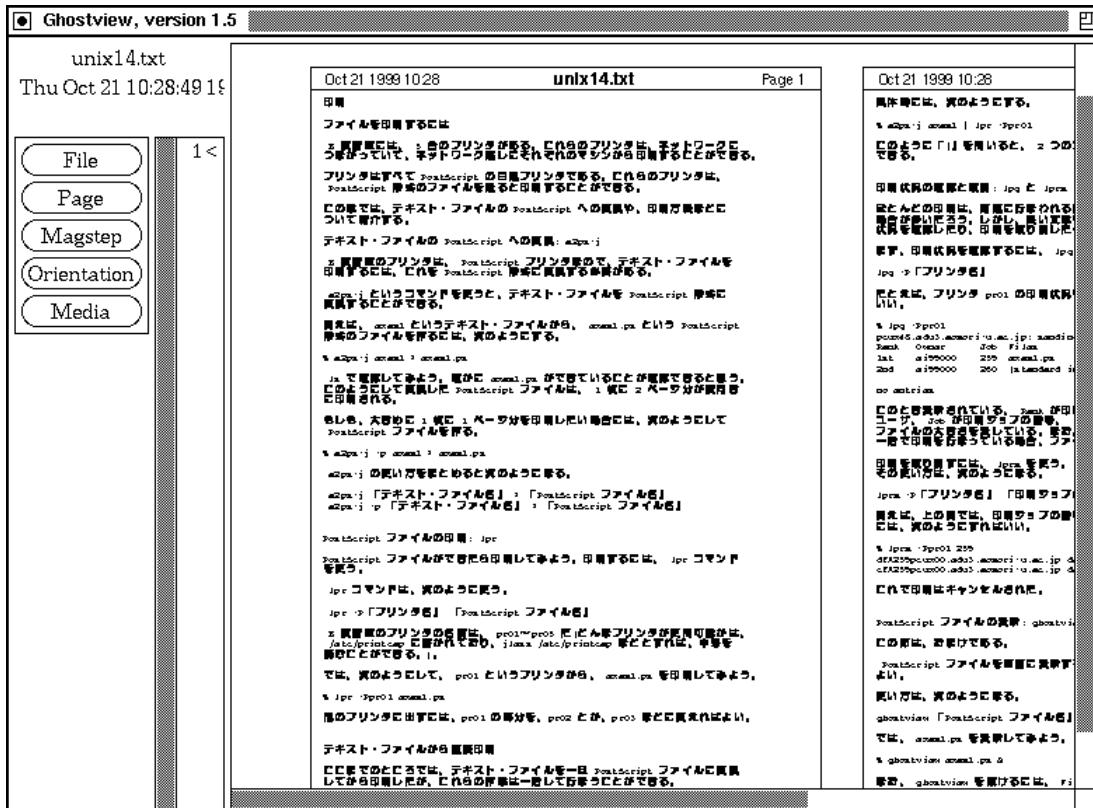
PostScript ファイルを画面に表示するには、`ghostview` コマンドを使えばよい。

使い方は、次のようになる。

```
ghostview 「PostScript ファイル名」 &
```

では、`exam1.ps` を表示してみよう。

```
% ghostview exam1.ps &
```



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

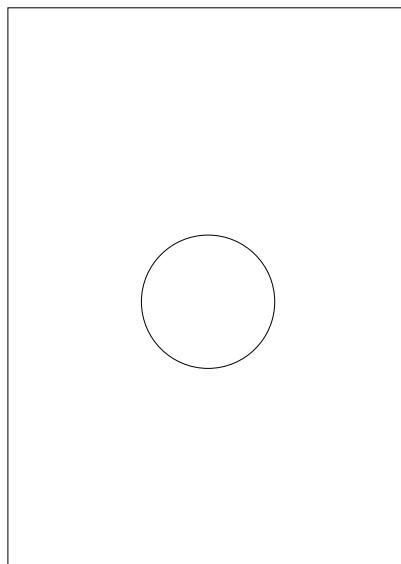
7.6 PostScript ってどんなものか?

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 &`と起動して、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全然かまわないが、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れ、更に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 UNIX システムを真似した 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、これらのことを行なうことができるようになるだろう。

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、とても便利に使える。

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。

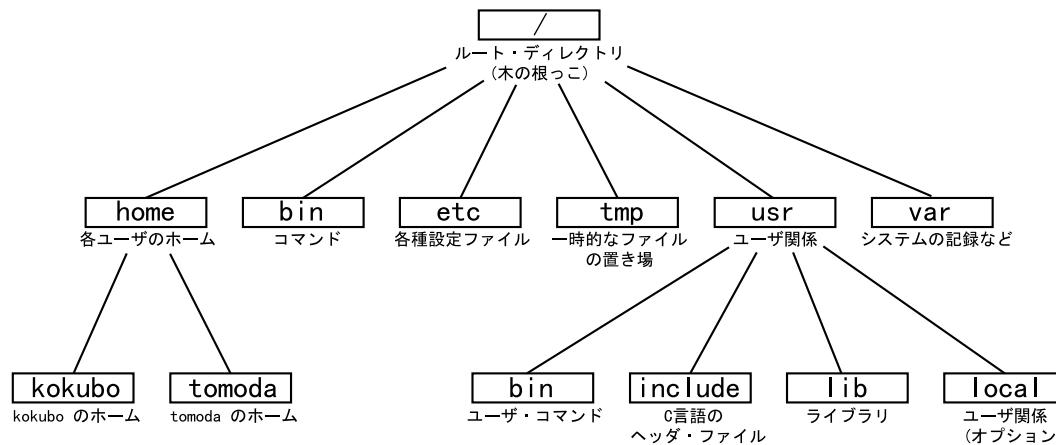


図 8.1: UNIX システムのディレクトリ構造

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、この中に人の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`」だ。そしてこれを、「/home/kokubo」と表す。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。ここまで演習で作ったファイルも、みんなホーム・ディレクトリにあるはずだ。

上の例では、`kokubo` さんは login すると、「/home/kokubo」に入る。そして、ここまで演習で `kokubo` さんの作ったファイルは、ここに置いてあるということだ。

• bin

「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だ。ちなみに、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。なお、UNIX システムでは、コマンドを入れるためにディレクトリに「bin」という名前を付けている。

「/」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が置いてある。

- etc

「etc」は「エトセトラ – etc.」である。UNIX システムでは、ここに各種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ます、「/usr/bin」にはユーザが使う各種コマンド、「/usr/include」には C 言語のヘッダー・ファイルというものが、「/usr/lib」にはユーザ・コマンドが使うライブラリというデータが置かれています。

また、「/usr/local」以下には、オプションのコマンドなどが置かれています。オプションのコマンドは、UNIX の基本システムには付いて来ないものだが、便利なものが多い。例えば、Mule などもオプションのコマンドである。

なお、この図には書いていないが、「/usr/local」の下にも、更に「bin」や「lib」などがある。

- var

「var」は「変化する – variable」という意味。login の記録や、メールや、システムの記録などの、日々変化するものが置かれています。

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だ。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ができる。

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。

図を見てもうとわかるが、「/」の下に「home」があるところまでは標準と一緒にだ。違うのは、この中が更に学年によって「inf8」、「ei97」、「ei98」、「ei99」と切り分けてあるところだ。そして、そ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、ei99001 さんのホーム・ディレクトリは「/home/ei99/ei99001」だ。同様に ei99002 さんは「/home/ei99/ei99002」となる。

そして、この章の後の方で説明するが、ユーザは自分でホーム・ディレクトリの中に「Report」だとか、Sono1 とかのディレクトリを作って、ファイルを整理することができる。

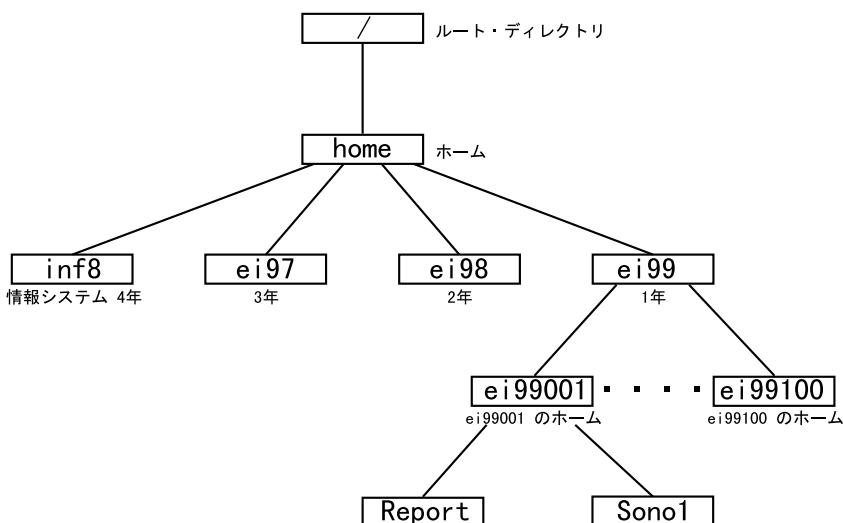


図 8.2: B 演習室の実際のホーム・ディレクトリの構造

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¹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。

¹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。

8.2.1 根っこからの通り道: 絶対パス

既に、ここまで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、順番にディレクトリを書いて表す方法を、「絶対パス」と呼ぶ。

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、「ei99」は絶対パスでは「/home/ ei99」となる。

たとえばその中に、ei99000さんのホーム・ディレクトリがあって、これは「/home/ ei99/ ei99000」だ。

そして、ei99000さんは、自分のホーム・ディレクトリの中に「Report」というディレクトリを作ることができる。これは「/home/ ei99/ ei99000/ Report」と表される。

またその中にディレクトリやファイルを作ることもできる。たとえば「11gatsu」というディレクトリを作ると、これは絶対パスでは「/home/ ei99/ ei99000/ Report/ 11gatsu」になる。

このように根っこ(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。

なお、ここまでディレクトリだけしか説明しなかったが、ファイルの場合も全く同様である。ei99000さんのホーム・ディレクトリ「/home/ ei99/ ei99000」に「ThisMonth」というファイルがある場合、このファイルは絶対パスでは「/home/ ei99/ ei99000/ ThisMonth」と書く。

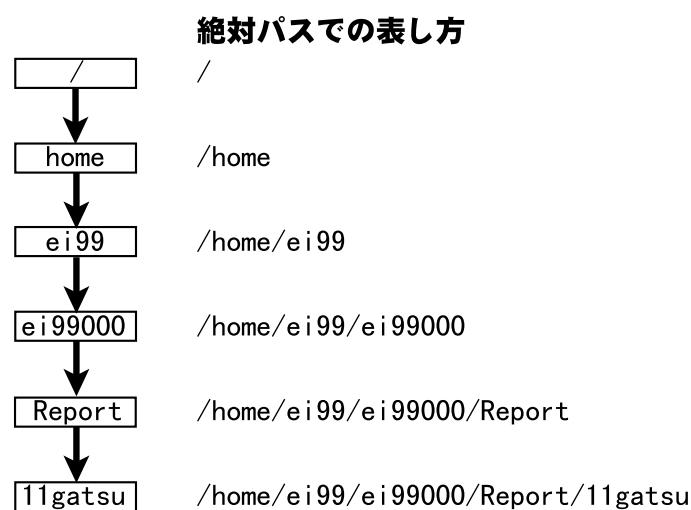


図 8.3: 絶対パスでの表し方

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、順番書いていけば、どんなディレクトリでも表せる。しかし、絶対パスだと、だんだん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面倒だ。

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8.4を見ながら説明しよう。これは、さっき絶対パスの説明に使ったものと全く同じだ。

まず、ei99000さんは、今自分のホーム・ディレクトリにいるとしよう。この下に、「Report」というディレクトリがあるとする。

するとこれは、相対パスでは、単に「Report」と書く。

絶対パスだと、「/home/ ei99/ ei99000/ Report」と長くて面倒だが、相対パスではこんなに短くなる。

それから、「Report」の下の「11gatsu」も、相対パスでは「Report/11gatsu」と、とても簡単に書ける。

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。

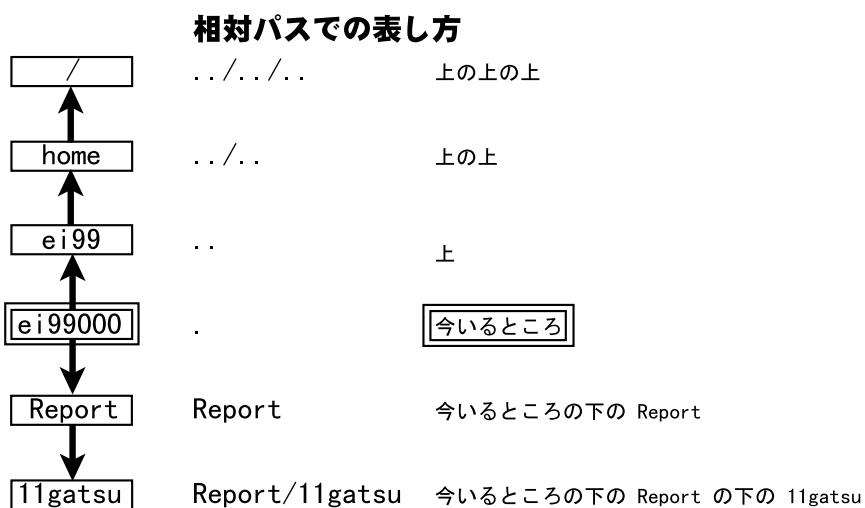


図8.4: 相対パスでの表し方

1つ上のディレクトリ: 「..」

ここまで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。

UNIXシステムでは、1つ上のディレクトリを「..」と表す。

つまり、自分のホームの一つ上の「/home/ei99」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。

自分が今いるディレクトリ：「..」

最後にもう一つだけ紹介しておく。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。

これもよく使うので覚えておこう。

たとえば、今「/home/ei99/ei99000」にいるなら、ここは相対パスで表すと「..」となる。

自分のいる場所によって相対パスは変わる

なお、以上の例は、あくまで「/home/ei99/ei99000」にいる場合の話だ。

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、「/home/ei99/ei99000/Report」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有るので、相対パスでは「..」となる。

また、「11gatsu」は、今いるところのすぐ下にあるので、相対パスでは「11gatsu」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。

相対パスでの表し方

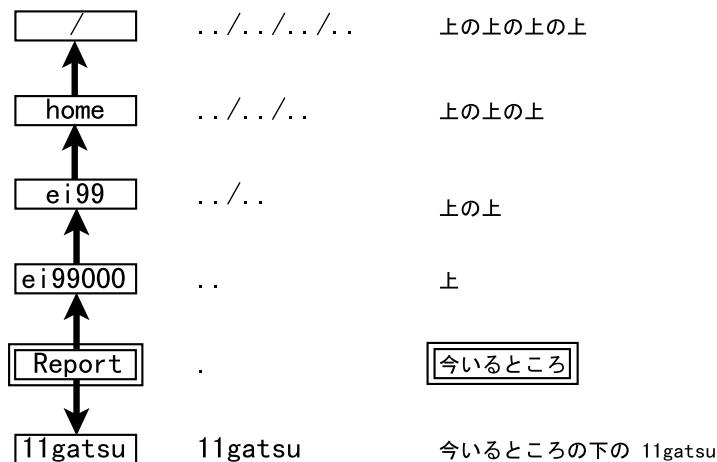
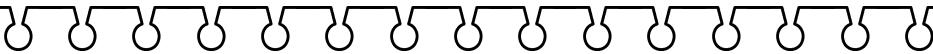


図 8.5: Report の中にいるときの相対パス

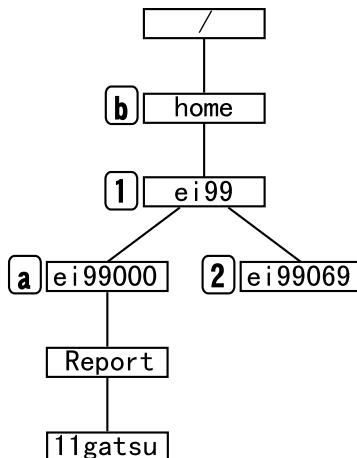
8.2.3 絶対パスと相対パス、どっちがお特?

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、自分が今いるところを指定するには「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」を使った方が簡単だ。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

1. 下の図の [1] の絶対パスは?
2. では、今、あなたは [a] にいる。すると [1] の相対パスは?
3. では、あなたが [b] にいるときの、 [1] の相対パスは?
4. [2] の絶対パスは?
5. では、今あなたが [a] にいるとき、 [2] の相対パスは?
6. では、今あなたが [b] にいるときはの [2] の相対パスは?



答え

1. /home/ei99
2. ..
3. ei99
4. /home/ei99/ei99069
5. ../ei99069
6. ei99/ei99069

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は?: pwd

login した直後にいる場所が、きみのホーム・ディレクトリだと、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「print working directory」の略で、「今お仕事をしているディレクトリを表示してね」という意味だ。

使い方は、単に次のように打てばいい。

pwd

さて今、図 8.6 のようなディレクトリ構造で、ei99000 さんは、「/home/ei99/ei99000」にいるとしよう。このとき、pwd を実行すると、

```
% pwd  
/home/ei99/ei99000  
%
```

となり、今いるのが「/home/ei99/ei99000」だと表示される。

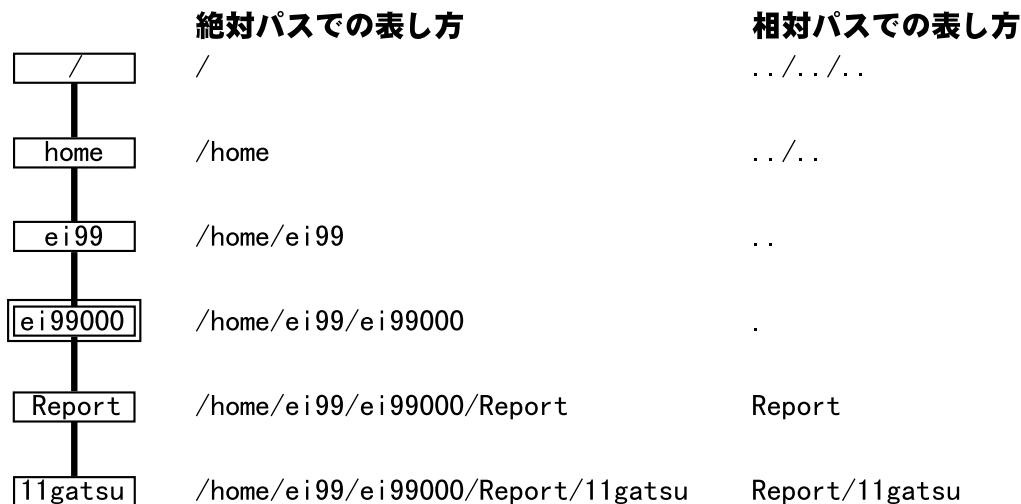


図 8.6: ei99000 さんが「/home/ei99/ei99000」にいるときの絶対パスと相対パス

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「change directory – チェンジ・ディレクトリ」の略だ。使い方は、

```
cd 「パス」
```

である。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。

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、図8.6のように、「/home/ei99/ei99000」としよう。

1つ上のディレクトリは、絶対パスでは「/home/ei99」で、相対パスでは「..」だ。

よって、1つ上に移動するには、「`cd ..`」と、「`cd /home/ei99`」のどちらでもいい。

とは言え、明らかに「`cd ..`」の方が楽なので、こちらを実行しよう。

```
% 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/ei99  
%
```

確かに、「/home/ei99」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、情報システム工学科1年みんなのディレクトリが見える。

```
% ls  
ei99001 ei99006 ei99011 ei99016 ei99021 ei99026 ei99031 ei99036 ei99041  
ei99002 ei99007 ei99012 ei99017 ei99022 ei99027 ei99032 ei99037 ei99042  
ei99003 ei99008 ei99013 ei99018 ei99023 ei99028 ei99033 ei99038 ei99043  
ei99004 ei99009 ei99014 ei99019 ei99024 ei99029 ei99034 ei99039 ei99044  
ei99005 ei99010 ei99015 ei99020 ei99025 ei99030 ei99035 ei99040 ei99100  
%
```

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後には / 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
ei99001/ ei99010/ ei99019/ ei99028/ ei99037/
ei99002/ ei99011/ ei99020/ ei99029/ ei99038/
ei99003/ ei99012/ ei99021/ ei99030/ ei99039/
ei99004/ ei99013/ ei99022/ ei99031/ ei99040/
ei99005/ ei99014/ ei99023/ ei99032/ ei99041/
ei99006/ ei99015/ ei99024/ ei99033/ ei99042/
ei99007/ ei99016/ ei99025/ ei99034/ ei99043/
ei99008/ ei99017/ ei99026/ ei99035/ ei99044/
ei99009/ ei99018/ ei99027/ ei99036/ ei99100/
%
```

`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。どんなオプションがあるかはオンライン・マニュアルに載っている。

一気に絶対パスで移動

次は、絶対パスを使って、一気に「`/bin`」に移動してみよう。

「`/bin`」は、UNIXの最も基本的なコマンドの置いてあるディレクトリである。

では「`cd /bin`」と打とう。

```
% 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
```

なお、ここで `ls -F` すると、次のようになる。ここで、*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*
chios*      domainname*  ls*           red*          sync*
chmod*      echo*        mkdir*         rm*           test*
cp*         ed*          mv*           rmail*        
csh*        expr*        pax*          rmdir*        
date*       hostname*   ps*           sh*
```

「`cd`」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。

じゃ、どっち使えばいいのかというと、「自分のいるところ近くなら、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・ディレクトリに戻ることにしよう。

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/ei99/ei99000
%
```

確かに、ホーム・ディレクトリに戻れたことがわかる。

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「make directory - メイク・ディレクトリ」で、そのままの意味だ。

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「Enshu」とかいうディレクトリを作ってみよう。

```
% mkdir Enshu  
%
```

では、`ls` で確認してみよう。

```
% ls  
1998nen 2000nen RMAIL ThisYear exam1  
1999nen Enshu ThisMonth Years  
%  
%
```

これでは、「Enshu」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  
1998nen 2000nen RMAIL ThisYear exam1  
1999nen Enshu/ ThisMonth Years  
%  
%
```

たしかに、「Enshu」というディレクトリがあるのがわかる。

では、「Enshu」の中に入ってみよう。これは、絶対パスなら「/home/ei99/ei99000/Enshu」で、相対パスでは「Enshu」なので、相対パスの方が簡単だ。よって、「cd Enshu」と打とう。

```
% cd Enshu  
%
```

では、`pwd` で確認してみよう。

```
% pwd  
/home/ei99/ei99000/Enshu  
%
```

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「remove directory – ディレクトリ削除」の略だ。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/ei99/ei99000  
%
```

では、次のようにして、さっそく作った「Enshu」を消してみよう。

```
% rmdir Enshu  
%
```

これで、消えたはずだ。`ls -F` で確認してみよう。

```
% ls -F  
1998nen      2000nen      ThisMonth      Years  
1999nen      RMAIL        ThisYear       exam1  
%
```

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 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。

たとえば、「CAL」というディレクトリを新しく作って、「ThisMonth」というファイルをその中に移動するには次のようにする。

まず、「CAL」というディレクトリがあるかどうか確認する。

```
% ls -F  
1998nen      2000nen      ThisMonth      Years  
1999nen      RMAIL        ThisYear       exam1  
%
```

すると、上のようにないことがわかる。そこで「CAL」を mkdir で作る。

```
% mkdir CAL  
%
```

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全く同じだ。ただ、ファイル名として、パスを書いてやればいい。

つまり、ThisMonth を CAL に移動するには、「mv ThisMonth CAL/ThisMonth」となる。

ちなみに、これは「mv ThisMonth CAL/.」とも書ける。

「.」をファイル名の代わりに使うと、「同じ名前で」という意味になるのだ。

このように「.」は、ディレクトリのときには「現在の場所」、ファイル名のときには「同じ名前で」という意味になるので、混同しないように注意しよう。

では、実際にやってみる。

```
% mv ThisMonth CAL/.  
%
```

ls -F で確かめてみると、たしかに「ThisMonth」がないことがわかる。

```
% ls -F  
1998nen      2000nen      RMAIL        Years  
1999nen      CAL/        ThisYear     exam1  
%
```

「CAL」に入って中を見て見ると、「ThisMonth」があることがわかる。

```
% cd CAL  
% ls -F  
ThisMonth  
%
```

なお、この説明では mv を使ってファイルを移動したが、代わりに cp を使えば、コピーになる。

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`

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」`

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、同時にいくつものプログラムを実行できる。これは、ブラウザ、ゲーム、エディタなどを同時に起動して、使うことができるという意味だ。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、本当に最近のことだ。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。

30年くらい前、UNIXが開発された頃になって、やっとタイム・シェアリング¹という方法で、同時にいくつものプログラムを走らせるができるようになった。これによって、ウィンドウをいくつも開いたり、何人もの人が同じコンピュータを同時に使えるようになった。

ちなみに、走っているプログラムのことを、プロセスと呼ぶ。同時にプログラムがいくつも実行できるUNIXのようなシステムでは、同時にプロセスがいつも走っている。

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。

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、同時にいくつものことができる。

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か見てみよう。今、走っているプロセスはps (Process Status: プロセスの状態) で見ることができる。では、単にpsと打って、実行してみよう。

¹Time 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、次々に少しづつ実行するという方法。

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

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」` とすればいい。
では、実際にやってみよう。たとえば、ei99000 さんが、自分のプロセスを全部表示するには、`ps -U ei99000` だ。

```
% ps -U ei99000
 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 ei99000
%
```

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、システムがプロセスを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) というのがそれだ。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。

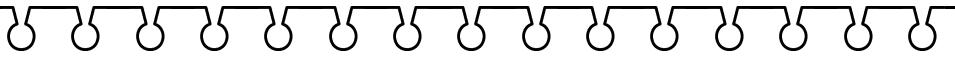
次の列の TT は、端末番号という。実は、`kterm` のように、コマンドを打てるウィンドウを開くと、システムがウィンドウごとに番号を割り振る。それが端末番号で、これを見ると、どのウィンドウから実行したコマンドかがわかる³。

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、コマンドの名前だ。

²少し難しい話なので、こここの部分はわからなくてもよい。一部しか表示されないことだけ知っていれば、とりあえず問題ない。なお、表示されていないのは、端末から実行されていないプロセスである。

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。



練習

1. まず、 X Window System で、いくつかのプログラムを起動しよう。そして、 ps を実行して、どんなプロセスが走っているか実際に見てみよう。

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。

ためしにマシンの中で動いているプロセスを全部表示してみよう。それには、`ps -aux` と打つ。これはたくさん出るので、次のようにページャー `less` にパイプしてやるといい。

```
% 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 |
|---------|-----|------|------|------|------|----|------|---------|---------|----------------|
| ei99000 | 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 -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 17 個の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、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 ei99000  28   0  652K   860K RUN     0:00  0.40%  0.04% top  
 279 ei99000  18   0  680K   984K pause   0:00  0.00%  0.00% tcsh  
 273 ei99000  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 ei99000  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 ei99000   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た。例えば **[Ctrl]+c** とか、 X Window System なら **twm** のメニューから **[delete]** などだ。

ここでは新しく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、プロセスを殺せばいい。プロセスを殺すには **kill** を使う。使い方は、

kill 「PID」

である。

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 を知らないといけないので、次のように **ps** で **xclock** の PID を調べる。

```
% ps -U ei99000
 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 ei99000
%
```

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

```
% kill 301
%
```

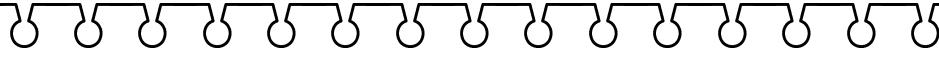
すると、**xclock** が消えたはずだ。なお、**kill** をすると、**[1] Done xclock** のようなメッセージが出ることがある。

では、念ため **ps** で確認してみよう。

```
% ps -U ei99000
 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 ei99000
```

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` という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 ジョブのコントロール

9.2.1 フォアグランドとバックグランドのジョブ

では、次にジョブについて紹介しよう。

ジョブというのは、まあ、だいたいプロセスのようなものと思ってもらえばいい⁴ジョブには、フォアグランド (foreground: 表) のものと、バックグランド (background: 裏) のものがある。

まず、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だ。実は、それぞれの `kterm` の画面では、フォアグランド・ジョブは、同時には一つしか走らない。つまり、普通に `ls` とか打った場合、`ls` が終わるまで、別のコマンドは実行できないのだ。

一方、バックグランド・ジョブは、いくつも走らせることができる。なお、コマンドをバックグランドで走らせるには、コマンドの最後に `&` を付けるだけでいい。

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だ。新しくウィンドウを開くときは、普通、`&` を付けて、いつもバックグランドで開いていたのだ。

少し説明がわかりにくかったかもしれないが、実際にやってみよう。

まず、Mule をフォアグランド (表) で走らせてみよう。単に `mule` と打とう。

```
% mule
```

こうすると、Mule が起動する。

Mule の方は、キーボードから値を入れたり、メニューを開いたりできる。しかし、Mule を起動した `kterm` の方は、`ls` とか打っても、実行されないはずだ。

これは、今、Mule がフォアグランドのジョブになっているということだ。フォアグランドのジョブは同時に 1 個しか走らないので、`ls` とか打っても実行されないのだ。

⁴ 実際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 つだが、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 まとまりのことだ。

なお、Mule の方を終了すれば、その間に kterm の方で打ったコマンドが次々に実行される。

では、次にバックグラウンドで Mule を起動してみよう。バックグラウンドで動かすには、`mule &` と、最後に `&` を付ける。

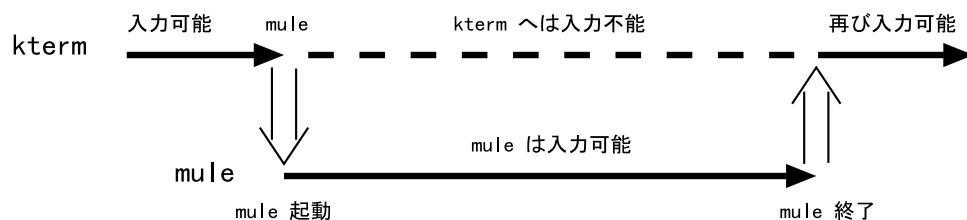
```
% mule &
[1] 544
%
```

今度は、Mule を起動した kterm の画面にも、コマンドが打てるはずだ。

こうして、X Window System でウィンドウを開く時には、バックグラウンドにしておけば、いくつも同時にコマンドを実行できる。

ちなみに、X Window System 以外のコマンドも `&` を最後に付けると、バックグラウンドで実行できる。これは、時間のかかるコマンドを実行させるときに便利だ。

フォアグラウンドで mule 起動



バックグラウンドで mule 起動



フォアグラウンドをバックグラウンドに変更

では、一度、フォアグラウンドで動かしてしまったジョブを、バックグラウンドにする方法を紹介しておこう。この話は少し難しいので、飛ばして読んでもかまわない。

たとえば、kterm から Mule をフォアグラウンドで実行してしまったが、途中でコマンドも実行したくなった場合、Mule をバックグラウンドに変更することができる。

実際にやってみよう。まず、Mule をフォアグラウンドで実行する。

```
% mule
```

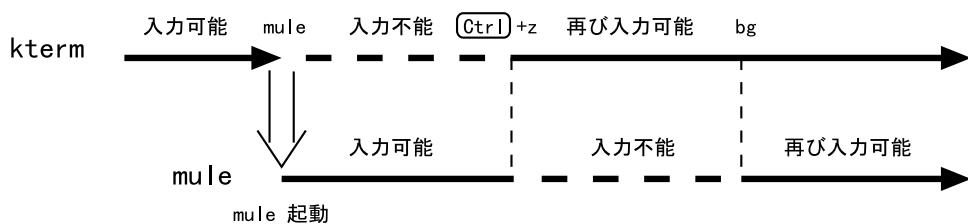
ここで、`kterm` の画面の方で、`[Ctrl]+z` を入力すると、ジョブを一時停止（サスPEND）と言う。終了ではないことに注意）できる。

```
% mule
^Z
Suspended
%
```

今、Mule は、一時停止状態で、何も入力できなくなっている。この状態で、`bg`（バックグランドの略）と打つと、一時停止している Mule のジョブをバックグラウンドに変更して、再び入力できるようになる。

```
% bg
[1]  mule &
%
```

フォアグラウンドをバックグラウンドへ変更



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」`

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 超入門: C のプログラミング

C 言語のテキストの古典中の古典（個人的には、初心者にはあまりすすめない）と言えば、ブライアン・カーニハンと、UNIX を開発したデニス・リッチャーが共同で書いた、「プログラミング言語 C」（石田 晴久訳 共立出版）だろう。

10.1.1 ソース・コードの作成

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` と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 言語モードで起動する¹。なお、C 言語モードでは、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²。

`hello.c`の中には、次のようなプログラムを書き、セーブして Mule を終了する。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。³。

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

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

これが、C 言語で書いたプログラムで、ソース・コード⁴という。

C 言語ははじめての人も多いだろう。そういう人は、このプログラムの意味はわからなくても OK で、なんとなくこんなものだという雰囲気を知っておいてくれればいい。

10.1.2 コンパイル

C 言語で書かれたソース・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」という意味だ。

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

うまくいくと、何もメッセージがない。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。

¹ ファイル名に「.c」が付いていると、自動的に C 言語モードになる。

² 実際には、プログラムの作成、コンパイル、実行まで行なえる。

³ 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 言語のプログラムを作ることは、どちらでも一緒である。

⁴ source code: 「(プログラムの)元になるコード」という意味。

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  
1997nen      1999nen      hello*      nsmail/  
1998nen      Enshu/       hello.c  
%
```

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、できた `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。すると次のように、`hello, world` と表示される。

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

なお、「`./`」は、相対パスで「現在いるディレクトリ」を表している。この話は、少し難しいので、わからなくてもいい。

基本的には、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 言語でプログラムを書き、ソース・コードをコンパイルして、作られている。

なお、C 言語以外にも、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。

パーミッ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。

現在いるディレクトリ中の
全ファイルの大きさ

| <code>% ls -l</code> | | | | | | | |
|----------------------|---|---------|------|------|--------------|---------|--|
| total 18 | | | | | | | |
| -rw-r--r-- | 1 | ei99000 | ei99 | 1969 | Oct 1 10:24 | 1998nen | |
| -rw-r--r-- | 1 | ei99000 | ei99 | 1961 | Oct 7 19:24 | 1999nen | |
| -rw-r--r-- | 1 | ei99000 | ei99 | 1973 | Oct 7 19:25 | 2000nen | |
| drwxr-xr-x | 2 | ei99000 | ei99 | 512 | Oct 21 20:20 | Enshu | |
| -rwxr-xr-x | 1 | ei99000 | ei99 | 8808 | Oct 23 11:12 | hello | |
| -rw-r--r-- | 1 | ei99000 | ei99 | 59 | Oct 23 11:10 | hello.c | |
| drwx----- | 2 | ei99000 | ei99 | 512 | Oct 1 11:12 | nsmail | |

↑ パーミッション
 ↑ リンク数
 ↑ ファイルの持ち主
 ↑ グループ
 ↑ ファイルの大きさ
 ↑ ファイルを最後に書き直した時刻
 ↑ ファイル名

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: read)」「書ける (w: write)」「実行できる (x: execute)」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、`1998nen` 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。

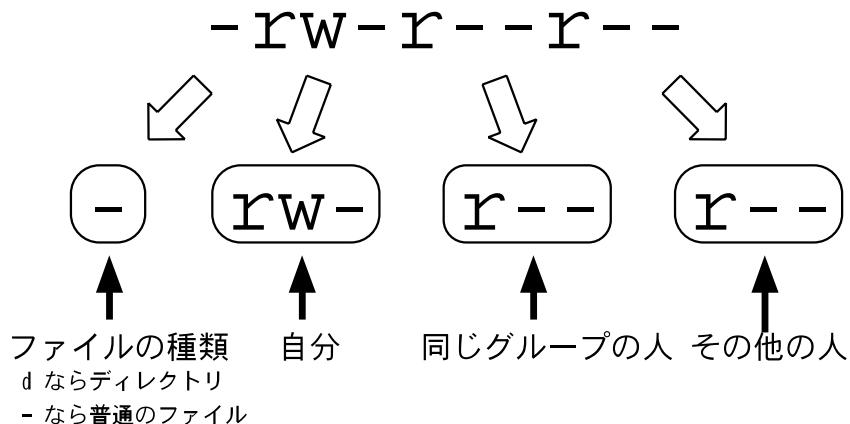


図 10.2: パーミッションの見方

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だ。つまり、例えば「-rw-r--r--」ならファイル、「drwxr-xr-x」ならディレクトリである。

で、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。

| | | | | |
|----------|----------|----------|-----|--------------------|
| r | w | x | --- | 読めない、書けない、実行できない |
| 読める | 書ける | 実行できる | r-- | 読めるけど、書けないし、実行できない |
| | | | rw- | 読んで、書けるが、実行できない |
| | | | rwx | 読んで、書けて、実行できる |

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。

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、「change mode – 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、図 10.4 を見て欲しい。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、2 進数の場合、下の方のケタから順番に、「1 の位」、「2 の位」、「4 の位」になっている。だから「110」は、「4 の位」と「2 の位」が 1 なので、 $4 + 2 = 6$ となって、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。

| パーミッション | 2 進数 | モード (8 進数) |
|------------------|-------------------------|---------------|
| --- | 000 | 0 |
| <code>r--</code> | 100 | 4 |
| <code>rw-</code> | 110 | 6 |
| <code>r-x</code> | 101 | 5 |
| <code>rwx</code> | 111 | 7 |
| | 4 2 1 の の の 位 位 位 | |

図 10.4: パーミッションからモードを求める方法

| | |
|---|---|
| <code>rW-----</code> | 自分だけが読んで書ける (秘密のファイル) |
|  モード 600 | |
| <code>rW-r--r--</code> | 誰でも読めるが、書けるのは自分だけ (秘密ではないファイル) |
|  モード 644 | |
| <code>rWX-----</code> | 自分だけが読んで、書いて、実行できる (秘密の実行ファイルやディレクトリ) |
|  モード 700 | |
| <code>rwxr-xr-x</code> | 誰でも実行できるが、書けるのは自分だけ (秘密ではないファイルや、ディレクトリ) |
|  モード 755 | |

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に挑戦してみよう。

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 ei99000 ei99 8808 Dec  8 16:24 hello
-rw-r--r-- 1 ei99000 ei99      59 Dec  8 16:24 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。

では、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。自分が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)。だから、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 ei99000 ei99 59 Dec 8 16:24 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`」だ。というわけで、次のように実行する。

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

`ls -l` で確認すると。

```
% ls -l  
[中略]  
----- 1 ei99000 ei99 59 Dec 8 16:24 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。

J :--%%-Mule: hello.c
File exists, but cannot be read.

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 ei99000 ei99 59 Dec 8 16:24 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、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。

The screenshot shows a terminal window titled 'mule@castor.di.aomori-u.ac.jp'. The window contains a C code editor with the following content:

```
#include <stdio.h>

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

Below the code, a message is displayed in the minibuffer:

J:--%%-Mule: hello.c (C)--L1--All-----
Note: file is write protected

ここに、何か文字を入れようとするとき、下の図のようにミニ・バッファに「Buffer is read-only: #<buffer hello.c>」と表示されて、結局文字を打てないはずだ。

```
J. :--%%-Mule: hello.c          (C)-  
Buffer is read-only: #<buffer hello.c>
```

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 ei99000 ei99      59 Dec  8 16:24 hello.c  
[中略]  
%
```

次に `hello` という実行型ファイルの方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になっている。

```
% ls -l  
[中略]  
-rwxr-xr-x 1 ei99000 ei99      59 Dec  8 16:24 hello.c  
[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、誰でも読めるが、書けるのは自分だけで、実行は誰もできないように変更してみよう。つまり、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 ei99000 ei99 8808 Dec 8 16:24 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 ei99000 ei99 8808 Dec 8 16:24 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)」`

第11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。

11.1 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・インターフェース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 GUI² をベースにしたシステムだ。

シェルは、コマンド・ライン・ベースのインター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・ベースのインターフェースと 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 種類のインターフェースには、それぞれ特徴がある。

11.1.1 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、「なんなく」使い方がわかり、とっつきやすい。このシステムは、ブラウザとか、お絵描きソフトとか、ゲームとかを、単に起動する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・インターフェース

11.1.2 コマンド・ライン・ベースのインターフェースの特徴

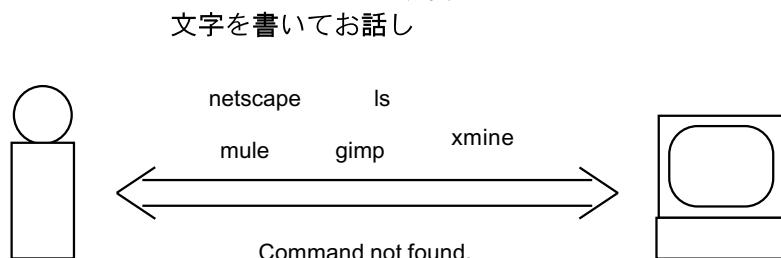
一方、コマンド・ライン・ベースのシステムの方は、文字を書いてコンピュータとやりとりをしているようなイメージになる。

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しできないため、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ぬので、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。

GUI の場合



コマンド・ライン・ベースの場合



11.1.3 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？

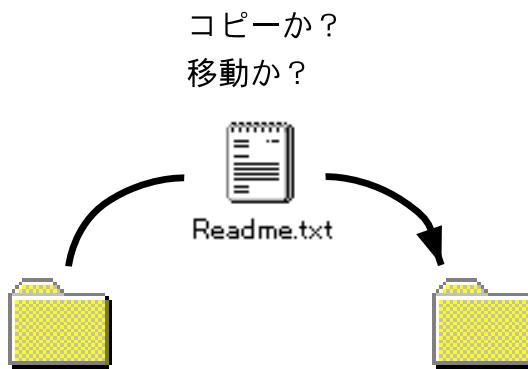
GUI にも問題はある。まず、コンピュータに細かい指示を出すには向いていないことだ。

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

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

の場合でも異なる⁴。



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⁵。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自動化するのにも向いていないのである。

11.1.4 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。

ピー」を実行する。

⁴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実際に仕事では、こういうことはよくある。

11.2 シェルの役割

シェルは、UNIX システムのコアに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）。

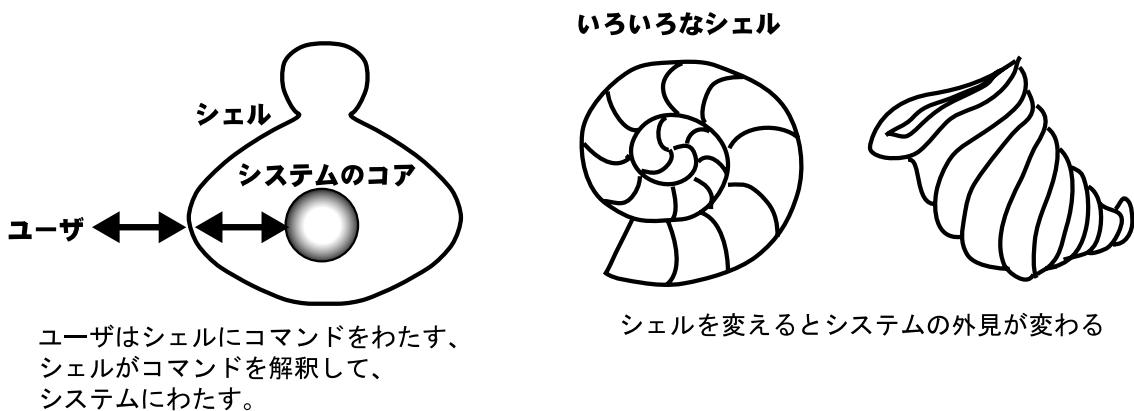
なぜ、そんな余計なものが間に入っているんだろう？

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？

そんなときに、コマンドをダイレクトにシステムに渡している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。でも、ユーザとシステムの間にシェルでワン・クッシュン入れれば、システムを直接改造しなくても、シェル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。

また、間にシェルが入ってくれれば、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できるというわけだ。

要するに システムは単純に、後はシェルにおまかせモード というコンセプトだ。



11.3 いろいろな貝殻

では、シェルを紹介していこう。

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

一時的に使ってみた場合には、そのまま sh とか csh とか打てばいい。

なお、きみたちの場合、何もいじっていなければ、tcsh になっている。

ここでは、それぞれを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。
- ksh コーン (Korn) さんの作った sh の機能拡張版。AT & T 社の製品で、フリー・ソフトではない。ただし、ksh 互換のフリー・ソフトも作られていて、FreeBSD にはこっちが入っている。
- 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 などの機能を取り入れている。もちろん、フリー・ソフト。
- 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。

11.4 シェルの機能

11.4.1 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だ。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。`w`、`ls`、`cal`、`oneko &`などと 4 つくらいコマンドを打ってみよう。ここまで打ったコマンドのリストは、`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きる。

```
% history
[中略]
 101 18:23   w
 102 18:23   ls
 103 18:23   cal
 104 18:24   oneko &
 105 18:25   history
%
```

この表示は、最初が「何番目」か、次が「何時」か、最後が「どんなコマンド」を打ったかを表している。なお、過去 100 個分を記憶しているために、ここでは `login` して最初に打った `w` 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` と打てばいい。全く同様に、103 番目の `cal` なら `!103` だ。

では、試しに打ってみよう。

```
% !101
w
 6:25PM up 12 mins, 2 users, load averages: 0.30, 0.46, 0.29
USER     TTY FROM           LOGIN@ IDLE WHAT
ei99000  p1 :0.0          6:13PM    - w
% !103
 12月 1999
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
%
```

と、このようにとても簡単だ。

! 「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` でいい。もちろん、`history` を呼ぶには単に `!h` だ。では、打ってみよう。

```
% !c
cal
 12月 1999
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
%
```

ちなみに、これは、プログラム書きながら、コンパイルするときに使うと便利だ。つまり、プログラムを書くには「`mule` 「ファイル名」 &」で、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。だから、もう一回編集したければ、単に `!m` でいいし、コンパイルしなおしたければ `!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

最後に直前のコマンドをくり返す方法だが、これは `!!` だ。

もし、直前に `cal` を実行していれば、`!!` で `cal` が実行される。

```
% !!
cal
 12月 1999
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
%
```

11.4.2 コマンド・ライン編集機能

`tcsh` などの場合、これら以外に次のような機能もある。

まず、`[Delete]` (または Mule と同様に C-p) を押していくと、一つずつ前のヒストリーを呼び出すことができる。また、`[Delete]` (C-n) で、一つ先のヒストリーを呼び出せる。

しかも、呼び出したヒストリーは、`[Delete]`(C-b) や、`[Delete]`(C-f) や、`[Delete]` 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。

11.4.3 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。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 `*` を付けて表示させるには、`ls -F` と打てばいい。

しかし、それを毎回打つのは面倒くさい。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'` としてやるとよい。これで `ls` と打つと、`ls -F` が実行されるようになる。

では、実際にやってみる。まず、`ls` と打ってみよう。

```
% ls
1998nen      2000nen      hello        nsmail
1999nen      Enshu        hello.c
%
```

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。ここで、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
1998nen      2000nen      hello*
1999nen      Enshu/
%
```

確かに `ls -F` と打ったのと同じだ。

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だ。

なお、`alias cp rm` などのようなアブナイことも、もちろんできてしまうので、使うときは注意しよう（こうすると、`cp` と打つと `rm` になってしまう。）。

ちなみに、エイリアスは、そのシェルを終了するまで有効である。

11.4.4 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]` を押すと補完する機能である。

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` してみよう。

```
% ls
1998nen      2000nen      hello*
1999nen      Enshu/      hello.c
%
```

するとこの中で、`E` で始まるのは、`Enshu` というディレクトリしかない。この場合、`cd` で `Enshuu` の中に移動するには、`cd Enshuu` と打てばいいのだが、このコマンドを全部打つ必要はない。

まず、`cd E` と打つ。

```
% cd E
```

ここで、`[Tab]` を押すと、`E` で始まるファイルやディレクトリは他ないので、次のように名前が補完される。

```
% cd Enshu
```

また、上の例では、`1` で始まるファイルは`1998nen`、`1999nen` と 2 つある。ここで、`Mule` で `1999nen` を編集したければ、`mule 1999` まで打って、`[Tab]` を押してあげると、`mule 1999nen` と補完される。

11.5 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 `renshu` というファイルを Mule で作る。

```
% mule renshu &
%
```

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

```
renshu
ls -F
cal
```

この `renshu` は普通のファイルで、`ls -l` を実行してみるとパーミッションは `rw-r--r--` であることがわかる。

```
% ls -l
[中略]
-rw-r--r-- 1 ei99000 ei99      7 Dec  8 16:30 renshu
[中略]
%
```

このパーミッションを、誰でも読めて実行できるが、書けるのは自分だけに変更してみよう。つまり、パーミッションは「`rwxr-xr-x`」なので、モードは「755」だ。

```
% chmod 755 renshu
%
```

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

```
% ./renshu
1998nen      2000nen      hello*
1999nen      Enshu/       hello.c
                   12月 1999
                   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
%
```

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。

11.6 ワイルドカード

シェルではいくつ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を使う。

まず、I、 my、 me、 mine、 th、 the、 this、 these、 who、 which、 when というファイルがあったとする。

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。

* 何個かの何かの文字を表す

例えば、 th* と書くと、 th や、 the や、 this や these のような、 th で始まる、ファイルに対応する。

それから、 *e と書くと、 e で終わる me、 mine、 the、 these に対応する。

また、同様に t*e と書くと、 the、 these に対応する。

また、単に * とすると、すべてのファイルに対応する。

? 何かの 1 文字を表す。

例えば m? なら、最初が m で、次に何か 1 文字がある、 me だけに対応する。

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* とか打てば、 cat th the this these と打ったのと一緒にになる。

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、 th、 the、 this、 these を一気に消せる。

特に、 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。

11.7 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` だ。この中をいじると、初期設定を変えることができる。

では、以前と同様に、`cp .cshrc .cshrc-orig` などと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-euc  
  
# japanese input method  
setenv CANNHOST 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
```

```
# aliases
#alias ls 'ls -CF'
#alias cp 'cp -i'
#alias rm 'rm -i'
#alias mv 'mv -i'
#alias cd 'cd \!*; echo $cwd'

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

# 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。

| | |
|--------------------------------|-----------------------|
| # aliases | |
| 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くか確認する。 |
| 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。 |

これはあくまで例なので、気に入らなければ変える必要は全くない。

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。

11.8 シェルの変更

シェルが何種類があることは説明したが、自分の基本にするシェルは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「change shell – 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 ei99000.  
Shell: /usr/local/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 7 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⁶。

```
/bin/sh  
/bin/csh  
/usr/local/bin/tcsh  
/usr/local/bin/bash  
/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 uxsrv1.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 uxsrv1.edu3.aomori-u.ac.jp
```

なお、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。`csh` と `tcsh` の初期設定ファイルは、`.cshrc`。`sh` と `ksh` の初期設定ファイルは、`.profile`。`bash` の初期設定ファイルは、`.bashrc`。`zsh` の初期設定ファイルは、`.zshrc` である。

⁶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12章 WWW ヘアクセスしてみよう

もともと、WWW (World Wide Web) は UNIX の世界を中心に広まったものだ。UNIX での、WWW を楽しんでみよう。この章では、Chimera の使い方や、Netscape Communicator の設定の方法を紹介する。また、lynx というテキスト・ブラウザや、HTML の文法チェックについても紹介する。

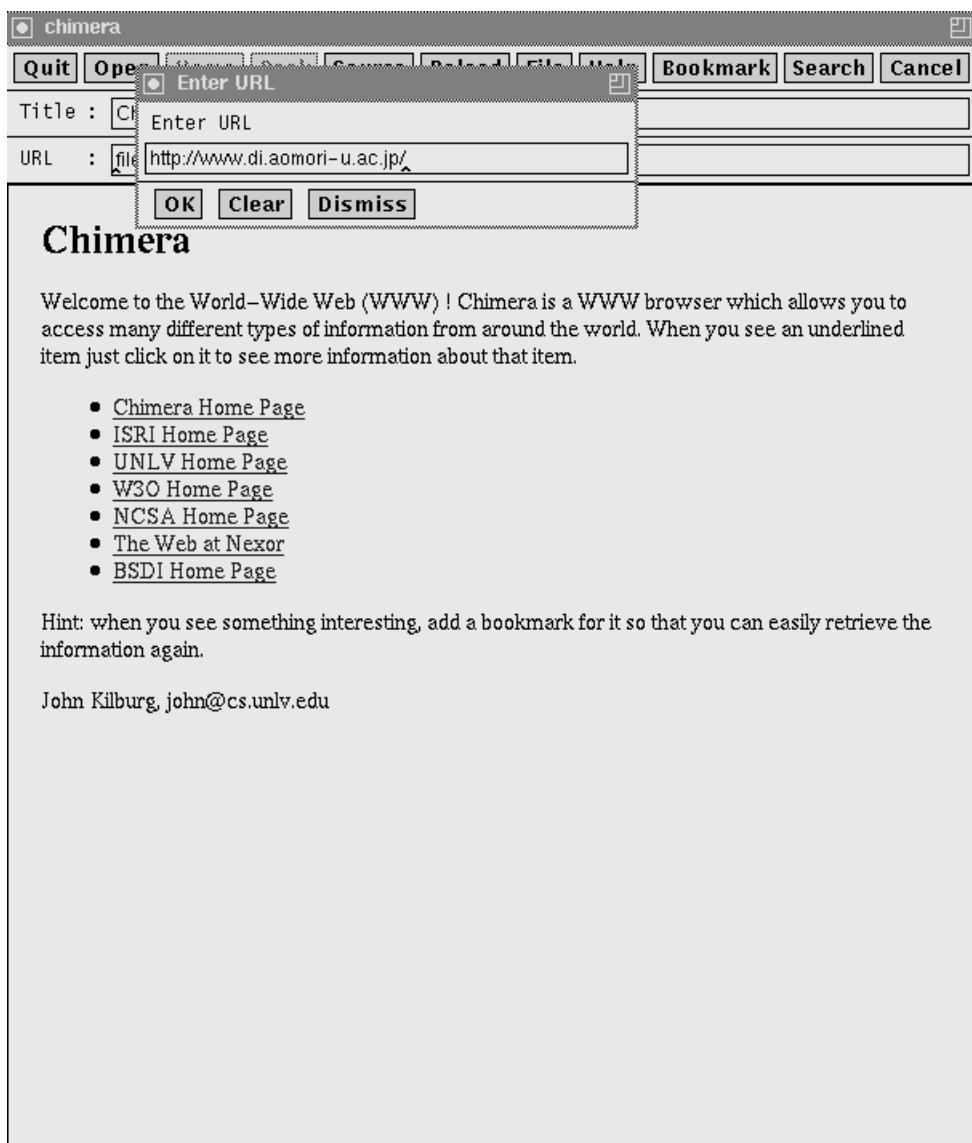
12.1 Chimera

日本語が表示できるブラウザのひとつに Chimera – キメラがある。これは、オールド・スタイルなブラウザで、画像の表示機能がいまいちである。起動するには、`chimera &` である。

% `chimera &`



URL を指定して、ページを見るには、**OPEN** をクリックする。するとダイアログが現れるので、ここに URL を入力し、**OK** をクリックする。



12.2 Netscape Communicator の設定の仕方

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。基本的に使い方は Windows NT のときと同じである。

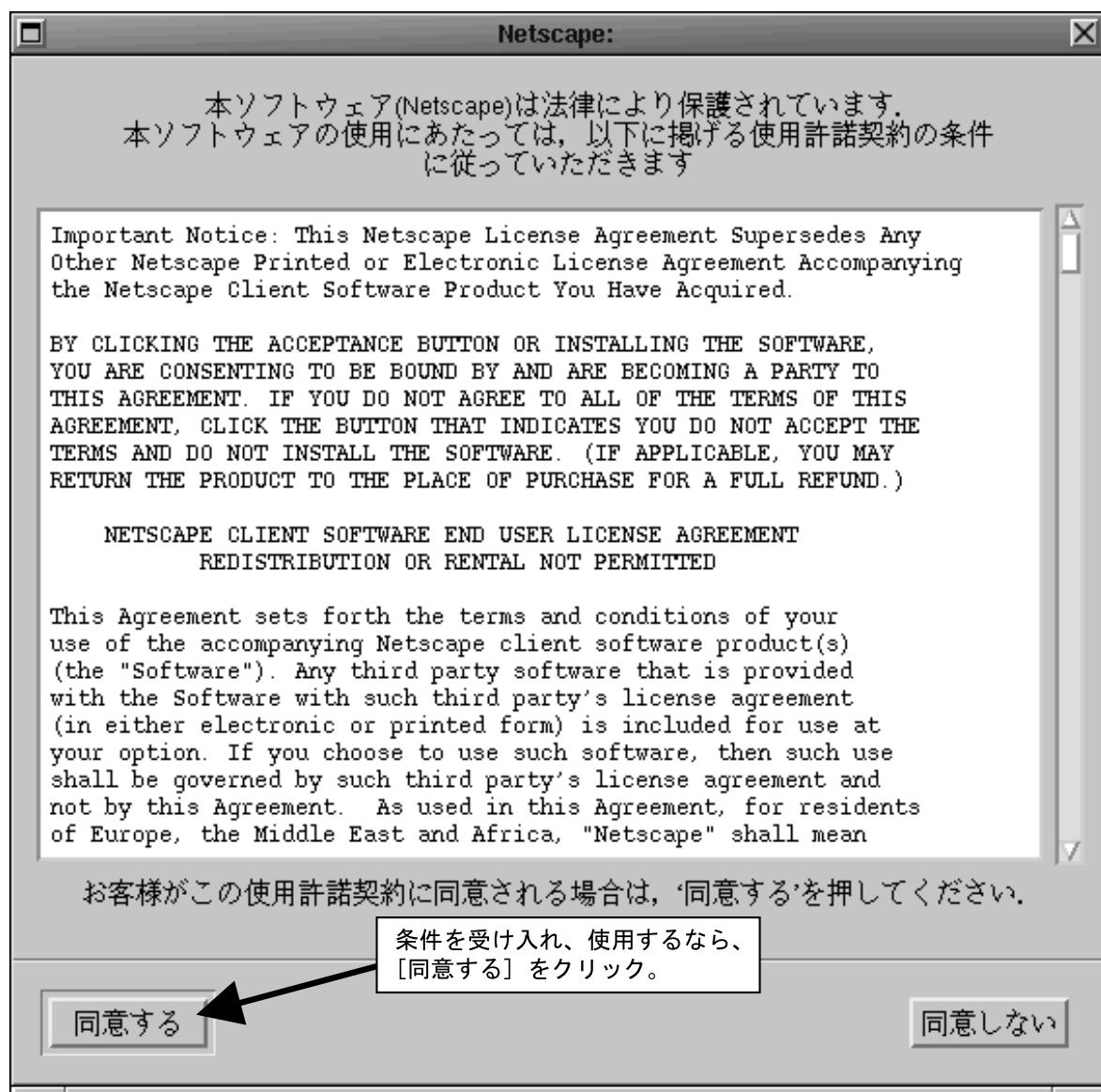
12.2.1 起動

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 &**

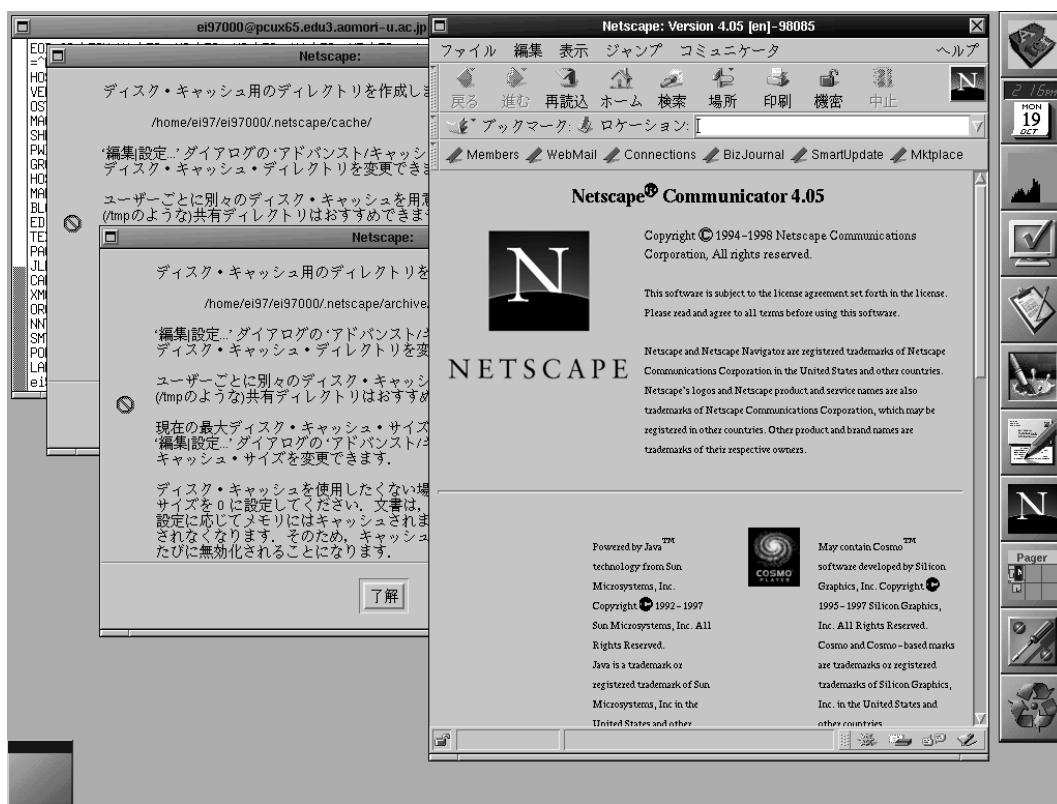
12.2.2 使用許諾条件

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。

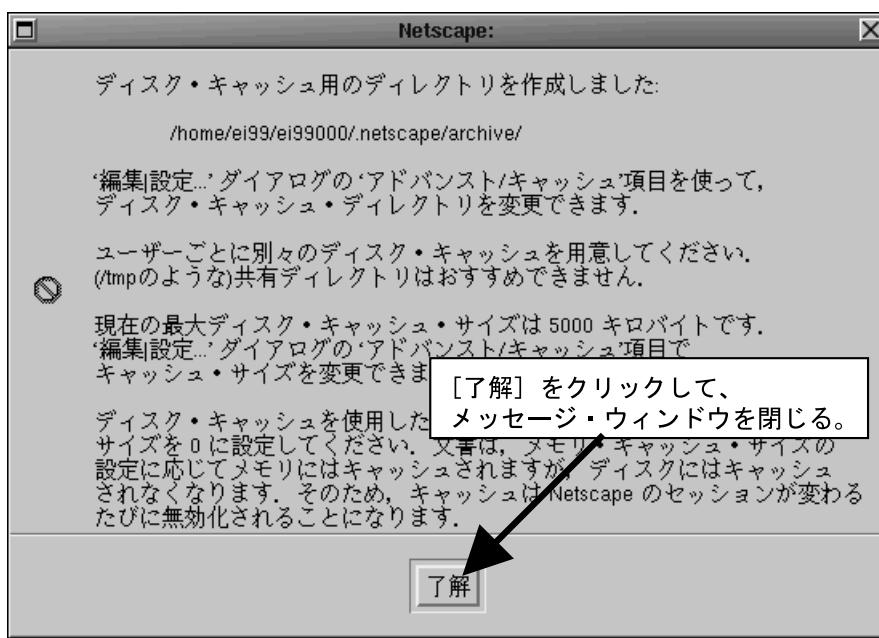


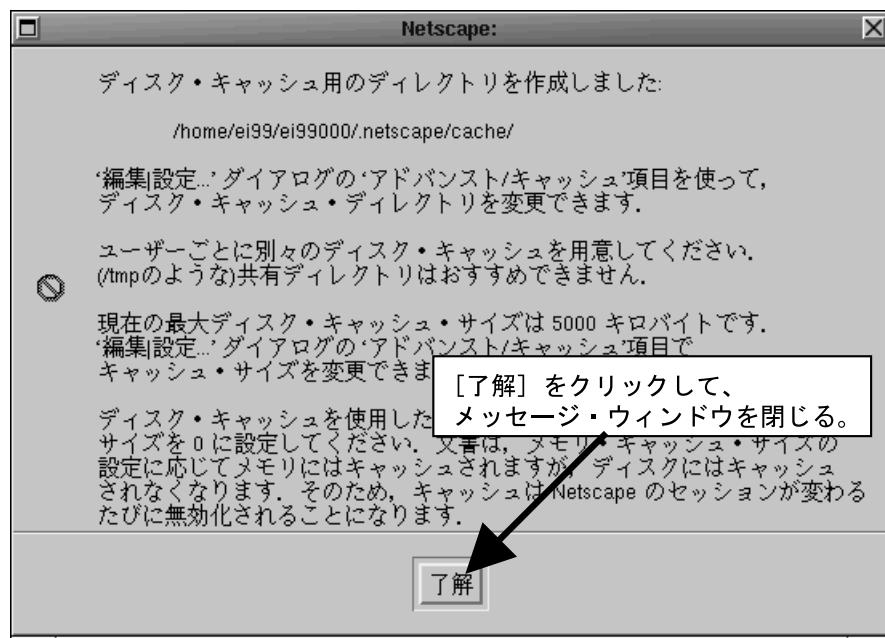
12.2.3 初回起動時メッセージ

メッセージがいくつかでて、Netscape が起動する。



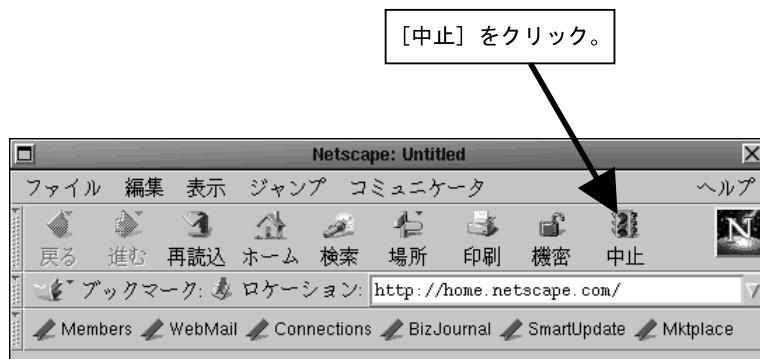
キャッシュを作成した等のメッセージは [了解] をクリックして閉じる。





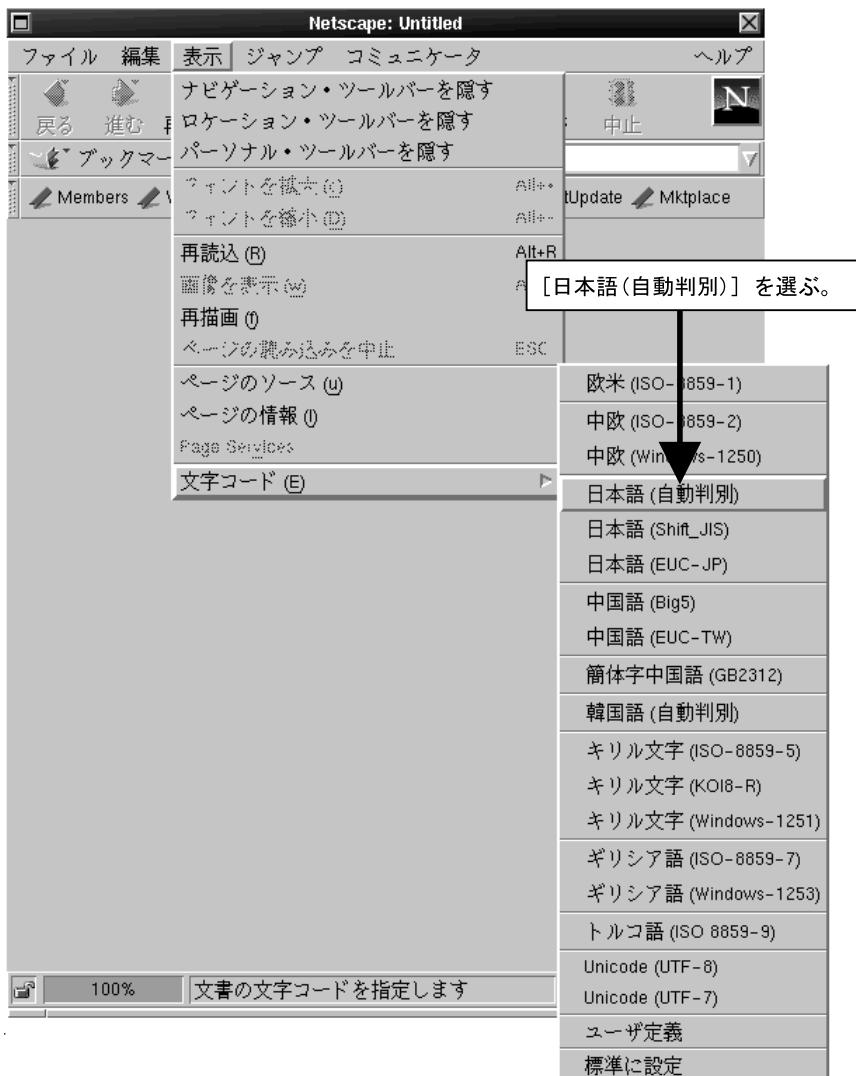
12.2.4 設定

Netscape 社のページにアクセスしようとしているので、[中止] をクリックして止める。

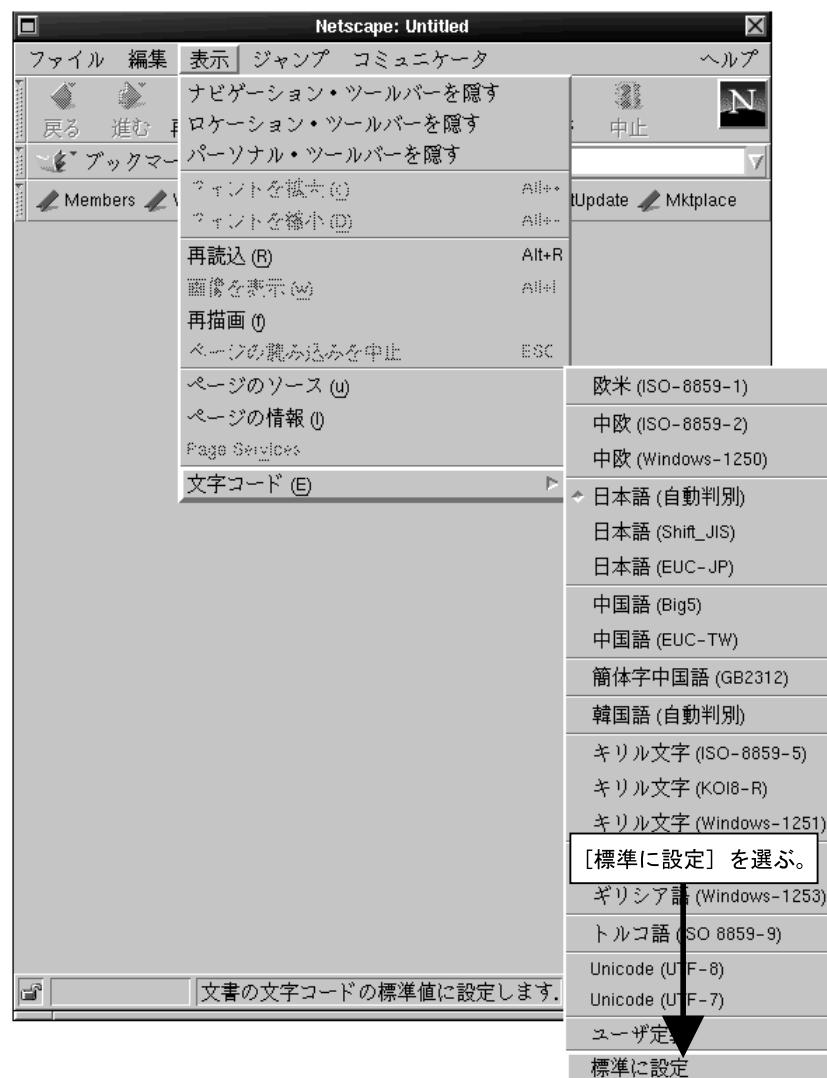


文字コードの設定

[表示] メニューから、[文字コード] を選び、その中の [日本語 (自動判別)] を選ぶ。

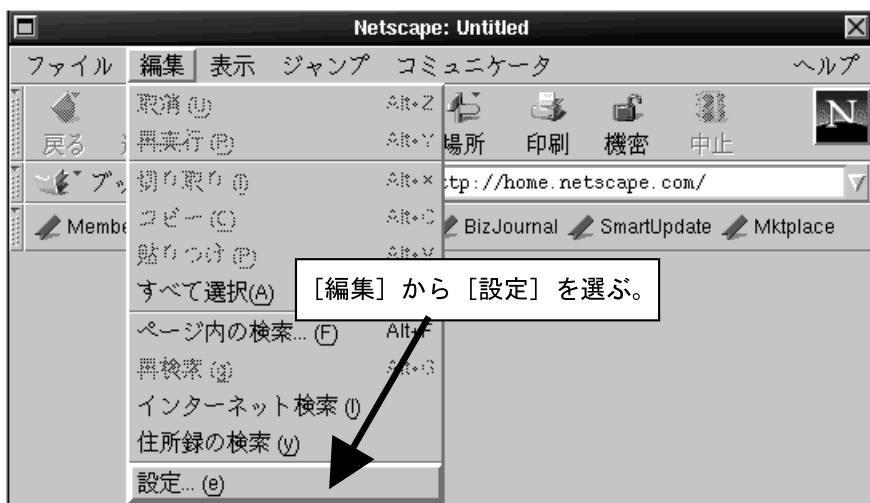


再び [表示] メニューから、[文字コード] を選び、その中の [標準に設定] を選ぶ。

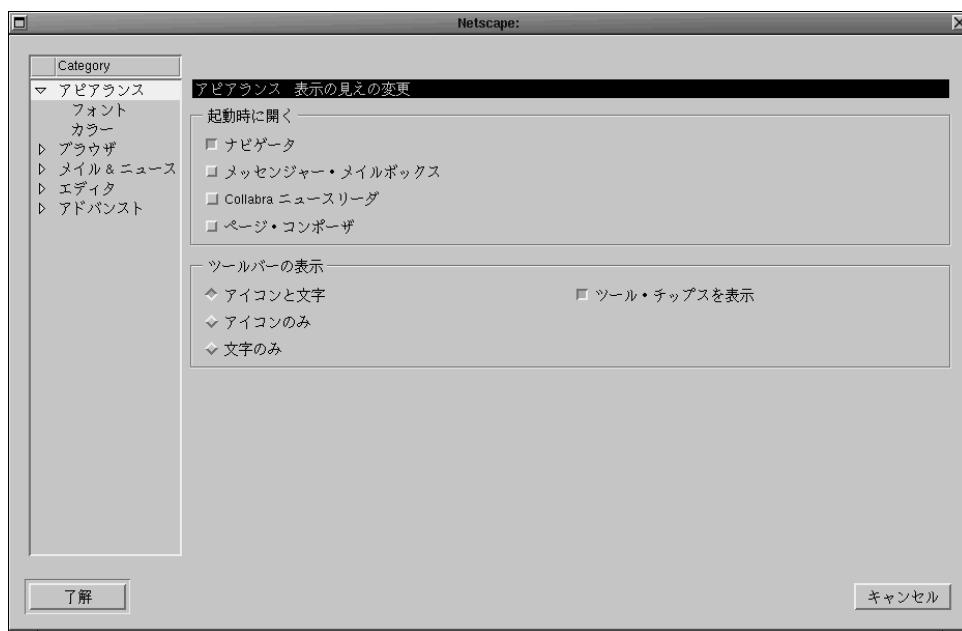


設定用ウィンドウの表示

[編集] メニューから、[設定] を選ぶ。

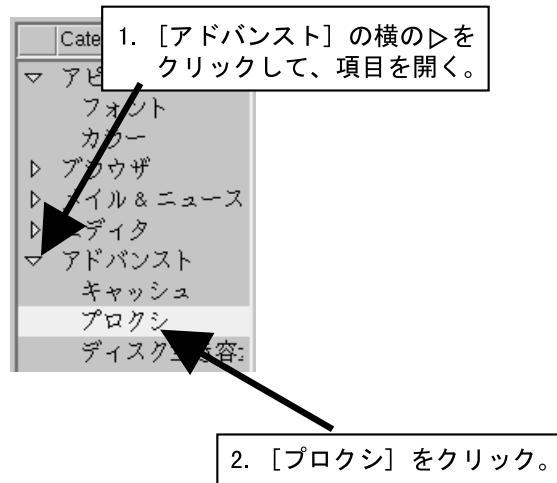


設定用のウィンドウが開く。

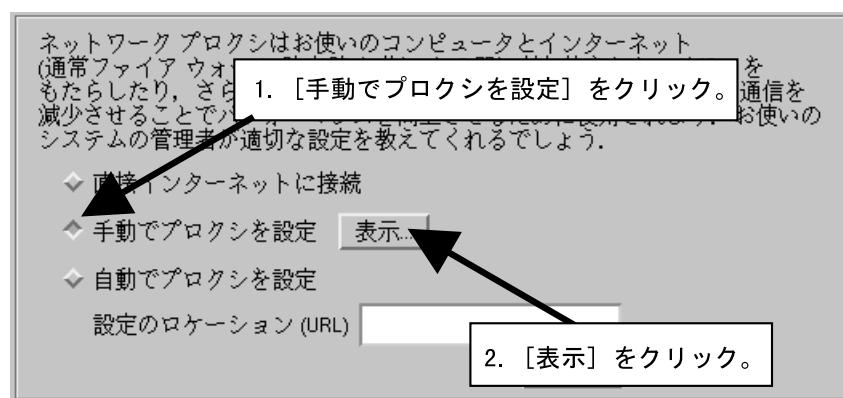


プロクシの設定

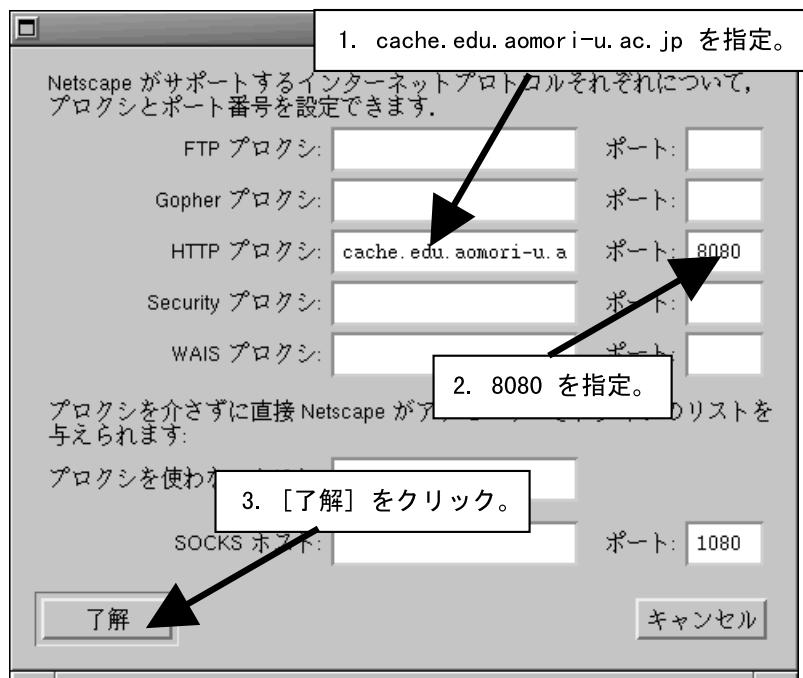
[アドバンスト] の項目の左側の ▶ をクリックして開き、その中の [プロクシ] をクリックする。



現れたウィンドウで [手動でプロクシを設定する] をクリックし、その横の [表示] をクリックする。



現れたウィンドウで [HTTP プロクシ] の項目を設定し、最後に [了解] をクリックする。

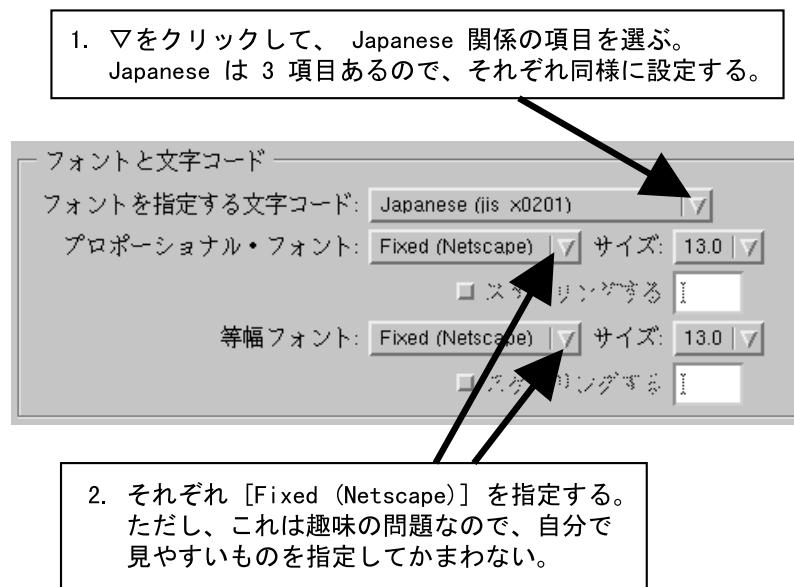


フォントの設定

[アピアランス] の項目から、[フォント] を選ぶ。



現れたウィンドウで、[フォントを指定する文字コード] にJapanese 関係の項目が、全部で 3 つあるので、それぞれに [Fixed (Netscape)] を設定する。



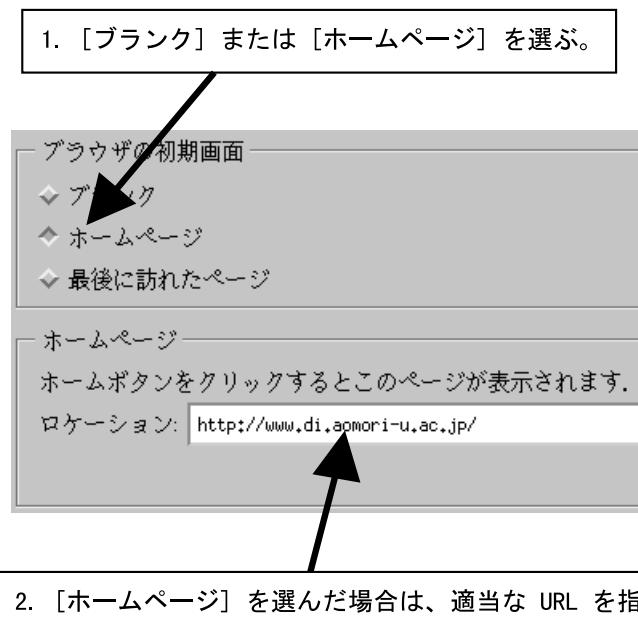
このように設定すると、次回起動時から、小さい文字が美しく表示可能になる。

ホームページの設定

[ブラウザ] の項目を選ぶ。



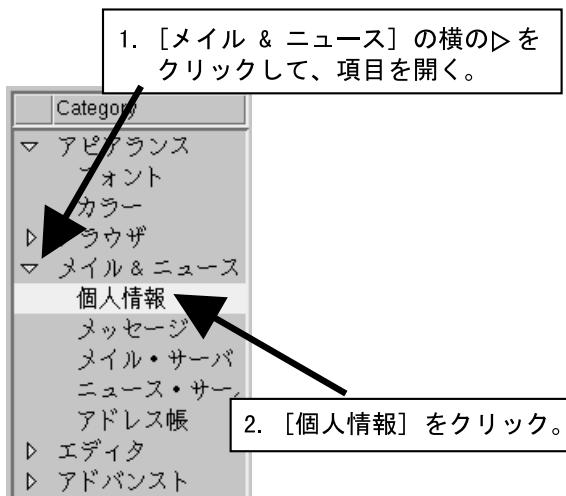
現れたウィンドウで、[ブラウザの初期画面] を [ブランク] または、[ホームページ] にする。
[ホームページ] にした場合は、[ホームページ] の [ロケーション] に、起動時にアクセスしたい URL を入力しておく。



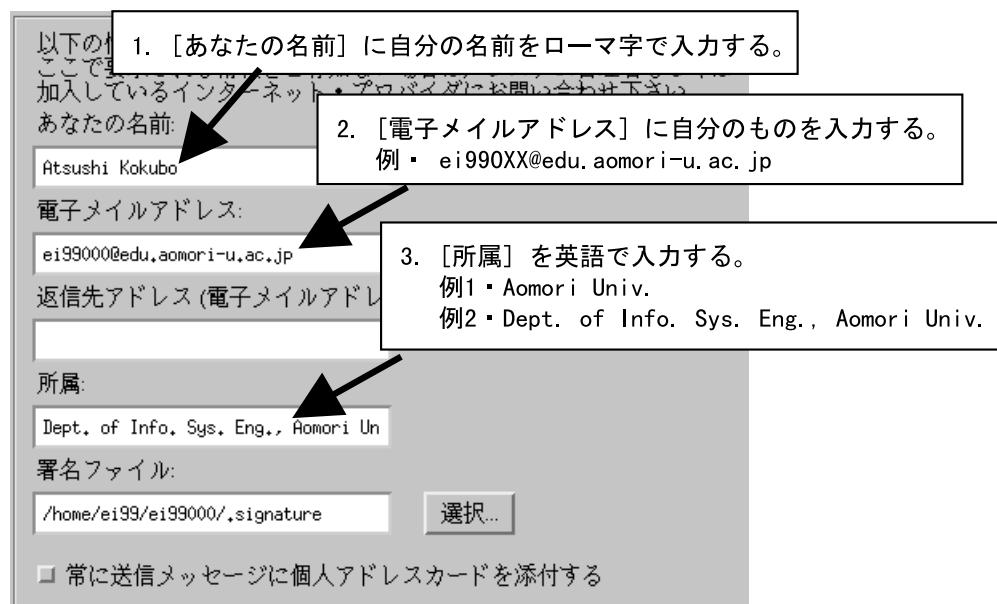
2. [ホームページ] を選んだ場合は、適当な URL を指定する。

メールの設定

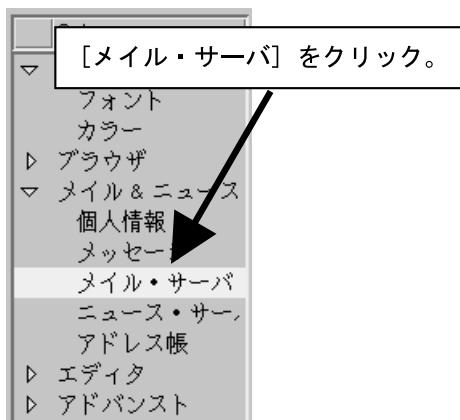
[メール & ニュース] の項目の左側の ▶ をクリックして開き、その中の [個人情報] をクリックする。



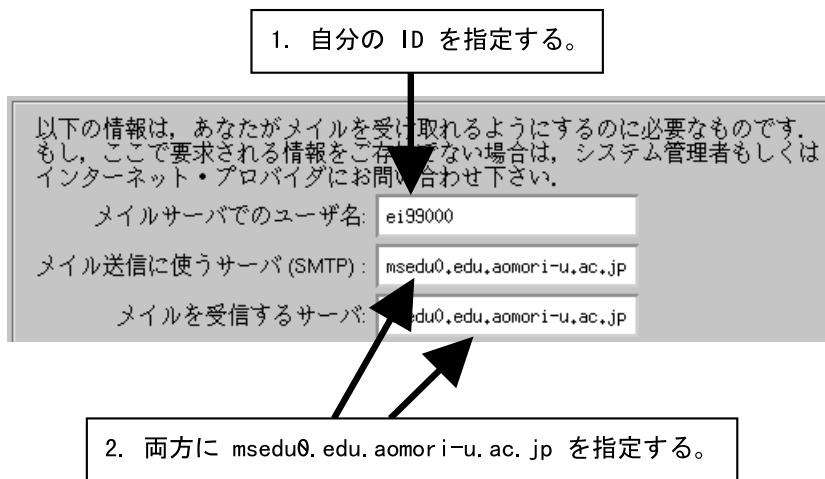
現れたウィンドウで [あなたの名前]、[電子メールアドレス]、[所属] などを設定をする。



[メイル & ニュース] の項目の [メイル・サーバ] を選ぶ。

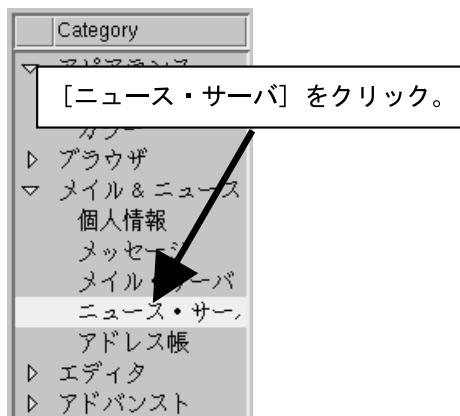


現れたウィンドウで [メイル・サーバでのユーザ名]、[メイル送信に使うサーバ (SMTP)]、[メイルを受信するサーバ] などを設定をする。

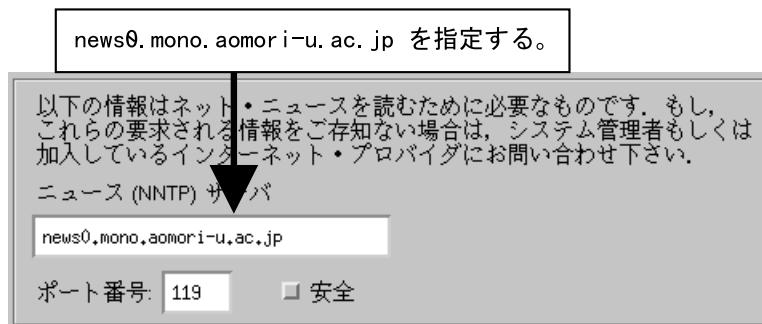


ニュースの設定

[メイル & ニュース] の項目の [ニュース・サーバ] を選ぶ。

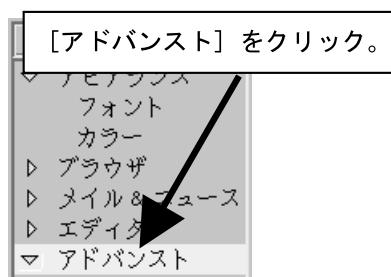


現れたウィンドウで [ニュース (NNTP) サーバ] を設定をする。

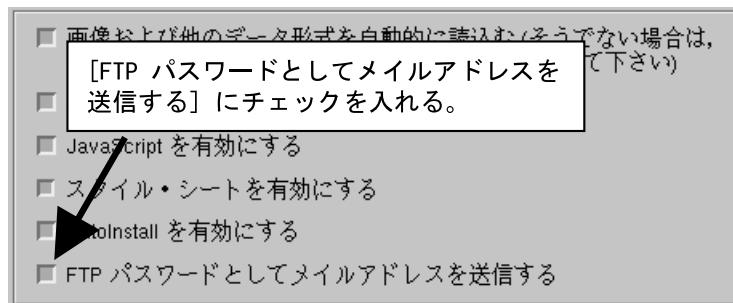


anonymous FTP の設定

[アドバンスト] の項目を選ぶ。



現れたウィンドウで [FTP パスワードとしてメールアドレスを送信する] をクリックする。

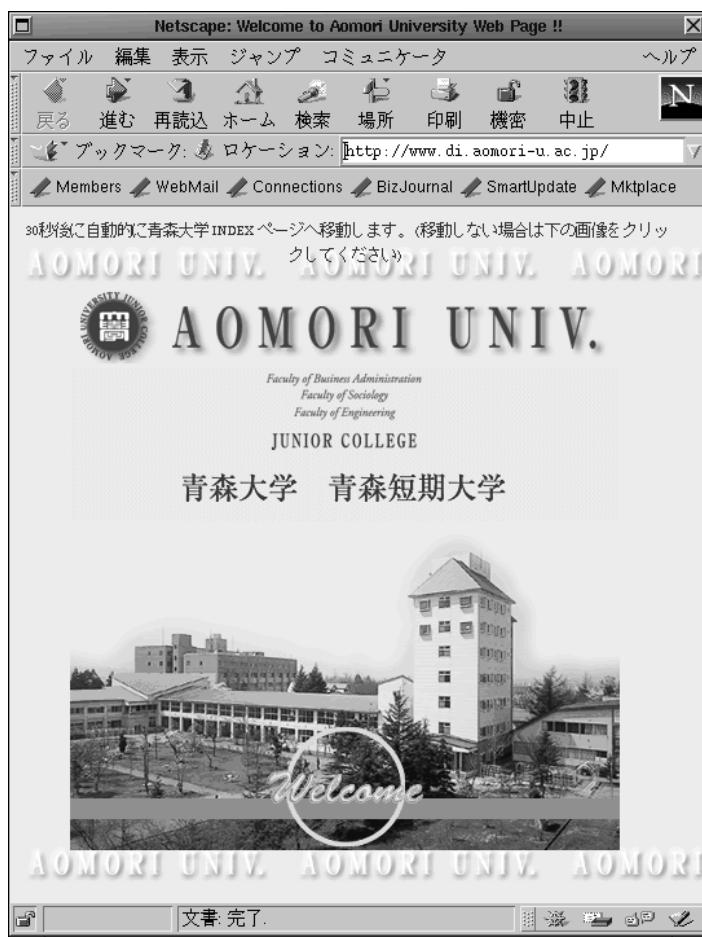


再起動

[了解] をクリックしていき、設定用のウィンドウをすべて閉じる。

[ファイル] メニューから、[終了] を選んで、一度終了する。

再び、`netscape &` と入力して、Netscape を再起動させる。



12.2.5 Netscape で日本語入力

Netscapeなどのアプリケーションに日本語入力したい場合には、次のようにして `kinput2` を あらかじめ 起動しておく。

```
% kinput2 &
```

このときの日本語入力は Canna を使って行なわれる。基本的には Mule で使用する場合と一緒だが、日本語入力の On/Off は `C-o` ではなく、`Shift+スペース` である。

12.3 テキスト・ブラウザ lynx

UNIX は元もと文字ベースの端末で動くように設計されていた。そこで、 GUI でなくても使用可能なコマンドが用意されていることが多い。

そのようなコマンドの一つ、 lynx は基本的に文字データだけを表示するブラウザである。起動するには、 lynx 「見たいページの URL」 である。

```
% lynx http://www.aomori-u.ac.jp/  
%
```

すると、次のような画面になる。



図 12.1: lynx の画面

ちなみに が一つ前のページに戻る。 が一つ先のページに進む。 が画面の上スクロール、 が画面の下スクロール。リンクをたどるには、リンクの上で である。

世の中には、このようなブラウザを使っている人もいるので、Web ページを作るときには、一応画像が全く表示できなくても、そこそこ意味が通じるようにしておいた方がよい。

なお、 lynx -image_links と起動すれば、画像の上で すると、一応表示させることもできる。

12.4 jweblint

HTMLを書いたときに、文法をチェックしてくれるコマンドもある。それがjweblintである。

例えば、Muleで次のようなtest.htmlというファイルを作ったとしよう。

```
<HTML>
<HEAD>
<TITLE>test</TITLE>
</HEAD>

<BODY>
<H1>これはテスト・ページだよ</H2>
</BODY>
</HTML>
```

jweblint test.htmlと打つと、次のように文法のミスを教えてくれる。

```
% jweblint test.html
test.html(1): 最初のエレメントが DOCTYPE の記述ではありませんでした。
test.html(6): おかしなヘディングです - 開始タグは <H1> ですが、終了タグは </H2>
です。
%
```

この章で紹介したコマンド

WWW関連コマンド他

- chimera : Chimeraの起動
- netscape : Netscape Communicatorの起動
- kinput2 : アプリケーションへの日本語入力
- lynx : テキスト・ブラウザlynxの起動
 - 使い方: lynx URL
- jweblint : HTML文法チェック器
 - 使い方: jweblint ファイル名

第13章 電子メールとニュース

mnews でメールとニュースを使ってみよう

`mnews` はミニ・ニュースリーダーの略である。`mnews` を使うと、電子メールやニュースの読み書きができる。

この章では、その方法を紹介しよう。なお、電子メールやニュースを使うには、マナーを忘れないように。

13.1 mnews の起動と終了

`mnews` を起動するには、「`mnews`」と打つ。

```
% mnews
```

すると、下のような画面になって、`mnews` が起動する。



なお、最初に終了する方法も紹介しておこう。`mnews` を終了するには、「`Q`」と打つ。

13.2 mnews でニュースを読む

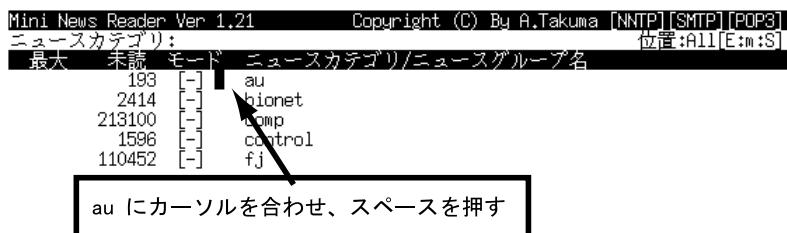
13.2.1 ニュースを読む

階層を降りるにはスペース

mnews を起動すると項目が並んでいて、それらはニュース・グループや電子メールを表している。

ニュース・グループの名前は、「fj.rec.games.video.playstaion」や「au.announce」などのようになっていることは、既によく知っていることと思う。mnews を起動すると、このような名前の、一番先頭の「fj」や「au」の部分が見えている。

たとえば、「au.announce」を読むには、矢印キーを使って、まず「au」にカーソルを合わせて、**[スペース]** を押す。すると、「au」の中に入って、「au」で始まるニュース・グループの一覧が見える。

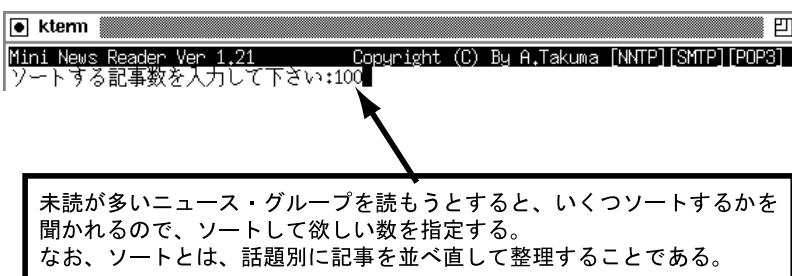


次に「au.announce」にカーソルを合わせて、**[スペース]** を押す。すると、「au.announce」の記事の一覧が現れる。

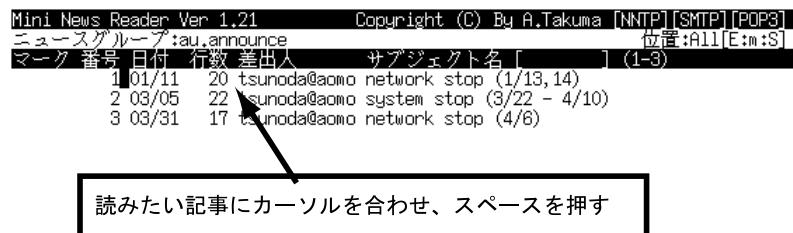


ただし、このとき、読んでいない記事が多いと、「ソートする記事数を入力して下さい」と言われる。ソートというのは、同じような話題ごとに記事を並べ替えることだ。

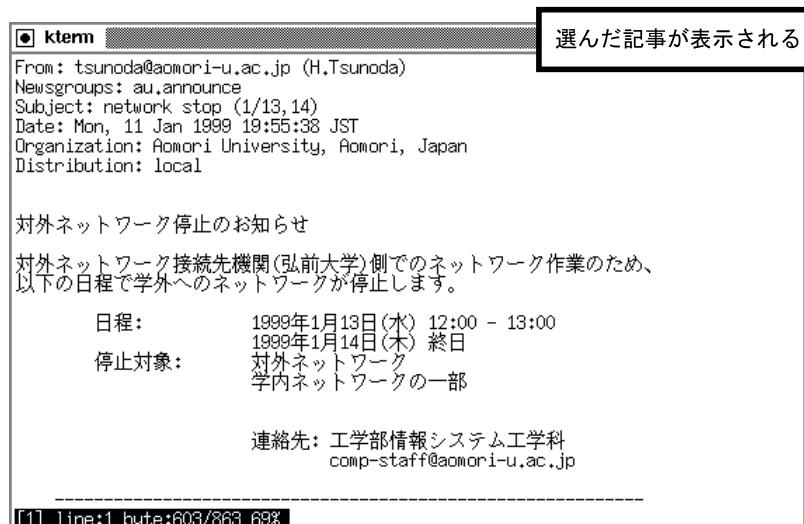
ここでは、好きな個数を指定すればいい。大きい数を指定すると、それだけ時間がかかる。ちなみに、ソートしなくてよければ、「0」と答えればよい。



次に、読みたい記事にカーソルを合わせて [スペース] を押す。



すると、記事が表示される。続けて [スペース] を押して行けば、次のページが表示される。
更に [スペース] を押して、記事の最後まで進むと、次の記事が表示される。

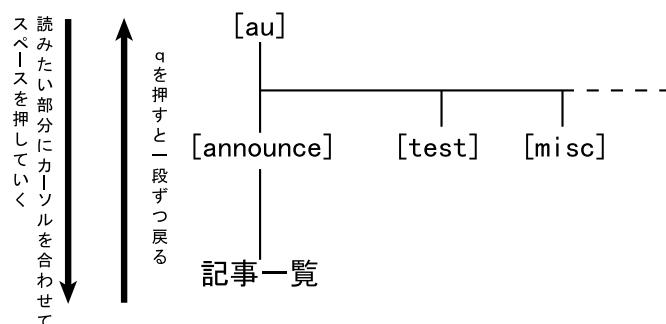


階層をのぼるには「q」

逆に、今、[au.announce] の記事を読んでいて、記事の一覧に戻りたければ、「q」を押す。
そして、もう一段元に戻りたければ、更に「q」を押せばよい。

つまり、下の階層に降りるには [スペース] を押し、上の階層にもどるには「q」を押せばよい。

階層の移動



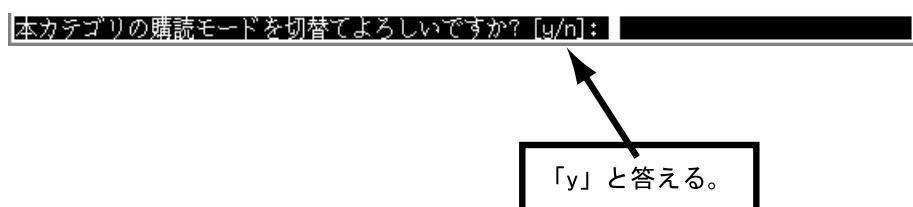
いらないニュース・グループ見えなくするには

趣味によっては、読まないニュース・グループはたくさんあるだろう。これらを表示しない方法を紹介しよう。

まず、読まないニュース・グループにカーソルを合わせて「u」と打つ。



「購読モードを切り替えるか」と聞かれるので、「y」と答える。

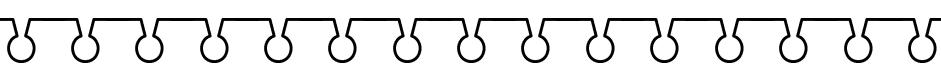


すると、ニュース・グループに「U」というマークが付く。このマークが付いたニュース・グループは、次回から画面に表示されなくなる。



ちなみに、間違って「U」を付けてしまった場合は、もう一度「u」と打つと元にもどる。

こうやって消してしまったニュース・グループはふだんは表示されないが、mnews の中で「L」と打つと、モードが切り替わって、一時的に見えるようになる。この段階で、「U」マークを「u」と打つと消すと、次回からは表示されるようになる。



練習

- 何か適当なニュース・グループを読んでみよう。

13.2.2 ニュースに記事を投稿するには

ニュースに投稿したものは、全世界に流れてしまうので、十分注意事項を守って使って欲しい。

なお、投稿のテストをしたい場合には、au.test で行う。ここに投稿したものは、青森大学の中から外へは出でていかない。

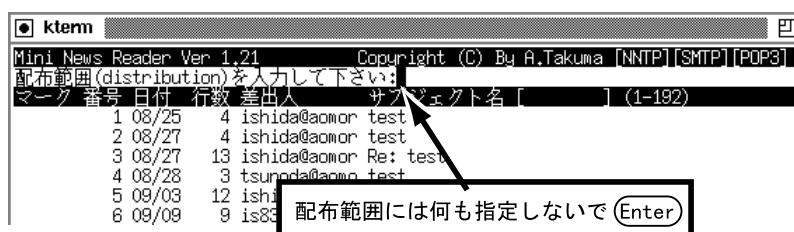
ニュースを投稿するには、まず、投稿したいニュース・グループに入って、他の記事が見えている状態にする。そして、「a」と打つと、投稿するかと聞かれるので、「y」と答える。



次にタイトルを聞かれるので、英語かローマ字で打つ。



次に配布範囲を聞かれるので、何も打たずに **[Enter]** する。

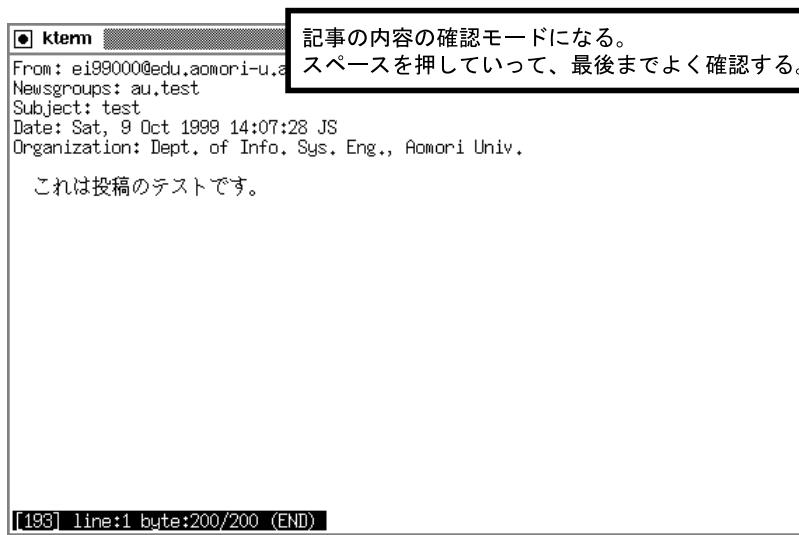


すると、Mule が起動する。ここで、「--text follows this line--」よりも下に、記事の本文を書く。

書き終わったら、「file」メニューから、「exit emacs」を選び、セーブして終了する。



すると、確認画面になるので、読んで確認する。`jless` とほぼ一緒で、[スペース] を押すと、先の画面が読める。

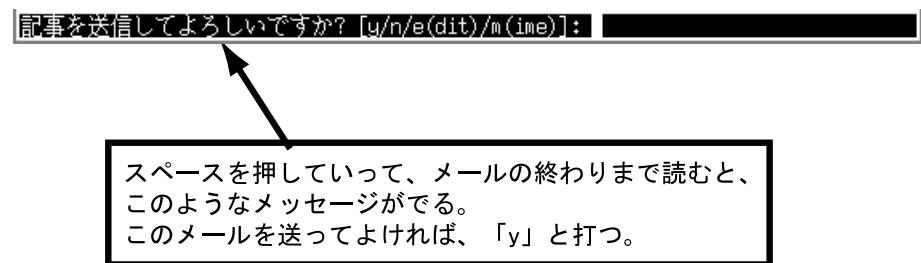


[スペース] を押して記事を読んでいき、ファイルの最後にたどりつくと、画面下にメッセージが出て、送つていいかどうか聞かれる。

ここで、「y」を押すと送信される。

「n」を押すと、また編集し直すか聞かれる。編集し直すを選ぶと Mule が立ち上がって書き

直せる。編集し直さないと選ぶと、記事の送信が中止される。

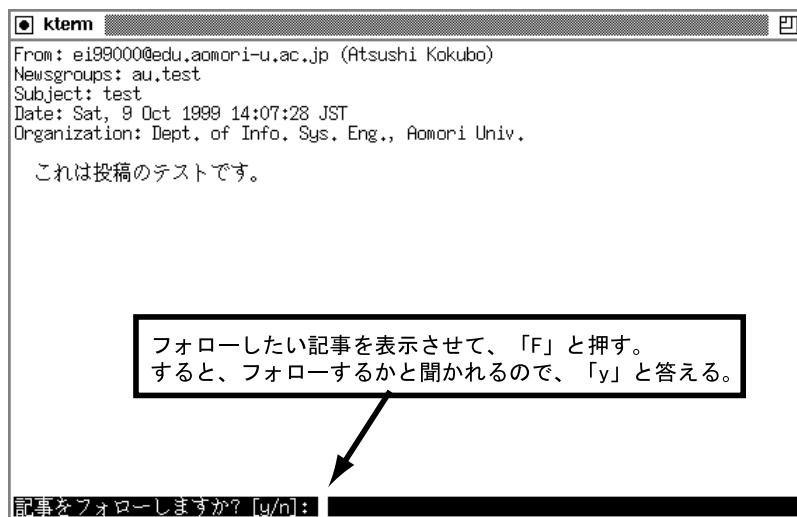


記事を送信すると元の画面に戻る。

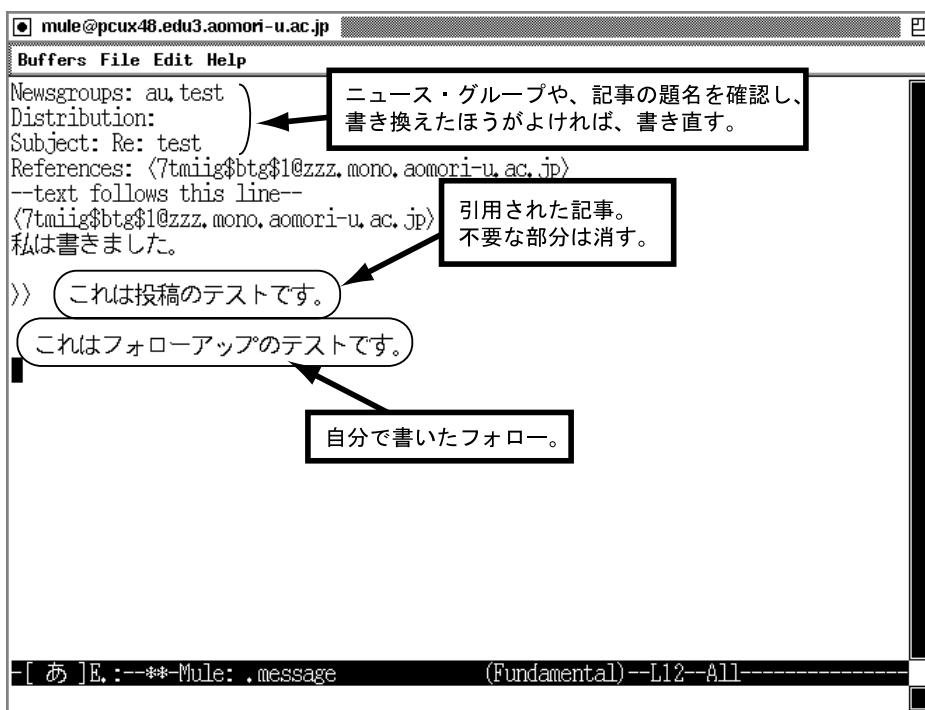
13.2.3 ニュースに記事にフォローするには

記事をフォローするのは、全世界向けに書く必要のある場合だけにする。個人的なことは、相手にメールで送ること。

フォローするには、ニュース・グループに入って、フォローしたい記事を表示し、「F」と打つ。すると、フォローするか聞かれるので「y」と打つ。



すると、Mule が起動して、相手の記事が変換されて読み込まれる。



ここで、投稿先のニュース・グループや、記事の題名などを確認し、必要ならば修正する。それから、引用された記事で、フォローするのに不要な部分はすべて消し、フォローを書き加える。

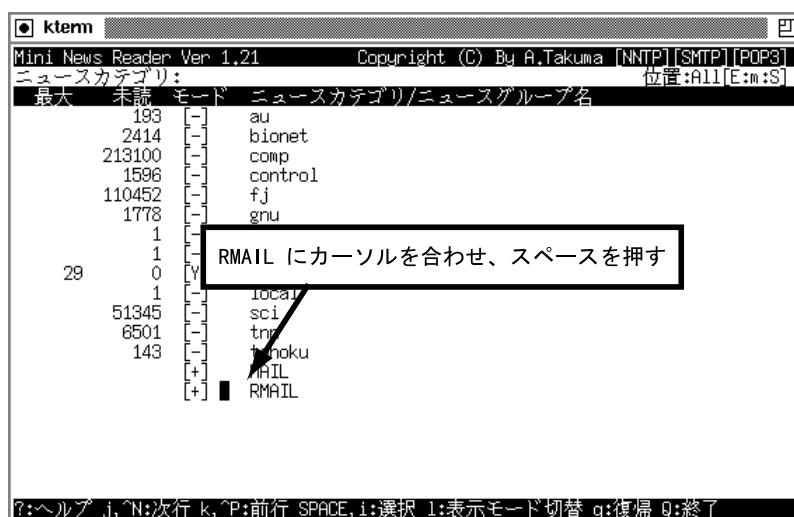
なお、意味もなく長い引用を全世界に流すことは無駄なので、引用は必要最小限にとどめる。書き終わったら、Mule を終了する。後は、普通に投稿するときと全く同様だ。

13.3 mnews で電子メールを使う

mnews でメールを使ってみよう。

まず最初に、矢印キーを使って、画面で下の方にある「MAIL」か「RMAIL」にカーソルを移動し、**【スペース】**を押す。

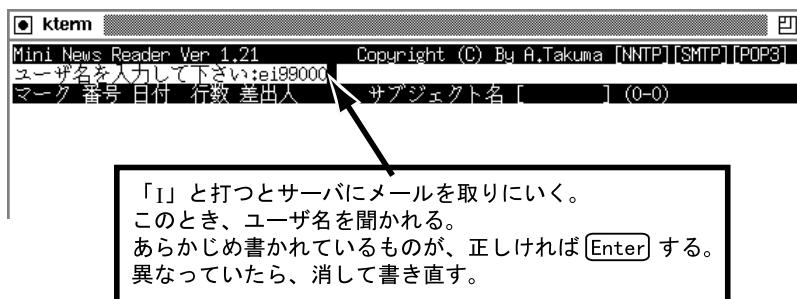
なお、ここでは「RMAIL」の方を使うことにする。「RMAIL」にカーソルを移動して、**【スペース】**を押してみよう。



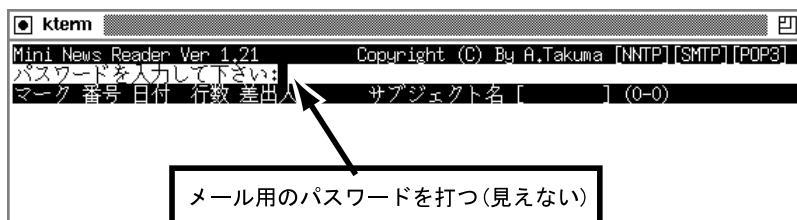
すると、次のような画面になる。もし、このとき、過去に読んだメールがあれば、一覧が表示される。



新しく届いたメールが届いているかどうかをチェックするには、「I」と打つ。すると、「ユーザ名」を聞かれる。そこに書かれているものが正しければ、[Enter] しよう。間違っている場合は、消して入力しなおす。

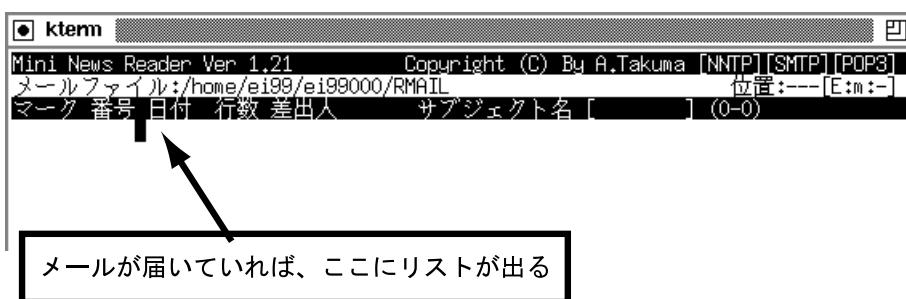


すると次は、「パスワード」を聞かれるので、「電子メール用のパスワード」を入力して、[Enter] する。



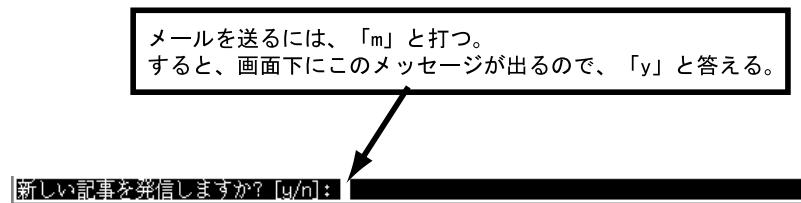
もし新着のメールがあれば、新たに表示される。来ているメールは、カーソルを合わせて、[スペース] を押すと読める。

なおこの例では、メールが来ていなかった。



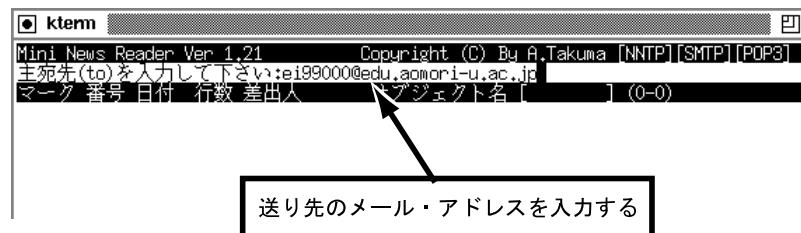
13.3.1 メールを送る

メールを送るには、メールのリストが表示されている状態にして、「m」(mail の意味) と打つ。すると、送るかどうか確認があるので、「y」と打つ。

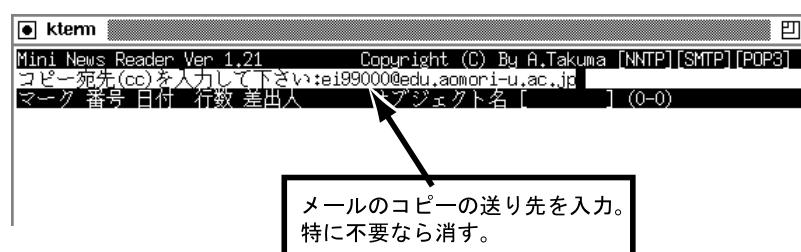


すると、送り先を聞かれる。送り先の電子メールのアドレスを、ここに打ち込んで [Enter] する。

今回は、練習のために自分自身に送ってみよう。なお、自分自身の電子メールアドレスは、「[自分の ID]@edu.aomori-u.ac.jp」だ。



すると、Cc: (カーボン・コピー) 先を聞かれるので、必要なら電子メールのアドレスを打ち込んで [Enter] する。不要なら、消してしまっていい。

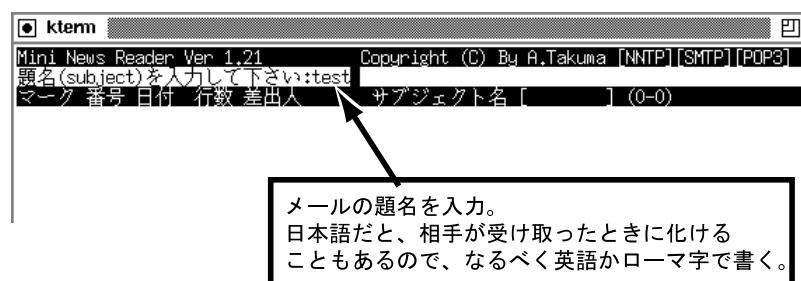


Cc: とは?

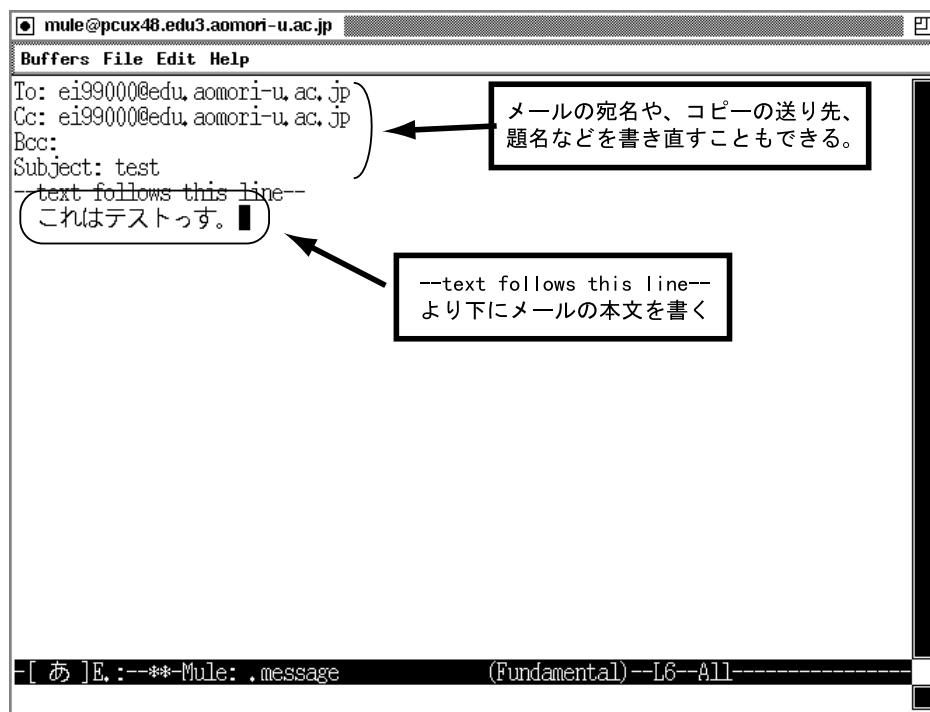
Cc: は、同じ内容のメールを 2 人以上に出したいときに使う。1 人目のアドレスを To: に書き、2 人目からは Cc: に書く。

ちなみに、自分のアドレスを書くと、送ったメールを取っておくのに使える (mnews のデフォルトはそうなっている)。

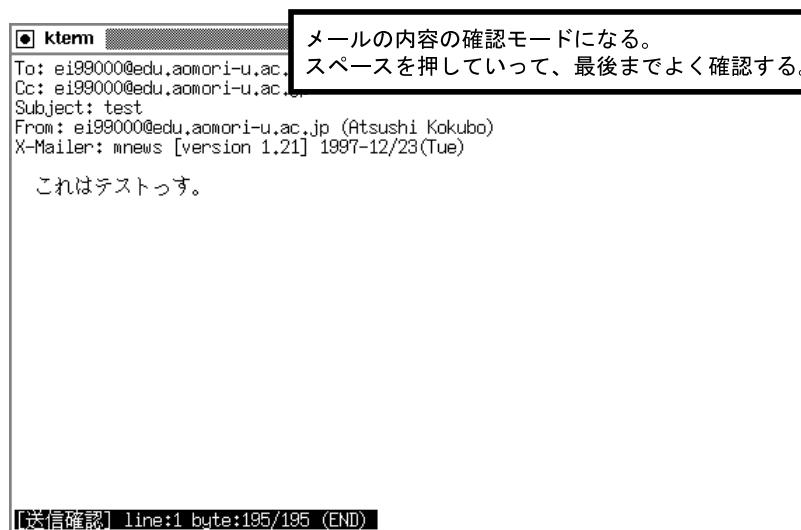
次に、メールの題名を聞かれるので、なるべく英語またはローマ字で書く。



すると、Mule が起動する。「--text follows this line--」の下に、メールの本文を書く。書き終わったら、「file」メニューから、「exit emacs」を選び、セーブして終了する。



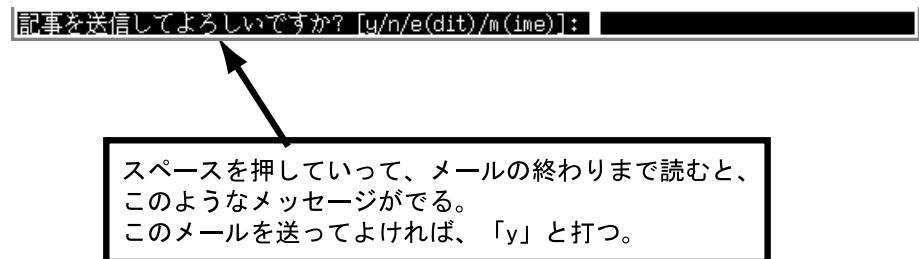
すると、確認画面になるので、読んで確認する。jless とほぼ一緒で、[スペース] を押すと、先の画面が読める。



[スペース] を押してメールを読んでいく、ファイルの最後にたどりつくと、画面下にメッセージが出て、送つていいかどうか聞かれる。

ここで、「y」を押すと発送される。

「n」を押すと、また編集し直すか聞かれる。編集し直すを選ぶと Mule が立ち上がって書き直せる。編集し直さないを選ぶと、メールの発送が中止される。

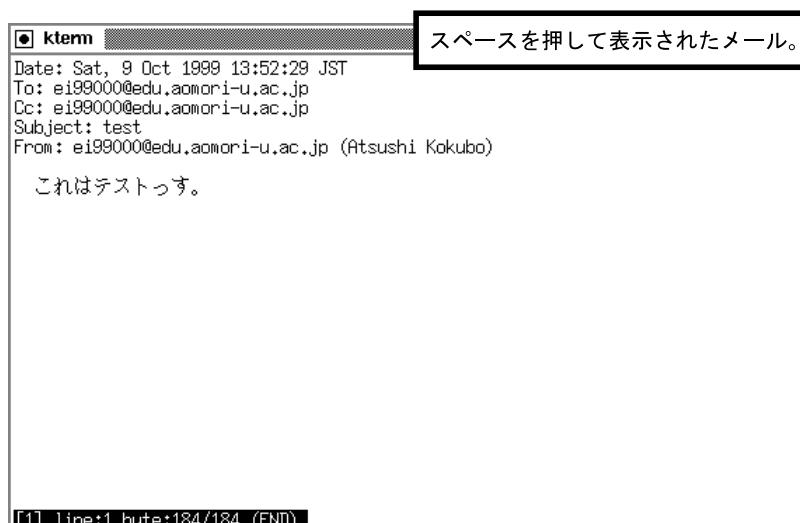
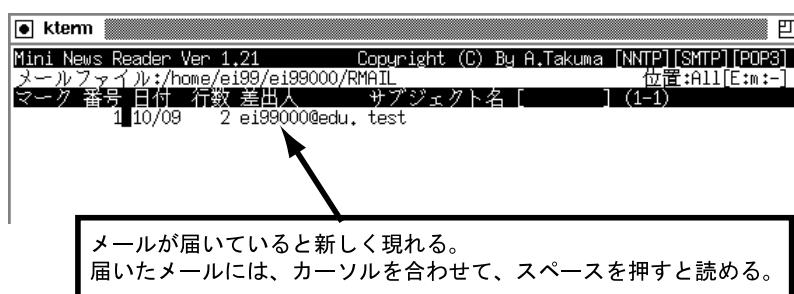


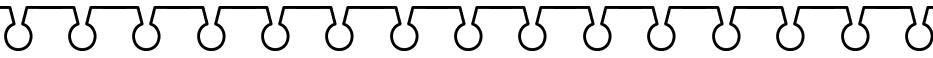
メールを発送すると元の画面に戻る。

今送ったメールが届いているはずなので、確認してみよう。新しく届いたメールを読むには「I」を押す。



自分で送ったメールがうまく届いていれば、表示されるはずだ。自分で送ったメールにカーソルを合わせて、**[スペース]**で読んでみよう。



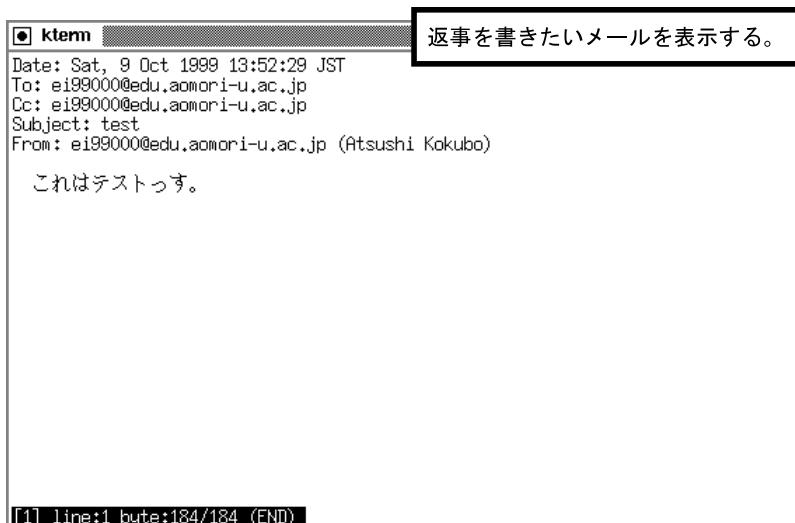


練習

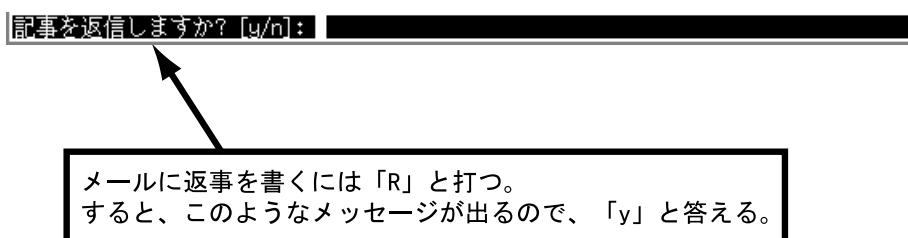
1. 自分にテストのメールを出して読んでみよう。

13.3.2 メールに返事を書く

メールに返事を書くには、まず返事を書きたいメールを読んで表示させる。

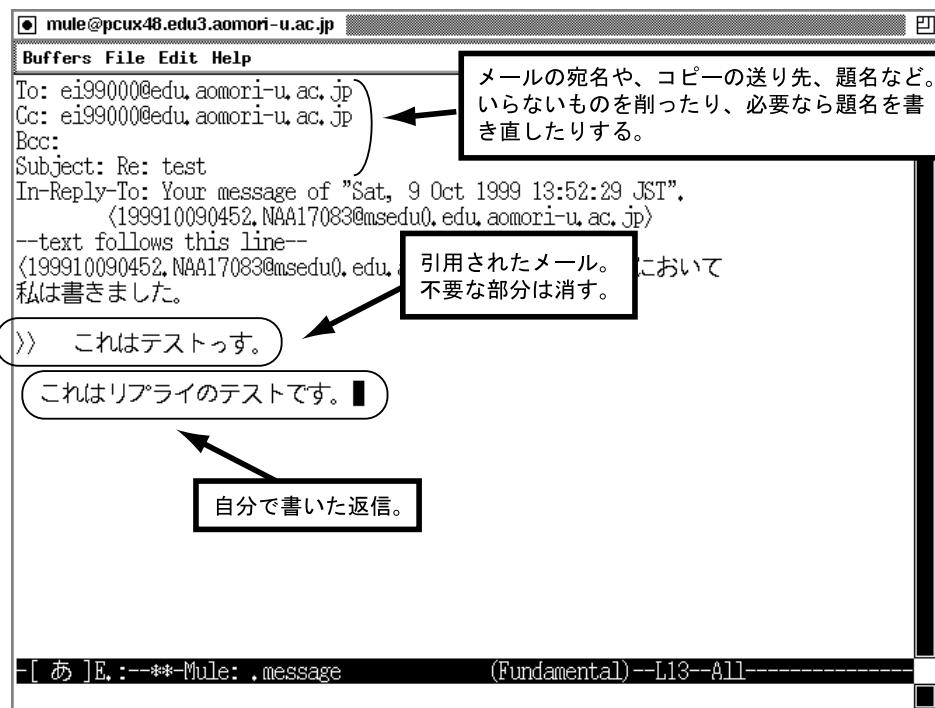


そして、メールを表示した状態で、「R」(リプライの意味)と打つ。すると、返事を書くかどうか聞かれるので「y」と答える。

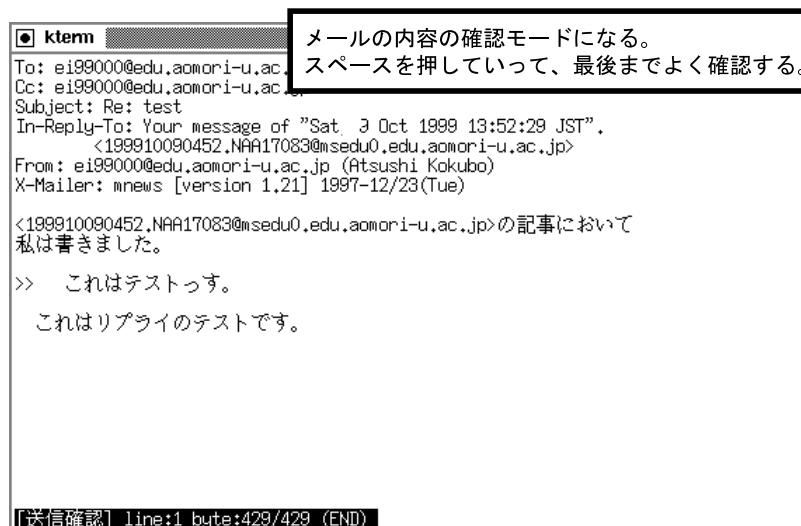


すると、Mule が立ち上がり、相手のメールが引用されている。

まず、To:(送り先) や、Cc:(コピーの送り先)などを確認して、問題があつたら、書き直そう。問題がなければ、引用されたもののうち、不要な部分をすべて消して、返事を書き加える。



書き終わったら、 Mule を終了し、セーブする。すると、メールの確認画面になる。



後は、普通に送るときと全く一緒で、 [スペース] を押して確認していく、ファイルの最後にたどり着くと、送るかどうか聞かれるので、「y」と答える。

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、電子メール、ネット・ニュースの使い方や、WWWへのアクセス方法を紹介してきた。今回は、その他のUNIXシステム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。それに関係したコマンドを紹介する。

14.1.1 誰がloginしているのか？

今、誰がloginしているかを調べるコマンドはいくつかある。

誰が何をしているのか?: w

一つ目は、wだ。wには「who and what」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。

単にwと打ってみよう。

```
% w
5:59PM up 1 min, 1 users, load averages: 0.28, 0.10, 0.04
USER     TTY FROM           LOGIN@ IDLE WHAT
ei99000  p1 :0.0          5:59PM    - w
%
```

これ表示の意味は、ei99000さんがいて、その人の端末番号はp1、このマシンのコンソールから(:0.0)からloginしている。そしてloginを始めたのは5:59PMで、現在wコマンドを実行しているということだ。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。

誰が login しているのか?: who

もう一つのコマンドは、前にも紹介したが、 who だ。同じように実行してみよう。

こちらの方は、現在何を実行しているかは表示されない。

```
% who
ei99000  tttyp1  Jan 14 17:59  (:0.0)
%
```

14.1.2 より詳しいユーザ情報を調べる: finger

ここまでところは、単に今 login している人の情報を表示するだけだったが、それ以外のユーザの情報を表示することができる。

finger は、マシン(ちなみに他のマシンも)のユーザの情報を表示することができる。まず、とりあえず使って見よう。

```
% finger
Login      Name          TTY  Idle  Login Time   Office      Office Phone
ei99000
%
%
```

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 の真価は、今 login していないユーザでも、他のマシンでも(ただし、そのマシンが動いていないといけない)調べられることにある。

まず、今、 login していないユーザの情報だが、 finger 「ユーザ名」で調べられる。

```
% finger ei99000
Login: ei99000                         Name:
Directory: /home/ei99/ei99000           Shell: /usr/local/bin/tcsh
Last login Thu Jan 15 03:18 (JST) on tttyp2 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。

また、他のマシンの情報も finger 「ユーザ名」@「ホスト名」で調べられる。

ここでホスト名について簡単に説明しよう。今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。試しに調べてみよう。

```
% hostname  
pcux51.edu3.aomori-u.ac.jp  
%
```

つまり、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 edu3.aomori-u.ac.jp のような部分を「ドメイン名」という。

実はいろいろ詳しい話もあるが、「ホスト名」についてはこれくらいにしておこう。

というわけで、話を元に戻す。つまり、次のようにすると、他のマシンのユーザ情報も調べられるのだ。

```
% finger ei99000@pcux52.edu3.aomori-u.ac.jp  
[pcux52.edu3.aomori-u.ac.jp]  
Login: ei99000 Name:  
Directory: /home/ei99/ei99000 Shell: /usr/local/bin/tcsh  
Last login Thu Jan 15 03:18 (JST) on tttyp2 from :0.0  
No Mail.  
No Plan.  
%
```

14.2 他のマシンに入る: telnet

むかし、むかし、コンピュータはとても高かった。実は、君たちが使っているマシン程度のコンピュータでさえ、100万とか200万とか、あるいはもっと金を払わないと手に入れられなかつた。

その頃は、UNIX(とか、VMSとか、IBMのメイン・フレームとか、その他の大型コンピュータ)は大活躍だった。一台だけUNIX(とか、他)の入った優秀なマシンを用意しておき、後は昔のPC-98とか、Macとか、その他のマシンから入って、みんなで使うのだ。

また、現在もプロバイダのマシンなどには、当時と同様に他のマシンから入って、みんなで使っている。

では、他のマシンに入る方法を紹介しよう。他のマシンに入るには、telnetを使う。

使い方は telnet、「ホスト名」だ。たゞ、左側に適当なマシンに telnet をかけてみよう。

ID とパスワードを正しく入力すると、次のように入ることができる。

```
% telnet pcux52.edu3.aomori-u.ac.jp
Trying 172.31.67.116...
Connected to pcux52.edu3.aomori-u.ac.jp.
Escape character is '^]'.

FreeBSD (pcux52.edu3.aomori-u.ac.jp) (tty0)

login: ei99000
Password: パスワードを打つ(見えない)
Copyright (c) 1980, 1983, 1986, 1988, 1990, 1991, 1993, 1994
          The Regents of the University of California. All rights reserved.

FreeBSD 2.2.8-RELEASE (TRI_UX) #0: Tue May 20 10:45:24 GMT 1997
%
```

後は、ウィンドウを開かないプログラムは、今まで通り使うことができる。例えば、

```
% w
 5:58PM up 6 mins, 2 users, load averages: 0.16, 0.05, 0.01
USER      TTY FROM                  LOGIN@  IDLE WHAT
ei99000  p0  pcux51                5:58PM    - w
kokubo   p1  :0.0                 5:53PM     4 -csh (tcsh)
%
```

または、

```
% finger
Login      Name          TTY  Idle  Login Time  Office      Office Phone
ei99000
kokubo    A.Kokubo      p1       4  Wed   17:53
%
```

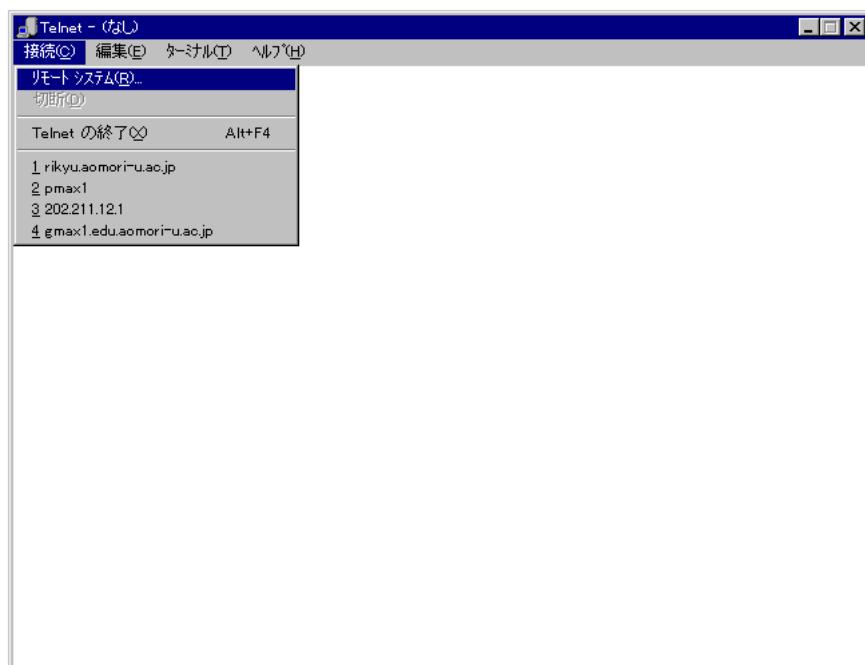
などなど。
ちなみに、抜けるには exit である。

```
% exit
logout
Connection closed by foreign host.
%
```

Windows からも telnet で入れる

Windows や Mac からも UNIX システムへは login することができる（逆は無理）。

Windows NT 4.0 の場合を紹介しておこう。まず、[スタート] の [プログラム] の [アクセサリ] の [Telnet] を選ぶ。すると、Telnet が起動するので、[接続] から、[リモート・システム] を選ぶ。



すると、次のダイアログボックスが出るので、ホスト名を入れて [接続] をクリックする。



後は、これまでと一緒である。

14.3 他のマシンの他のユーザとおしゃべり

UNIX システムでは、他のマシンの他のユーザと会話することができる（もちろん、同じマシンでも可）。

それには `talk` や `phone` を使う。

14.3.1 `talk`

`talk` の使い方は次の通りである。

`talk 「ユーザ名」@「ホスト名」`

では、実際に使ってみよう。今回は、初めてなので、2人で打ち合せてから、片方から `talk` をかけてみよう（ちなみに2人以上では同時に `talk` できない）。

たとえば、`pcux52.edu3.aomori-u.ac.jp` を使っている、`kokubo` というユーザと話をしたい場合には、次のように打つ。

```
% talk kokubo@pcux52.edu3.aomori-u.ac.jp
```

すると、自分の画面が次のようになるはずだ。

```
[No connection yet]  
[Waiting for your party to respond]
```

一方、pcux52 の kokubo さんの画面には、次のように表示される。

```
%
```

```
Message from Talk_Daemon@pcux52.edu3.aomori-u.ac.jp at 18:04 ...
talk: connection requested by ei99000@pcux51.edu3.aomori-u.ac.jp
talk: respond with: talk ei99000@pcux51.edu3.aomori-u.ac.jp
```

上の英語を大ざっぱに肝心なところだけ訳すと、次のようになる。

```
Talk_Daemon@pcux52.edu3.aomori-u.ac.jp からのメッセージ
talk: pcux51.edu3.aomori-u.ac.jp の ei99000 さんが、つなぎたいと言っています。
talk: お返事したければ talk ei99000@pcux51.edu3.aomori-u.ac.jp と打ってね。
```

というわけで、talk をかけられた人は返事をしてみよう。

言われた通りに「talk ei99000@pcux51.edu3.aomori-u.ac.jp」と打てばいい。

```
%
```

```
Message from Talk_Daemon@pcux52.edu3.aomori-u.ac.jp at 18:04 ...
talk: connection requested by ei99000@pcux51.edu3.aomori-u.ac.jp
talk: respond with: talk ei99000@pcux51.edu3.aomori-u.ac.jp
```

```
talk ei99000@pcux51.edu3.aomori-u.ac.jp
```

すると、talkかけた方の人の画面には

```
[No connection yet]  
[Waiting for your party to respond]  
[Ringing your party again]  
  
[Waiting for your party to respond]  
[Connection established]
```

返事した人の方の画面には、

```
[No connection yet]  
[Connection established]
```

となる。後は打ったものが相手の画面にでるはず。ちなみに英語しか打てないので、ローマ字か英語で会話してみよう。

終了するには、**[Ctrl]+c** をどちらかが打てばいい。

また、一方的に話すには `write 「ユーザ名」@「ホスト名」` が使える。これは、マシンの管理者が、マシンを停止するのを他のユーザに知らせたりするのに主に使うもので、普通の人があまり使うとうざったいので、注意しよう

`talk` や `write` を無視する

ちなみに、話したくもない相手から、`talk` や `write` されることだってある。そんなときは、`talk` や `write` のかかるてくるウィンドウで

```
% mesg n  
%
```

としておくとよい。

これでもう、「誰からも `talk` も `write` もかかるない静かな環境」を手に入れることができる。いつでも常に永遠にどのウィンドウでも、これを有効にしたければ、システムの初期化ファイル `.cshrc` に書いて、`login` しなおそう。

ちなみに、元に戻すには、

```
% mesg y  
%
```

と打てばいい。`.cshrc` に書いている場合には、書き直して `login` しなおせばいい。

14.3.2 3人以上で会話: phone

`talk` に似たようなコマンドに `phone` があり、これは3人以上でも会話できる。使い方は `talk` と基本的には同じで、`phone 「ユーザ名」@「ホスト名」` である。

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、次のように書くと、勝手に打った文字をひらがなに変換してくれる。

```
.phonerc —————  
set code euc
```

実際使って見ると、次のような画面になる。

The screenshot shows a terminal window titled "console". It contains two distinct session logs:

```
---- kokubo@pollux on ttym1 (アツシ コクボ) -----
やあ!
```



```
---- kokubo@deneb on ttym1 (アツシ コクボ) -----
どもども
```

3人目を呼ぶには、[Esc] を押して、コマンドモードにし、call 「ユーザ名」@「ホスト名」とコマンドを打つ。

The screenshot shows a terminal window titled "console". The user has entered the command "call" followed by the target host name:

```
Command> call kokubo@castor
```

3人目が答えると、次のようにになって、会話することができる。

The screenshot shows a terminal window titled "console". It displays three separate sessions on different hosts:

- Session 1: "kokubo@pollux on ttym1 (アツシ コクボ)" - Response: やあ!
- Session 2: "kokubo@deneb on ttym1 (アツシ コクボ)" - Response: どもども
- Session 3: "kokubo@castor on ttym1 (アツシ コクボ)" - Response: きました!

抜けるには **[Ctrl]+c** と打つ。下のように本当に抜けるか? と英語で聞かれるので、 **y** と打つて抜ける。

The screenshot shows a terminal window titled "console". It displays three separate sessions on different hosts:

- Session 1: "kokubo@pollux on ttym1 (アツシ コクボ)" - Response: やあ!
- Session 2: "kokubo@deneb on ttym1 (アツシ コクボ)" - Response: どもども
- Session 3: "kokubo@castor on ttym1 (アツシ コクボ)" - Response: きました!

At the bottom of the window, the text "Really quit?" is displayed.

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 「ユーザ名」

または、 finger 「ユーザ名」 @ 「ホスト名」

hostname : ホスト名の表示

telnet : 他のマシンへのログイン

使い方: telnet 「ホスト名」

talk : 他のユーザと会話

使い方: talk 「ユーザ名」 @ 「ホスト名」

write : 他のユーザの画面に文字を表示

使い方: write 「ユーザ名」 @ 「ホスト名」

msg : 他のユーザからのメッセージの On/Off

使い方: メッセージの Off: msg n

メッセージの On: msg y

phone : 他のユーザと会話

使い方: phone 「ユーザ名」 @ 「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。そこには、UNIX System V のマニュアルが一揃い完備しており、これが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 BSD 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 FreeBSD 上に自分で install した ASCII の日本語 pLATEX を主に使用して作成した。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

1999 年 12 月 1 日 初版 第一版

著者: 小久保 温

〒 030-0943 青森市 幸畠 2-3-1 青森大学 工学部 情報システム工学科

TEL: 0177-38-2001 (青森大学代表) / FAX: 0177-38-2030 (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

〒 030-0813 青森市 松原 2-15-2 TEL: 0177-22-1523 / FAX: 0177-75-1610

印刷: 青森コロニー印刷

〒 030-0943 青森市 幸畠 字松元 62-3 TEL: 0177-38-2021 / FAX: 0177-38-6753

A 1st Step of BSD by Atsushi Kokubo

©Atsushi Kokubo 1999, Printed in Japan.

Atsushi Kokubo
Aomori Univ. Press

A First Step of B3D

