UNIX ガイド

京都産業大学
計算機センター事務室

'94.3.23

経緯

私が初めて UNIX を触ったのは 1987 年頃だったように記憶している。今から思えば素晴らしく低いスペックのマシンで、当然ながら漢字も出ないような状態で tty 端末をつないで使っていた。

その頃の UNIX には全くと言って良いほどガイドとなるドキュメントが存在しなかった。UNIX はもともとマニュアルについては極端なリファレンス指向で、ガイド的なマニュアルは大抵の場合システムには付いてこない。

今となってもどうやらその状況は同じで、書店に行っても他のプラットフォームに比べて入門的な書籍がどうももの足りない。その状況下で 1993 年の夏に計算機センターが学内共用の UNIX マシンを導入し、運用を開始した。しかも 1994 年の春から一般学生にも授業と関わりなくアカウントを出すという。そこで授業などとは関係なく UNIX 環境を利用することになる学生のために、簡単なガイドを書かなくてはならないと思った。そうしてこのドキュメントは書かれたのだ。

このドキュメントはまだ最初の段階で、私もどのような内容にしたら実際良いのか判らずに書いた部分が多い。足りない部分もあるだろう。妥当でない説明も、くどい部分も多々あるだろう。ここは駄目だという苦情は、訂正、書換えの努力に転化して欲しい。私は自分が書いたドキュメントに何の未練もない。まずいと思うところはどんどん直して変えていって欲しいと思う。そうしてこのドキュメントが実際に役に立つものとして成熟して行けば良いなと考えている。

配布、改変

謝辞

このドキュメントを作成するに当たり、さまざまな人やものに感謝しなくてはならない。まず Mathematica の章を書いてくれた笠君に感謝。NeXT の部分を書いてくれた開原君にも感謝。付録の情報処理教室利用要項については片岡さんに感謝しなくては。このドキュメントの非常に多くの部分について添削して貰った坪内さんには大きな感謝を。同じく UNIX 初心者なのを良いことに実験台兼誤字探しを押しつけられた磯野さんにも感謝を。私がこのドキュメントを書く為に非常に多くの時間を割いたせいで、きっと他の人には負荷が掛かったと思われる。計算機センターの他の方々にも謝意を表したい。

日頃から多くのドキュメントを自宅で書き続けてきて、今回もその一部が使われることになった。結局、今回のために新たに起こした原稿のほとんども書いた事になる自宅の Mac SE に感謝。もう 1988 年からの付き合いになる。職場で原稿を書いたり挿絵を書いたりするのに活躍してくれた私の Mac IIfx にも感謝。X window, Emacs など、今まで全く私が使わなかった道具を提供してくれた DEC-3300 にも感謝。非常に気に入らないところが今だに沢山あるが、よしとしよう。TeX に関しては SPARCcenter2000 にも感謝しなくては。価格がベラボウに高くて厭味な奴だけれど、性能的には気に入った。普通なら嫌になる TeX の作業が随分楽にできた。UNIX そのものも含めて TeX, Emacs, Wnn, X などのソフトウェアを作ってくれた全ての人達に感謝。今回は xv というソフトの作者 John Bradley に特に感謝したい。

ちょっと古い話しになるけれども、1987 年に私に「UNIX をやってみないか」と勧めてくれた黒住先生、後に導入された Sun3/60 を一緒になって使い込んでいった栗原、当時の特研の指導教員だった吉川先生。皆さんにも感謝したい。

最後に。結局私は私のパートを今回ほとんど 2 週間と少しで書いた。この期間私を支えてくれたすべての人々に感謝を忘れてはいけない。

1994.3.23 計算機センター 安田 豊

目 次

第 1 章

はじめに

1.1　コンピュータ利用上の注意

まず初めに、学内のコンピュータを利用するに当たっては、個人が大学の共有資源を利用している、と言う事に留意しておいてください。つまり一つの環境に複数の人間が同居するわけですから、そこには規則、慣習、道徳、法律が存在するということです。この章での説明は、まだコンピュータについて全く予備知識のない人については実像をなさない、抽象的に過ぎる部分を含んでいると思われます。そのような場合は、コンピュータの利用に慣れて行く過程で、幾度か読み返して理解して行くのが良いでしょう。

1.1.1　規則

各設備に関してはそれらの管理者がそれぞれに利用の規則を定めているでしょう。使えるようになったからと言って自由奔放に使っていいと言うものではありません。学内のコンピュータ機器はあなたのものになった訳ではないのですから、それぞれの設備の規則を管理者に確認しながら利用してください。本文中にも様々な規則について言及するかも知れません。注意して読み進んでください。

1.1.2　慣習、道徳

大勢の人達でコンピュータを共用する（交替で使う、と言ってもいいかも知れません）のです。皆が気持ちよく利用できるようにお互いに注意してください。自分が何かをしようと思うとき、それが自分で「良い」と思われる目的の為であったとしても、それに関わる作業のために誰かが困ることがあるのではないか、誰かの迷惑になることはないか、そこに注意してください。だからと言って初心者がやたらと引込み思案になる必要もありません。誰か自分より経験があって、良く知っていそうな人に聞けば良いのです。慣習、道徳とは不文律ですので、ここではこれ以上書きません。後はあなた方自身で修得し、実践してください。

1.1.3　法律

一般社会では著作権法というものが存在します。これはコンピュータを利用する上でも適用されます。例えば著作権法によって保護されているものを勝手にコンピュータ上に持ち込んで利用した場合、罪に問われる可能性があります。新聞記事の無断転載などがこの可能性を持っています。法律違反は「知らなかった」では済まされないことを勿論知っていますね？

1.2　サポート

上記のような注意をあなたが遵守して学内のコンピュータを利用していくのであれば、以下のサポートを受けることが出来ます。

1.2.1　相談窓口

学内のコンピュータのうち、幾らかは計算機センターが管理しています。これらのコンピュータを利用するに当たって質問、要望、トラブルがあれば計算機センター窓口の担当員に尋ねるのがいいでしょう。また、学内からの電話であれば内線電話 2578、2575、2576 で連絡が付きます。

1.2.2　各種ドキュメントはどうやったら得られますか？

各種のドキュメントの入手については上記の相談窓口に問い合わせてください。

第 2 章

UNIX はいかが

UNIX は現在、大学や研究機関で最も多く利用されている利用環境の一つです。産業用にも非常に多く利用されており、今後も多方面で利用されるでしょう。京都産業大学にも UNIX 環境が複数あります。ここではその中でも計算機センターが管理、運用しているマシンで利用出来るサービスについて、その概要と利用方法を説明します。

2.1 UNIX 環境

2.1.1 計算機センター管理の UNIX 環境

計算機センターでも幾つかの UNIX コンピュータを管理しています。計算機センターが管理し、産業大学の教員、学生に提供しているコンピュータのうち、UNIX 環境のコンピュータは以下のものです。ホスト名とはネットワーク上の各コンピュータの名前です。以降各コンピュータはホスト名で表現します。

機種名	ホスト名	
SPARCcenter2000	cc2000	計算機センター 1 階に設置の Sun 社製コンピュータ。
DEC-3300	csosf01～41	理学部 4 階 21 情報処理教室に 41 台設置の DEC 社製コンピュータ。
NeXTstation	ccns001～015	計算機科学研究所 3 階 C2 情報処理教室に 15 台設置の NeXT 社製コンピュータ。

上記のコンピュータを利用するためには「ユーザ名」と「パスワード」と呼ばれるものを取得して計算機センターに利用申請をする必要があります。この申請によってあなたは「ユーザ名」でコンピュータに登録され、ようやく各コンピュータが利用可能な状態になります。この登録が行われないと、あなたは UNIX コンピュータを利用する際にコンピュータから問い合わせられる「ユーザ名」と「パスワード」に答えられず、結果としてコンピュータを利用することが出来ません。上記のコンピュータを利用するために必要なユーザ名とパスワードは一つだけです。一つのユーザ名とパスワードで上記の全てのコンピュータを利用できます。

ユーザ名とパスワード以外の要素でも上記 3 種類の UNIX コンピュータたちは密接に連係しています。この環境を「cc 環境」と呼ぶことにします。

2.1.2　ユーザ名とパスワード

ユーザ名とは「みんなで使うコンピュータ」をある人が利用する際に、誰がコンピュータを使うかを識別するために、利用者個人ごとに一意に決定されたキーワードです。コンピュータ上での利用者の名前だと認識してもいいでしょう。コンピュータ上で扱い易いように 8 文字以内の英小文字と数字で構成されています。ユーザ名は次に説明する方法によって利用者が自分で決定します。但し、教員も、学生も、京都産業大学に在籍している間はずっと同じ名前で通さなければなりません。**途中での変更は認めませんので、慎重に名前を考えてください。** ユーザ名は電子メイルを UNIX 上で利用する際の宛名にもなりますので、あなたを連想し易い名前がいいでしょう。逆に複雑な名前を付けると誰も覚えられなくなります。反社会的な名前を付けたりすると誰からも相手にされなくなったり、後で困るかも知れません。また、自分の希望している名前を既に誰かが使っていたとしたら残念ながら別の名前を考えて貰うことになります。とにかく他の人と重複しないように一意に決まっていることが重要です。

この、**学内で誰も使っていない名前をユーザ名として獲得する作業は Sign 登録作業と呼ばれており、まだ Sign 登録が済んでいない利用希望者が自分自身で行います。**（Sign 登録は京都産業大学独自のものですので、この用語は一般的ではありません。）

2.1.3　パスワードの安全対策

パスワードとは、本人確認のための秘密のキーワードです。毎回、あなたはコンピュータを利用する最初のときにユーザ名を計算機に通知します。しかし誰かほかの人があなたの名前を「騙る（かたる・他人の名前を悪用する）」かも知れません。これでは困るのでユーザ名と一対一に対応した、コンピュータと利用者本人しか知らない秘密のキーワードを用意します。ユーザ名と、それに対応するパスワードの組み合せを正しくコンピュータに伝えられた者は、その本人に違い無いと言う訳です。ユーザ名は公開のものですが、パスワードは非公開です。誰にも教えないように、知られないようにして下さい。計算機センターでもパスワードが何であったかを調べることは出来ません。忘れないように注意してください。

ところで最近パスワードを調べて悪意に満ちた事をする人がいます。このような人達のために我々が防御策を立てていかなくては成らないのは大変馬鹿馬鹿しい事ですが、それを怠って悪意に満ちた人の攻撃を受けても困ります。そのようなことがないように、まずパスワードを誰にも推測されにくいものにする事を強く推薦します。

UNIX の世界ではパスワードは 8 文字以内の英文字と数字で作りますが、例えば english とか alphabet 等の英単語は大変危険です。ローマ字の名字や名前も駄目です。単なる数字の列なども危険です。例えば az802g など、全く誰も考え付かないような文字列が最良ですが、これはなかなか覚えにくいかも知れません。例えば takotyu（たこちゅー）や harahoro（はらほろ）などのように英文辞書や辞典に出てこない単語がいいでしょう。NikkEi* （にっけいあすたりすく）などの様に途中に記号や英大文字をまぜたりするとかなり強力です。

2.1.4　Sign しよう！

Sign 登録するには幾つかの方法が提供されていますが、ここではそのうちの幾つかを示します。

- 計算機センター相談窓口で担当員に登録して貰う。
  但しこの方法だと春先など込み合う時期はかなり時間的に待たされる可能性があります。時間的に空いていそうなときに来てください。

- 2 号館の 4 階 21 情報処理教室にある DEC-3300 を利用して登録する。
  この方法だと窓口が込んでいても待ち時間無しで登録できます。

- 学内の UNIX コンピュータを利用している知人（ないしは指導教員）に手伝って貰う。
  この方法だとわざわざ 2 号館 4 階まで行かなくても作業が出来ます。cc2000 コンピュータにユーザ名 sign でリモートログインすると、DEC-3300 を利用した場合の「いよいよ Sign 登録」での作業と同じ事が出来ます。

ここでは DEC-3300 を利用して Sign 登録をする手順を紹介します。これだと誰にでも簡単に出来て、計算機センターの手間も省けますので、ぜひこちらにチャレンジしてください。

部屋に入って DEC-3300 の電源を入れる

部屋に入って電源を入れる説明については 15 ページの 2.3.1 と、その次の 2.3.2 に詳しく書いてあります。ちょっと回り道ですが、そこを参照して下さい。

いよいよ Sign 登録

電源をうまく入れることが出来たら 15 センチ四方の枠が画面中央に表示されており、その下左の方に「ユーザ名」と「パスワード」と書かれていることを確認してください。まずこの状態で `sign` とタイプ（キーボードのキーをその通りに一つずつ押す事）します。一つずつアルファベットのキーを押すたびに、そのキーに対応する文字が「ユーザ名」と書かれた部分の右横に現れます。（キーは押しっぱなしにすると繰り返して押した事になりますので、押したら指を離して下さいね。）

`sign` とタイプ出来たらリターンキーを押して下さい。リターンキーはアルファベットの並びの右端にある、ひときわ大きなキーです。「Return」と書いてあります。

もしもアルファベットのキーを打ち間違えた時は delete キー（⌫）を押して下さい。一度押すたびに一文字ずつ戻ります。

続いて `touroku` とタイプします。今度はタイプした `touroku` は「パスワード」の右側には現れません。ちょっと不安ですが、間違えないように確実にタイプして、最後にリターンキーを押してください。もししばらくして「login 情報が正しくありません」と表示されたら、`sign` か `touroku` かいずれかを打ち間違えたのです。リターンキーをもう一度押せば、また `sign` をタイプするところからやり直す事が出来ます。うまく行けばしばらくすると画面中央に 20 センチ四方の枠が現れ、そこに以下のようなメッセージが表示されていると思います。

これ以降は学生ユーザ安田さんが yasu3 というユーザ名で Sign 登録する例を用いて説明します。例を見ながら、実際にはあなた自身の情報をタイプして行って下さい。教員、大学院生、その他の人でも操作そのものは殆ど同じです。いずれにしても表示される質問に次々と答えて行くだけです。

```
### Sign up system. version 0.2 ###
 1 GAKUSEI
 2 KYOUIN
 3 INSEI
 4 Misc.
 9 exit
Which type would you want to regist (0-9) ?
```

まず、どの種類の人かを尋ねてきています。安田さんは学生ですから、`1` をタイプしてリターンキーを押しました。

```
Which type would you want to regist (0-9) ? 1
Would you want to do this many times (y/n) ?
```

ここでは繰り返して何人分も登録するかどうかを尋ねています。今回は安田さん一人分しか登録しませんからnをタイプしてリターンとしました。

```
Would you want to do this many times (y/n) ? n
User registration program by Y.Yasuda.
Typein Gakusei-Number   :
```

ここでは登録する人の学生証番号を尋ねています。安田さんの番号は473088ですから、473088とタイプしてリターンとしました。

```
Typein Gakusei-Number   :473088
Typein favorite username:
```

いよいよ希望のユーザ名を尋ねています。安田さんは素直に名前の通りyasudaが良いなと思ってyasudaとタイプしてリターンしました。

```
Typein favorite username:yasuda
 ** 'yasuda' is already used by another user.
Typein favorite username:
```

すると、上記のようなメッセージが表示されて、もう一度ユーザ名の希望を聞いてきました。つまりyasudaは既に他の人が使っているのです。仕方がありませんから安田さんは「やっさん」と呼んでねという積りでyasu3とタイプしてリターンキーを押しました。(勿論一回で誰も使っていないユーザ名を見つけられる人もいるでしょう。)

```
Typein favorite username:yasu3
Typein your full name   :
```

うまく行ったようです。今度はフルネームを尋ねています。安田さんは素直にYutaka Yasudaとタイプしてリターンしました。ここで入力した名前は電子メイルなどを送ったときに利用されます。

```
Typein your full name   :Yutaka Yasuda
keyword 'yasu3' was registered in resource list.
 **
 ** remember, your username is 'yasu3' **
 **                password is 'tdwa' **
 **
Type return key to exit.
```

うまく登録できました。ここで表示されているユーザ名(例ではyasu3)と初期パスワード(例ではtdwa)をどこかに書き留めて記録してください。後で利用申請を計算機センターに出す時と、利用者登録が済んだ後で必要になる情報です。
ひょっとするとここでも何等かの失敗があって、もう一度やり直す必要があるかもしれません。その場合は上記の例とは違って、エラーがあったからやり直してね、という表示が出ますので注意して見てください。

ユーザ名とパスワードをメモしたら、この状態を終るためにリターンキーを一度押してください。

DEC-3300 をシャットダウンして電源を切り部屋を出る

シャットダウンから電源を切って部屋を出るまでの説明については 21 ページの 2.3.9 、2.3.10、2.3.11 に詳しく書いてあります。再びちょっと回り道ですが、そこを参照して下さい。

2.1.5 ユーザ登録申請

Sign 登録が終ったら計算機センターにて自分を Sign 登録したユーザ名で登録手続きをしましょう。そのためには計算機センターが主催する「コンピュータ利用オリエンテーション」[1]に参加した際に配布された「利用登録申請書」に必要事項を記入し、計算機センター相談窓口の担当員に提出してください。提出の際に再び「騙り（かたり）」を防ぐために必ず本人が、学生証をもって来てください。提出された書類を元に、計算機センターがいつ登録作業を行うかは時期によってまちまちです。急いでいる人はいつ利用可能になるか窓口で確認しておくべきでしょう。

[1]まだ参加していない人は次の開催スケジュールを今すぐ掲示版でチェックしましょう。計算機センター相談窓口前にも掲示しています。

2.2 さあ！とにかく login

ユーザ登録が済めばいよいよコンピュータを利用することになります。UNIX コンピュータを利用する際には、先に述べたユーザ名を使って、誰がコンピュータをこれから利用するのかをその都度確認する作業があります。これを login と呼びます。また、利用が終了したときに、今まで使っていたコンピュータを解放して、他の人が使えるようにするために行う作業を logout と呼びます。login してから logout するまでの間を「セッション」などと呼ぶ場合もあります。logout する代わりに「セッションを終了する」などと表現するときもあるでしょう。

以下に具体的にそれぞれのコンピュータごとに実際の login , logout 作業について説明します。機種ごとに説明していますから、あなたが利用したいと思っている機種の節だけ読んで下さい。そこでは UNIX コンピュータの操作の基礎となる「コマンド」を入力できるようになるところまでを説明しています。

これ以降はあなたがコンピュータの操作を実際に行いながら読み進んで行くように書きます。

おっとその前にコンピュータを操作するときに必ず触らなければならない、あの「キーボード」について説明しておきます。

2.2.1 キーボード

コンピュータは殆どの場合「あれをしろ」とか「これをしろ」と、決められた文法で指示を（呪文のような）文字列としてコンピュータに入力する事によって操作します。すなわちキーボードによって文字を打ち込まなければならないわけです。

- キーボードのキーを、入力したい文字の順番に押して行くことを「タイプ」もしくは「打ち込み」などと言います。コンピュータの操作関係の説明で、「〜〜とタイプする」もしくは「〜〜と打ち込む」などと書いてあった場合は、その通りキーを押すのだなと解釈してください。
- 単語と単語を分けるために空白を一つ入力したいときはキーボード最下段の一番長いキーを押します。
- 普通にアルファベットのキーをタイプしただけだと一般的には英文字の小文字（a b c など）が入力されます。英大文字（A B C など）を入力したい場合はシフトキーを押しながらアルファベットのキーを押すことによって行います。シフトキーはキーボード上に二つあり、左側ならアルファベットの並びの最も左下にある「Z」のすぐ左、右側ならアルファベットの並びの最も右下にある「M」のいくつか右にあります。キーには「shift」などと書いてあるでしょう。キーボードによっては上矢印が書いてあったりします。
- 文字を入力していて次の行に行きたくなったり、コマンド（後述）を入力し終って実行させたいときには「リターン」キーを押します。改行キーは右側のシフトキーの真上、もしくは二段上くらいにあります。「Return」「改行」などと書いてあります。
- 打ち間違えた文字の修正には delete キーを利用します。一度 delete キーを押すと、一つ前の文字が消えます。delete キーは大抵改行 (return) キーの真上、もしくは二段上くらいにあります。一般的には「DEL」「delete」などと書いてあります。文字ではなく ⌫ や、←などと絵で表現してある場合もあります。
- UNIX で時々「コントロールC」などと言う表現を使うときがあります。表記では Cntrol - C や C-c などと書かれている事が多いようです。この場合はコントロールキーを押しながらアルファベットの C キーを押すことを意味しています。コントロールキーは、大抵アルファベットの A キーの左か、左シフトキーの下かどちらかにあります。キーには大抵「Control」「CTRL」などと書いてあります。

2.3 DEC-3300 を使ってみましょう

DEC-3300 は二号館 4 階の 21 情報処理教室に設置してあります。DEC-3300 それ自身を利用するばかりでなく、計算機センターが用意している SPARCcenter2000 や、DEC-3500 を利用する場合の窓口ともなるでしょう。

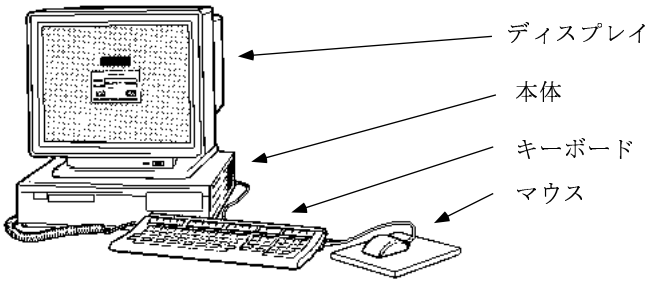

図 2.1: DEC-3300 外観

2.3.1 まず部屋へ

ともあれ、置いてあるところへ行ってみましょう。DEC-3300 は二号館 4 階の 21 情報処理教室に設置してあります。この部屋は常時カードロックシステムが働いており、学生証を入り口のドア横の機械に通すことによって鍵が開きます。入室可能な時間帯、曜日、期間に限りがありますので、付録の「情報処理教室利用要項」を一読してください。もしもカードを通しても鍵が開かない場合は、カードの向き、表裏、カードを通過させる速度（遅すぎても速すぎても駄目）を変えてやってください。どうしても駄目な場合はカードの磁気情報がおかしくなっている可能性があります。カードを通す機械にあなたのカードでなぜ鍵が開かないか、その理由が出ていますので、それを控えて計算機センターまで連絡してください。

部屋に入るときは、そこで授業をやっていないことを確認しましょう。もし授業中であれば入室して利用していいかどうか指導教員に確認を取るのが礼儀と言うものです。部屋に誰も居なかった場合は照明、空調機が切れている場合があります。これらのスイッチは部屋に入ったそのすぐ右手の壁に集中して置いてありますので、自分で適当に操作してください。部屋の管理そのものは理学部事務室が行っていますので、何か部屋の状態について質問、要望があれば理学部事務室までお願いします。コンピュータについての質問、要望は計算機センターの相談窓口までお願いします。

2.3.2 電源を入れる

自分が利用するコンピュータを選びます。これが最初の利用であれば、電源の入っていないマシンを選ぶのがいいでしょう。というのは、もし電源が入っていたら、それは誰かが既に使っているものかも知れないからです。もしディスプレイ（テレビの様な画面）の右下のスイッチが「○」の方に押し込まれていたら、まずこれを「｜」の方に押し込んでください。電源が入っていないマシンはスイッチの「｜」の方が押し込まれているのに、その横の緑色のランプが点灯していないことで識別できます。緑色のランプがどのくらい明るく点灯するか判らないので、点灯しているのかいないのか区別が付かないと言う人は入り口に最も近いマシンのランプを見てください。このマシン (csosf01) は常時電源が入っていますので、比較になるでしょう。

ランプを確認せずに、画面が真っ暗だからと言ってそのマシンの電源が切れているかどうかは判りません。DEC-3300 には、しばらく誰もキーボードを触らなかったら利用途中でも画面を真っ暗にしてしまう機能があるからです。念のためにキーボードの空白キーを一度押して、それでも画面に何の変化もないことを確認しましょう。電源が入っていないマシンが見つかったら、そのマシンの本体（ディスプレイの下敷になっている厚さ 10 センチ位の箱）の、正面に向かって左奥（背面）にある電源スイッチを入れてください。このスイッチは古典的な家庭の壁にある電灯のスイッチに似ていて、シーソーのようにパッチンと倒して入り切りします。

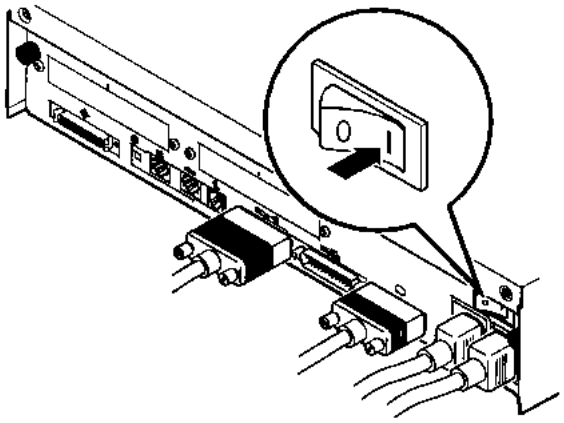

図 2.2: DEC-3300 本体背面：電源スイッチを入れる

電源が入ったらファンが回る音がして、十数秒後に「ピーポーパーポー」と[2]音がします。それから画面に色々な文字が流れていって、合計 3 分足らずで利用可能な状態になります。

2.3.3 login する

画面中央には 15 センチ四方程度の枠が表示され、その中に更に二つの小さな枠が取られ、上の枠の左に「ユーザ名」、下の枠の左に「パスワード」と表示されていると思います。キーボードを使ってまずあなたのユーザ名をタイプします。ユーザ名をタイプし終り、リターンキーを一度押せば今度はパスワードを入力できる状態になります。ここでパスワードを入力するのですが、パスワードは横から見ている他の人にバレないように、タイプしてもその文字が画面に表示されません。キーを打ち間違えないように注意して入力し、もう一度リターンキーを押します。うまくユーザ名とパスワードを入力できれば画面表示が変わります。失敗すれば「login 情報が正しくありません」と画面中央に 5 センチ四方位で表示されますのでもう一度リターンキーを押してからやり直します。

もしこれがあなたにとって Sign 登録後初めての login であればパスワードは Sign 登録時にあなたがメモしたものの筈です。もし何度試しても login 出来ないようであればユーザ名、パスワードのいずれかが間違っているか、まだあなたのユーザ登録作業が計算機センター側で完了していないのです。計算機センターに自分のユーザ名が既に登録されているかどうか確認してください。もしもユーザ名も正しく、また登録作業も完了しているなら、これはパスワードが違っているとしか考えられません。計算機センター相談窓口に連絡してパスワードを強制変更して貰ってください。

[2] 未知との遭遇？

2.3.4　ちょっとメッセージ

login すると、ときには利用者宛の通知、おしらせが以下のように画面に表示されるかも知れません。

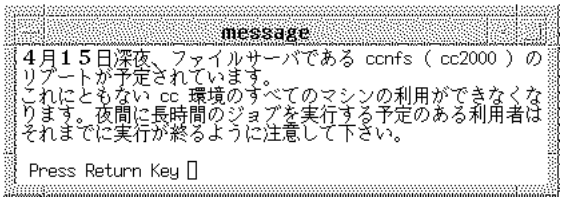

図 2.3: login 直後のメッセージ

よく読んで、自分に関係があるかどうか確認してください。確認が済めば、リターンキーを一度押すだけでこの表示を消すことが出来ます。もしも内容に付いてよく判らないことがあれば計算機センター相談窓口まで連絡するのがいいでしょう。

2.3.5　基礎知識

これから先は DEC-3300 では「ウィンドウ環境」に関する基礎知識が必要です。この章の最後の節「ウィンドウ環境」にその説明がありますので、まずそこを読んで下さい。

それから次に進みましょう。

2.3.6　ターミナルはあるかな？

login に成功すると以下のような画面表示になっていると思います。

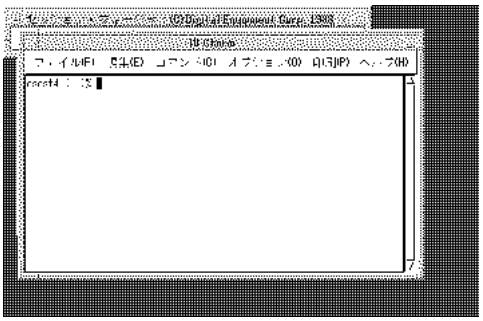

図 2.4: X の初期画面

画面の中に「DECterm」というタイトルのついた以下の様なウィンドウがあるかどうかさがしてください。そのようなウィンドウが見つかれば、ひとつ飛ばして「ターミナルは大丈夫かな？」まで進んで下さい。もしも見つからなければこのまま次に進んで下さい。

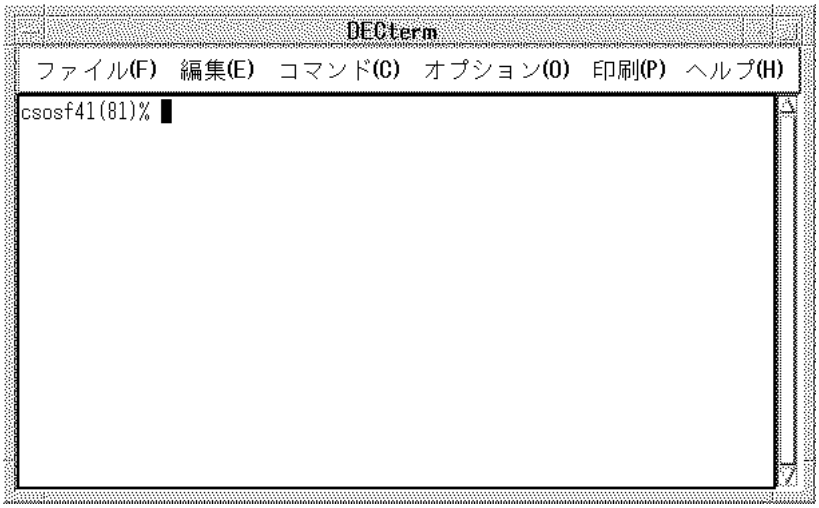

図 2.5: 漢字端末エミュレータ

ターミナルを起動する

画面左上の「セッションマネージャ」と呼ばれる以下の様なウィンドウに注目します。

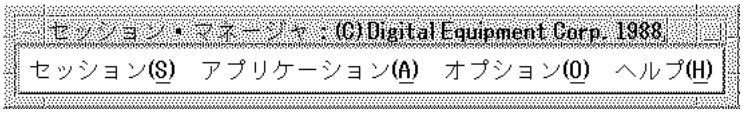

図 2.6: セッションマネージャ

このウィンドウの「アプリケーション」メニューから、「漢字端末エミュレータ」を選択します。

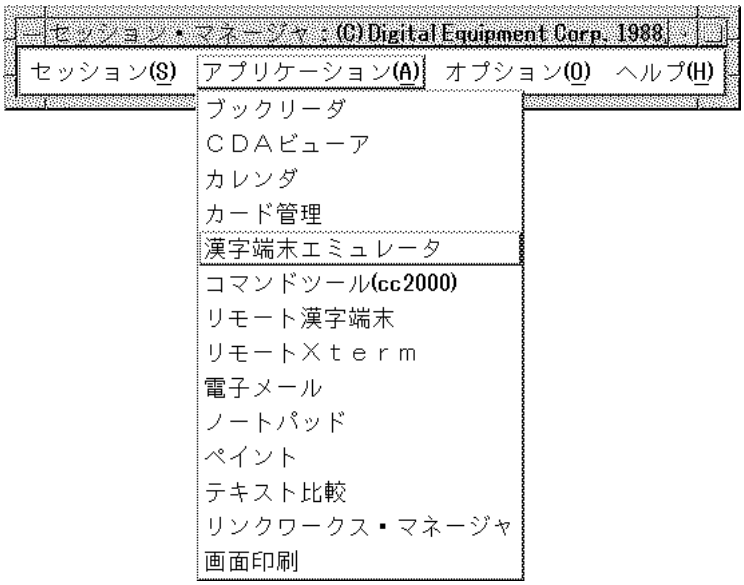

図 2.7: セッションマネージャ（アプリケーション起動）

ターミナルは大丈夫かな？

ターミナルのウィンドウの左隅には csosf01(81)% などと表示されていると思います。(但し数字は違うかもしれません。）ここで試しに date とタイプしてリターンキーを押してみてください。文字をタイプしても csosf01(81)%に続いて表示されない場合は、ターミナルウィンドウがアクティブになっていません。一度 csosf01(81)% 辺りをクリックしてターミナルウィンドウをアクティブウィンドウした後、date とタイプし直してください。以下のように今日の日付と時間が表示されれば大丈夫、ターミナルを使える状態になっています。

```
csosf01(81)% date
1994 年 02 月 27 日（日）20 時 05 分 03 秒
csosf01(82)%
```

2.3.7 パスワードを変える

もしもこれがあなたに取って Sign 登録後初めての login ならば、ここでまずパスワードを変更しましょう。Sign 登録時のパスワードは機械的に決められたもので、余り安全とは言えません。ぜひパスワードをあなた自身が選んだ単語に変更して下さい。パスワードを決める際には 10 ページの 2.1.3 で既に述べた事に注意してください。

パスワードを変更するには yppasswd コマンド[3]を利用します。上の節で説明したターミナルを見ると、

```
csosf01(81)%
```

[3]綴りがちょっと変ですが、間違ってはいません。

などとなっていますね。(但し数字は 01 と 81 ではないかもしれません。) ここで yppasswd とタイプし、リターンキーを押します。すると以下のような状態に成りますね。(以下はユーザ名 yasuda の例。yasuda の部分にはあなたのユーザ名が表示されているはずです。)

```
csosf01(81)% yppasswd
Changing NIS password for yasuda
Old NIS password:
```

指示通りここでは古い、つまり今先ほどまで使っていたパスワードをタイプし、リターンキーを押します。ここでは login の時と同じくパスワードをタイプしている間、横の人の覗き見によってあなたのパスワードがバレないように、タイプした文字は画面に表示されない事に注意してください。表示されなくともタイプした文字はちゃんと入力されていますから、安心して確実に一文字づつタイプして最後にリターンキーをタイプして下さい。すると今度は以下のようなプロンプトを表示します。

```
New password:
```

では指示通り新しい、つまり次からこれにしようと言うパスワードをタイプし、リターンキーを押してください。ここでもタイプした文字は表示されません。今度は以下のようなプロンプトを表示します。

```
Retype new password:
```

これはいましがたタイプした新しいパスワードが打ち間違いで無いことを確認するためのものです。もう一度新しいパスワードをタイプしてリターンキーを押してください。今度は以下のような状態になります。

```
NIS passwd changed on ccnic
csosf01(82)%
```

この NIS passwd changed が表示されれば成功です。もしもパスワードが短すぎたりタイプミスがあったりしたらなんらかのエラーメッセージが表示されます。途中でおかしくなったと思う場合は C-c (つまりコントロールキーを押しながらCキーを押す) で csosf(82)% を表示させます。それからもう一度 yppasswd コマンドをやり直します。

2.3.8 logout する (セッションを終了する)

他にやりたいことも色々あるでしょうが今回は最初の利用でしょうし、まず一通りの操作をやってみると言う意味で、logout をこの場でやってみることを勧めます。画面左上のセッションマネージャの「セッション」メニューから「セッション終了」を選びます。

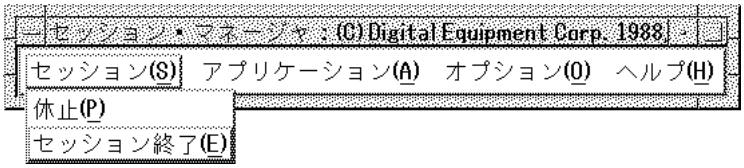

図 2.8: セッションマネージャ (セッション終了)

すると念のために以下のような問い合わせをしてきます。

この問い合わせに対してはマウスのポインタを「はい」ボタンの上に持っていって、マウスの左ボタンでクリックすることで答えます。

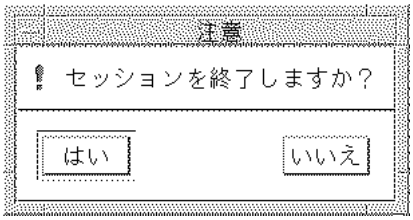

図 2.9: セッションマネージャ（終了問い合わせ１）

ひょっとすると上図のような問い合わせではなく、以下のような問い合わせになるかも知れません。

図 2.10: セッションマネージャ（終了問い合わせ２）

この問い合わせに対してはマウスのポインタを「いいえ」ボタンの上に持っていって、マウスの左ボタンをクリックすることで答えます。(もちろん将来あなたが DEC-3300 を使い込んでいって、幾らか設定変更をしたい場合はここで「はい」を選ぶことになるでしょう。)

これが DEC-3300 での logout 操作です。実際には logout という名前のコマンドも操作も実行しないことに注意してください。DEC-3300 ではつまり logout とはセッションの終了だという事です。セッション終了の操作を行なって数十秒すると、最初に電源を入れて login 操作をする直前の状態に戻ります。もしもその状態にならない、つまり正しくセッションが終了できなければ、計算機センター相談窓口まで連絡下さい。そのまま放置するのはいけません。

2.3.9 シャットダウンする

セッションが終了できれば、最初に電源を入れて login する直前の状態に戻ります。

※ 注 意 ※

本体とキーボードの間に「このマシンの本体電源は切らないでください」というようなカードが置いてあった場合は、この節と次の節は読み飛ばして 2.3.11 まで進んでください。恐らくそのマシンは誰かほかの人が夜間利用するつもりなのです。

ここで今度はユーザ名に `shut` とタイプしてリターン、パスワードに `down` とタイプしてリターンキーを押してください。それから 1 分足らずで画面が暗くなって、画面左側に >>> が表示されます。この操作をシャットダウンと呼んでいます。

>>> 表示がどうしてもでない場合は何かトラブルがあったと思われます。決して >>> が表示されていない状態で電源スイッチを切らないで下さい。また、おかしなまま放置するのもいけません。計算機センター

相談窓口まで連絡下さい。また、部屋の入り口に最も近い場所にある csosf01 と、csosf01 正面に向かって右隣にある csosf02 の二台については電源は常時入りっぱなしです。このマシンは `shut` ユーザ名ではシャットダウンできませんし、また電源も切らないでください。

2.3.10 電源を切る

>>> 表示が出たら、本体（ディスプレイの下敷になっている厚さ 10 センチ位の箱）の、正面に向かって左奥（背面）にある電源スイッチを切ります。最初に入れたときとは逆の方向に倒すわけですね。これで電源が切れます。このときディスプレイ右下のスイッチは操作しないでください。

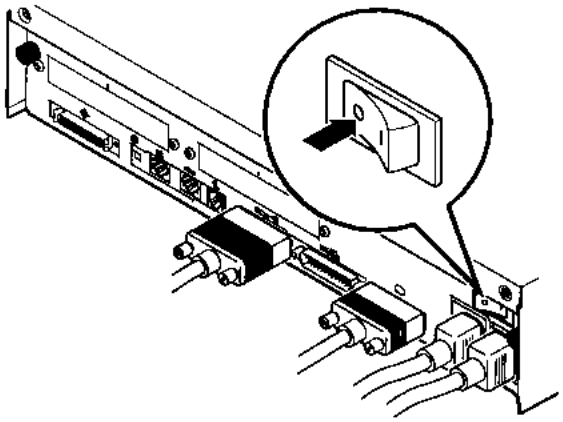

図 2.11: DEC-3300 本体背面：電源スイッチを切る

2.3.11 部屋を出る

マシンの電源が切れたら部屋を出ます。このとき、部屋に誰も残らないようなら空調機、照明を切って行きましょう。

2.3.12　さて、さて、

最初の利用はこれでおしまいです。大変よくできました。あとは精進して UNIX ユーザへの道を歩まれるわけですね。それには UNIX での一般的なコマンド操作などについて説明している、第 3 章へと進んで下さい。

2.3.13　マニュアルなど

DEC-3300 のマニュアルは部屋の入り口から最も離れた奥のロッカーに置いてあります。但し一部だけしかありませんので皆で見ることは出来ません。それより DEC-3300 それ自身を使ってマニュアルを読みましょう。17ページの 2.3.6 のすぐ次の「ターミナルを起動する」のところで紹介している「セッションマネージャ」に再び注目してください。「アプリケーション」メニューから「ブックリーダー」を選んで下さい。ブックリーダが起動され、以下のようなウィンドウが表示されるでしょう。

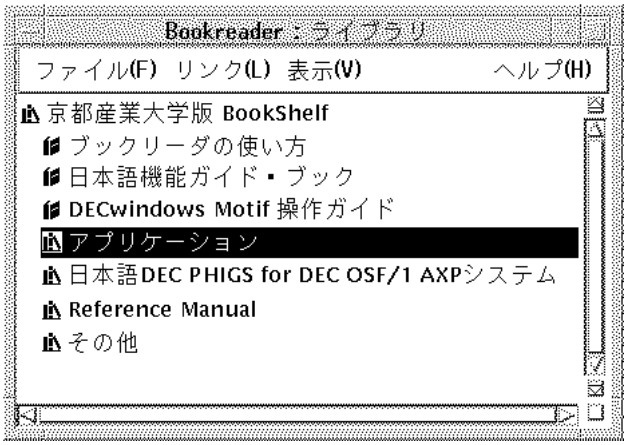

図 2.12: ブックリーダ

あとは表示されている各項目 (行) をダブルクリックすれば読み進むことが出来ます。ブックリーダそのものの使い方については上の図にある「ブックリーダの使い方」をダブルクリックすることによって知ることが出来ます[4]。

[4] と、少なくとも DEC-3300 のメーカである DEC 社は言っていますが、さてさて本当でしょうか？

2.4 NeXTStation を使ってみましょう

NeXTStation は計算機科学研究所棟 3 階の C3 情報処理教室に設置されているコンピュータです。NeXT では UNIX の環境をマウス操作やグラフィック (絵柄) を多用して、視覚的にわかりやすく利用することができます。使用できる主なソフトウェアとして、「Mathematica［マセマティカ］」（数式処理ソフト）「文机［ふづくえ］」（日本語ワープロ）「Improv［インプロブ］」（表計算ソフト）があります。また「Objective C［オブジェクティブ シー］」でオブジェクト指向のプログラミングでのアプリケーション開発ができます。

この節を読み進めるに当たって、はじめて使う人にもわかりやすいマニュアル「ようこそ NeXT...」[5] をこの文章ではときどき参照してもらうことになりますので手元においてから次へ進むとよいでしょう。

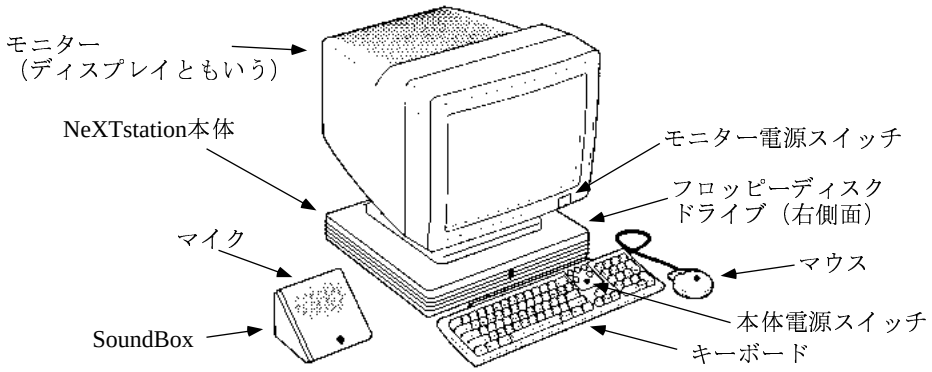

図 2.13: NeXTStation システム外観

2.4.1 まず部屋へ

まず、置いてあるところへ行ってみましょう。NeXTStation は計算機科学研究所棟の C3 情報処理教室に設置してあります。この教室は付録の「情報処理教室利用要項」に記載されている日・時間帯ならいつでも使えますが、次のことに留意してください。

部屋に入るときは、そこで授業をやっていないことを確認しましょう。もし授業中であれば入室して利用していいかどうか指導教員に確認を取るのが礼儀というものです。部屋に誰も居なかった場合は照明、空調機が切れている場合がありますので自分で適宜操作してください。照明スイッチは部屋に入ってすぐの右側、空調装置は入り口の反対側にある窓際のデスクの向こう側に設置されています。部屋そのものの管理ですが、計算機科学研究所事務室が行っていますので、何か部屋の運用について質問、要望があれば計算機科学研究所事務室までお願いします。また、コンピュータについての質問、要望については計算機センターの相談窓口でお受けしています。

2.4.2 電源を入れる

まず自分が利用するコンピュータを選びましょう。もし今回 NeXTStation を初めて使うのであれば、一連の操作に慣れるため、電源の入っていないマシンを選ぶのがいいでしょう。もし次の条件に当てはまっていれば、そのマシンの電源は入っています。

1. 画面に何か映っている。

[5] C3 情報処理教室に入って左側の書架に並んでいる、60 ページほどの薄いマニュアルです。

2. 画面は暗いが、マウスを左右に転がすと急に画面が明るくなる。

3. モニターの電源が切れているが、次に述べている方法に従ってモニターの電源を入れて 10 秒程度待つと、何かが映る。もしくは 10 秒ほど待ってマウスを転がすと画面が急に明るくなる。

NeXTStation を使うためには NeXTStation 本体の電源とモニター[6]の電源を入れなくてはなりませんが、まず最初にモニターの電源スイッチを ON にしてください。画面の右下に「○」と「｜」が付いているスイッチなのですが、「○」は「０」（ゼロ）を表し「ＯＦＦ」を意味し、「｜」は「１」を表し「ＯＮ」を意味します。今回は「｜」の方にスイッチを押し込んでください。すると、スイッチの隣の緑色の電源ランプが点灯します。次に NeXTStation 本体の電源を入れます。キーボード上の図 2.14 の位置に「Power」と書かれた 1 つだけ緑色をしたキーがありますが、これが本体の電源スイッチです。これを押すと本体の電源が入り「カチッ・ブーン」という音とともに本体が起動しはじめます。

次の段階である login の準備を整えるまでしばらくかかりますので、そのまま数分間お待ちください。

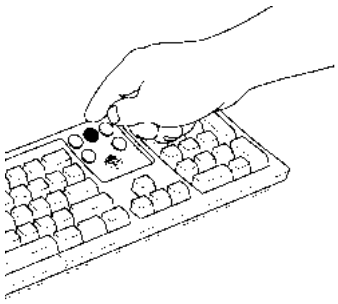

図 2.14: NeXTStation キーボード：本体電源を入れる

[6]いわゆるテレビの形をした画像を表示する装置

2.4.3 login する

画面中央には 10 センチ四方程度の枠が表示され、その中に更に二つの小さな枠が取られ、上の枠の左に「Name:」、下の枠の左に「Password:」と表示されていると思います。(図 2.15を参照) キーボードを使ってまずあなたのユーザ名をタイプします。ユーザ名をタイプし終り、リターンキー (↵ と書いてあるキーのこと) を一度押せば今度はパスワードを入力できる状態になります。ここでパスワードを入力するのですが、パスワードは横から見ている他の人にバレないように、タイプしてもその文字が画面に表示されません。キーを打ち間違えないように注意して入力し、もう一度リターンキーを押します。うまくユーザ名とパスワードを入力できれば画面表示が変わります。(図 2.16を参照) もし失敗すれば「いいえ」と首を振るように中央の枠が左右に移動します。

図 2.15: login 画面 :

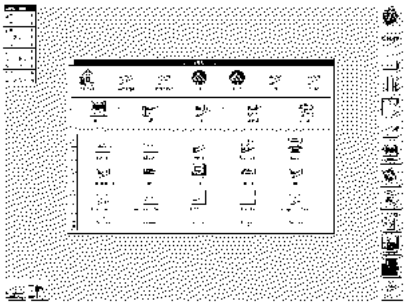

図 2.16: login が成功すると…

もしこれがあなたにとって Sign 登録後初めての login であればパスワードは Sign 登録時にあなたがメモしたものの筈です。何度試しても login 出来ないようであればユーザ名、パスワードのいずれかが間違っているか、まだあなたのユーザ登録作業が計算機センター側で完了していないのです。計算機センターに自分のユーザ名が既に登録されているかどうか確認してください。もしもユーザ名も正しく、また登録作

業も完了しているなら、これはパスワードが違っているとしか考えられません。計算機センター相談窓口に連絡してパスワードを強制変更して貰ってください。

2.4.4 ちょっとメッセージ

login すると計算機センターからのお知らせが表示されているかもしれません。よく読んで、自分に関係があるかどうか確認してください。もしも内容に付いてよく判らないことがあれば計算機センター相談窓口まで連絡するのがいいでしょう。

2.4.5 基礎知識

NeXT を便利に使う上で知っておいた方がいいことがいくつかあります。前述のマニュアル「ようこそ NeXT...」の「マウスの使用法」と「ウィンドウ」のところを読めば、この後の文章もわかりやすくなりますので、ぜひそちらの方を一度読んでみてください。

2.4.6 パスワードを変える（NeXT に初めて触れるなら…）

もしもこれがあなたにとって Sign 登録後初めての login ならば、ここでまずパスワードを変更しましょう。Sign 登録時のパスワードは機械的に決められたもので、余り安全とは言えません。ぜひパスワードをあなた自身が選んだ単語に変更してください。パスワードを決める際には 10 ページの 2.1.3 で既に述べた事に注意してください。また、より高い安全性のためパスワードを定期的に変更することをおすすめします。ただ、自分でパスワードをどのように変えたかを忘れないようにしてください。

NeXT でパスワードを変更するには Preferences アプリケーションのパスワードボタンを押して行います。詳しくはマニュアル「ようこそ NeXT...」の「パスワードの設定」のところを読んでください。

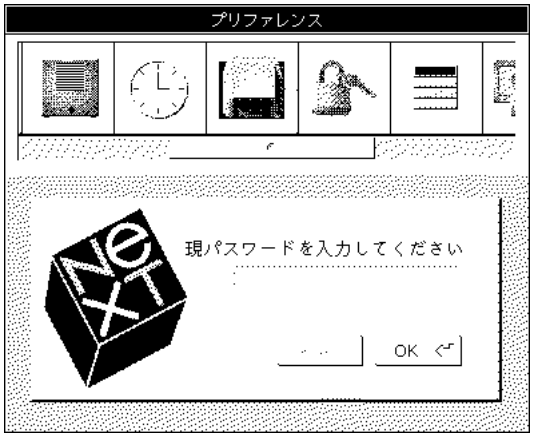

図 2.17: パスワード変更のウィンドウ

2.4.7 logout する

他にやりたいことも色々あるでしょうが今回は最初の利用でしょうし、まず一通りの操作をやってみると言う意味で、logout をこの場でやってみることを勧めます。

まず画面の左上にあるメニュー（図 2.18）の一番下にある「ログアウト」をクリックして下さい。[7]

図 2.18: ログアウトのためのメニュー

すると logout パネルが画面に現れます。

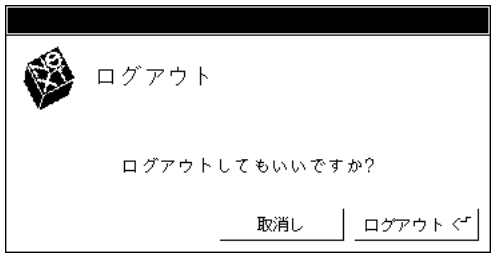

図 2.19: logout パネル

logout パネルは本当に logout してよいかどうかを聞いてきますので、ここで logout してよいならマウスのポインタを「ログアウトする」ボタンの上に持っていってマウスの左ボタンを押しましょう。

[7] もし、左上のメニューが「Workspace」ではない場合は、画面の右上にある  をクリックしてメニューを切替えて下さい。

2.4.8　電源を切る

ログアウトしてしばらく待つと再び図 2.15 の login 画面が出てきます。次に、電源を切るために最初に電源を入れたのと同じ図 2.14 のキーボード上の緑の「Power」キーを押してください。

「Power」ボタンを押すと画面に電源 OFF パネル（図 2.20）が表示されますので「電源を Off にする」ボタンを押してください。しばらくすると「NeXTstation 本体の」電源が切れます (画面が真っ暗になる)。そして本体の電源が切れたのを確認したらモニターの電源スイッチを「○」側に押して、モニターの電源を切ってください。

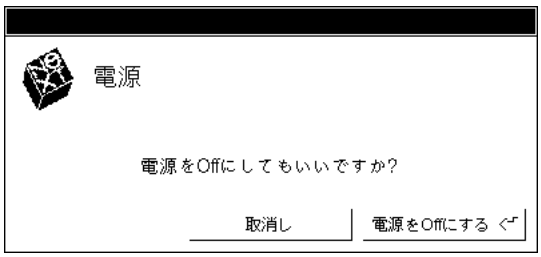

図 2.20: 電源 OFF パネル

なお、部屋の入り口を入って正面にある ccns015 と、その左側の ccns013 の 2 台については本体の電源は常に入れっぱなしにしておいてください。その 2 台に関しては誤操作を防ぐため、キーボード上の「Power」キーも効かなくしていますので、「Power」キーを押して図 2.20 の電源 OFF パネルが出てこなくても慌てないでください。

2.4.9　部屋を出る

マシンの電源が切れたら部屋を出ます。このとき、教室に誰もいなくなるようなら空調機、照明を切って行きましょう。

2.4.10　マニュアルなど

NeXTStation に関するマニュアルは、教室に入って左側の書架に各アプリケーションのものも含めてすべて置いています。どれも台数分しかありませんので教室外への持ち出しは絶対しないでください。

2.4.11　さて、それから

NeXTStation に関しては説明はここまでです。NeXTStation は、UNIX コンピュータと呼ぶにはかなり異質で、NeXT 独自の環境が用意されています。勿論普通の UNIX コンピュータとして使うことも出来ますが、ここでは説明しません。本来の NeXT コンピュータとしての利用方法については前の節で説明したマニュアルの中の、特に「ようこそ NeXT...」と「ユーザーズガイド」などを参照して下さい。

2.5 SPARCcenter2000 を使ってみましょう

SPARCcenter2000 は DEC-3300 や NeXTStation の様に、直接その前に座って利用することが出来ません。SPARCcenter2000 は、恐らく他のパソコンや UNIX コンピュータから遠隔利用される事になるでしょう。ここではまず最も SPARCcenter2000 の能力が活かせる環境として、DEC-3300 から遠隔利用する方法を紹介します。

2.5.1 DEC-3300 に login する

まず DEC-3300 に login しなければなりません。15ページの 2.3を、2.3.6まで実行してください。

ターミナルを起動する

2.3.6 のすぐ後にある「ターミナルを起動する」の手順と全く同じ様にして、今度は「漢字端末エミュレータ」の代わりに「コマンドツール (cc2000)」というメニューの項目を実行してください。SPARCcenter2000 のコマンドツールのウィンドウが一つ現れます。

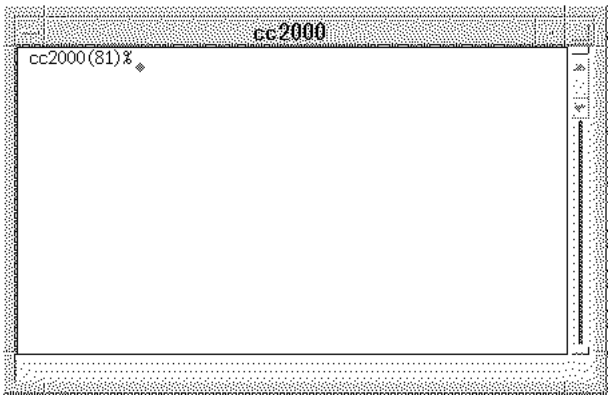

図 2.21: SPARCcenter2000 のコマンドツール

ターミナルは大丈夫かな？

コマンドツールのウィンドウの左上隅には `cc2000(81)%` などと表示されていると思います。(但し数字は違うかもしれません。) ここで試しに `date` とタイプしてリターンキーを押してみてください。文字をタイプしても `cc2000(81)%`に続いて表示されない場合は、ターミナルウィンドウがアクティブになっていません。一度 `cc2000(81)%` 辺りをクリックしてターミナルウィンドウをアクティブウィンドウした後、`date` とタイプし直してください。以下のように今日の日付と時間が表示されれば大丈夫、コマンドツールを使える状態になっています。

```
cc2000(81)% date
1994 年 02 月 27 日 (日) 20 時 05 分 03 秒
cc2000(82)%
```

2.5.2 logout する

他にいろいろやりたいこともあるでしょうが、これが最初の SPARCcenter2000 の利用です。一通りやってみると言うことから、ここで logout を試してみましょう。

DEC-3300 からコマンドツールを用いた SPARCcenter2000 の利用を終了するという事は、コマンドツールを終了するという事です。`cc2000(83)%` に続いて `exit <Return>`とするとコマンドツールは終了し、ウィンドウが消えます。

```
cc2000(82)% exit
```

特に SPARCcenter2000 に login したわけでもありませんから、ここで `logout` などとはしない事に注意して下さい。

2.5.3 DEC-3300 から logout する

更に本当に利用を終了するには 20ページの 2.3.8から 2.3.11 までを実行してください。

2.5.4 さて、さて、

これ以降の SPARCcenter2000 の使い方は UNIX での一般的なコマンド操作などについて説明している、第 3 章へと進んで下さい。

2.5.5 マニュアルなど

各情報処理教室のロッカーにはマニュアルが置いてありますが、SPARCcenter2000 のマニュアルもそのなかに含まれています。(大抵背表紙が青紫色で) 出版元が SunSoft となっているのがそれです。各機能ごとに分けられて、合計数冊あります。それぞれ一部ずつしかないので皆で同時に見ることは出来ません。うまく共有してください。

2.6 ウィンドウ環境

予備知識として最近のコンピュータでは当たり前になってきた、ウィンドウ環境の操作方法について説明しておきます。ウィンドウ環境と曖昧に呼んでいますが、厳密な呼び名はありません。最近のコンピュータはユーザの操作がやりやすい様に、画面に絵柄を表示します。つまりそれぞれの絵柄に意味を持たせ、絵で表示されるものを操作させることでコンピュータにユーザの希望を伝えようと言うわけです。勿論コンピュータがユーザに伝えたい希望（もしくは情報）も絵で表示できるわけで、要はコンピュータとユーザの対話に絵柄を利用しよう、という訳です。このアイディアはアメリカから輸入されたものですから、一般にはグラフィカルユーザインタフェイスなどと英語で呼ばれています。

今や UNIX のグラフィカルユーザインタフェイスと言えば X Window と呼ばれるシステムをベースにしたウィンドウ環境と相場が決まっています。DEC-3300 もこの X Window をベースにしたウィンドウ環境を持っています。SPARCcenter2000 も X Window 環境を利用することが出来ます。

NeXTStation は X Window ではなく、独自のウィンドウ環境を持っています[8]。

これらのウィンドウ環境でキーボードに代わってコンピュータの操作に活躍するのが次に説明する「マウス」です。

2.6.1 マウス

「マウス」とはキーボードの横に置いてある以下のようなものです。

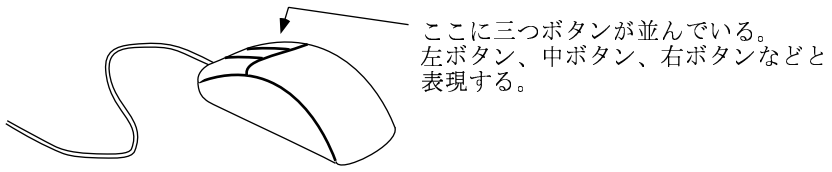

図 2.22: マウス外観

三つのボタンが上の方（奥の方）に付いており、その向こうに細い電線が付いています。ボタンの数はコンピュータの種類によってまちまちなのですが、UNIX 用では三つが一般的です。手前側を頭に見立てると、向こうのおしり側に尻尾が付いている形からネズミのつもりでマウスと呼んでいます。利き手で握って机もしくは専用の台の上を前後左右に滑らせて使います。

図 2.23: マウスの握り方と移動

初めはちょっと使いにくいと感じるでしょう。これは練習して慣れるしかありません。滑らせて動かすこ

[8] 現在の UNIX の業界では珍しいことですが、パーソナルコンピュータなども含めたコンピュータ全体で見れば、何種類ものウィンドウ環境がそれぞれ独自に存在しています。Macintosh も独自のウィンドウ環境を持っています。

とに注意して下さい。強く握って押さえつけたり、前後左右に動かす時に浮かせたりしないように注意して下さい。マウスはキーボードでは指示できない、画面に表示されている「もの」に対して指示を与えるために利用します。画面を良く見ると、マウスの前後左右の動きに連動して画面上を上下左右に動く小さな矢印（もしくは小さな✕印）が見つかるでしょう。これを指示したい対象（絵柄）の上まで運んでいって、マウスボタンの操作によって動作をその対象に対して起こさせるのです。

- 小さな矢印を「指し示すもの」という意味でポインタ、マウスポインタと呼びます。マウスポインタは状況に応じて形が変わります。例えば✕などになることもありますが、これも同じくマウスポインタと表現します。
- マウスにはボタンが（UNIX コンピュータでは大抵）三つ付いています。それぞれ左ボタン、中ボタン、右ボタンと呼んでいます。
- マウスのボタンを押しっぱなしにすることを「プレス」と呼んでいます。
- マウスのボタンを押してすぐ離すことを「クリック」と呼んでいます。
- マウスのボタンを二度続けて短い時間間隔でクリックする（要するに続けて二回クリックする）ことを「ダブルクリック」と呼んでいます。一度目のクリックと二度目のクリックの間にマウスが移動しないように注意して下さい。ボタンを強く押さえている人は力が余ってマウスも一緒に押してしまうようです。
- マウスのボタンをプレスして、そのまま移動することを「ドラッグ」と呼んでいます。目標の場所までマウスポインタが移動したら、ボタンを離します。
- 上記のクリック、ドラッグなどの操作の説明で、特にボタンを指定せずに「クリックする」と表現してある場合は左ボタンで操作を行なうことを意味しています。
- マウスを動かしていて、例えばもっと右端までポインタを移動したいのにマウスを更に右に動かすスペースがなくなってしまう時があるかも知れません。この場合は一旦マウスを持ち上げて少し左にマウスを運び、それからまたマウスを右に滑らせます。
- ポインタを選びたいものの絵柄の上まで持っていって、どのボタンでクリックすると（もしくはダブルクリックすると）どのような反応をするかは状況に応じてまちまちです。一応標準的な規約はあって、このような絵柄のものに対してクリックするとどう反応すると言う事が決まってはいるのですが、そう完全でもありません。これについては慣れて行くしかないでしょう。

2.6.2 ウィンドウ環境の画面

では、DEC-3300 の login 直後の標準的な画面表示を例に、ウィンドウ環境のものの呼び方を説明します。

ウィンドウ

なにはともあれウィンドウ（窓）です。ウィンドウ環境では一般的に以下のような四角い枠の中にさまざまな情報が表示されます。

このような枠（窓？）のことを「ウィンドウ」と呼んでいます。ウィンドウはコンピュータの画面の中に一つだけとは限りません。大抵幾つも表示させて使うことになるでしょう。つまり一つの画面の中に複数の小さな画面が幾つも作れる、と言う訳です。

図 2.24: ウィンドウの例

ウィンドウの移動

ウィンドウは画面上のどこにでも配置することが出来ます。ウィンドウの枠の上の部分を見ると、そこにはタイトルが表示されている部分があると思います。

図 2.25: ウィンドウのタイトル部分

このタイトル文字の真上にマウスポインタを移動して、そのままマウスの左ボタンを押してドラッグ（ボタンを押しっぱなしにしたまま移動）します。するとポインタの移動と共にウィンドウ、もしくはウィンドウの外枠が移動するのが判るでしょう。目的の地点までドラッグしたら、マウスのボタンを離します。

ウィンドウの上下関係を変える

ウィンドウを同時に二つ三つ出す程度ならせいぜい重ならない場所に移動するだけでいいのですが、画面の大きさには限りがあって、大抵ウィンドウは重なりあってしまいます。下の図のような状態です。

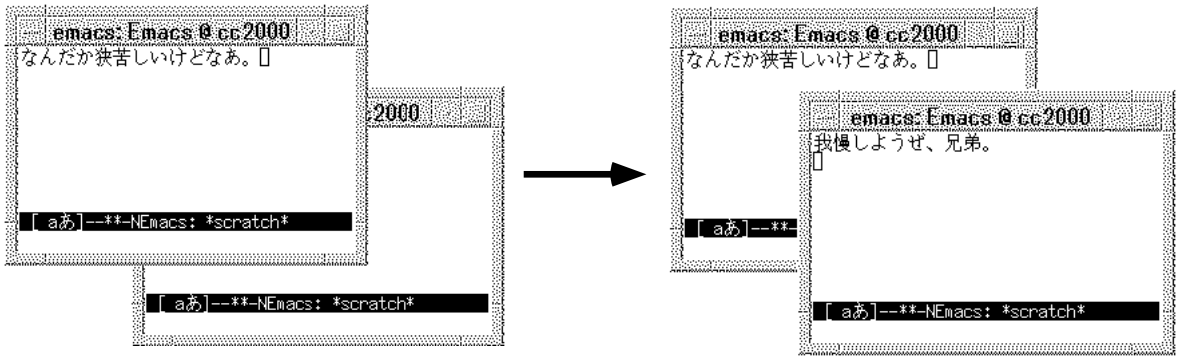

図 2.26: ウィンドウの上下関係を変える

ウィンドウの上下関係を変えたい場合は、自分が上に持ってきたいと思うウィンドウのどこかの部分をマウスの左ボタンでクリックします。簡単ですね。この図では初め左上のウィンドウが右下のウィンドウ

を下敷にしていましたが、右下のウィンドウの見えている部分をクリックすることによってその上下関係が逆転したところを示しています。

ウィンドウを選ぶ

現在のウィンドウ環境ではコンピュータに対する指示は、画面上のたった一つのもの（例えばウィンドウ）にしか同時には行なえません。つまりどのような操作にしても、その指示対象はたった一つのものに向けられているのです。そのため、常にウィンドウ環境ではたったひとつのウィンドウが指示対象として「選ばれて」います。この選ばれたウィンドウのことを「アクティブウィンドウ」と呼んでいます。「そのウィンドウを指示対象として選ぶ」ということを「そのウィンドウをアクティブにする」などともいいます。

目標のウィンドウをアクティブにするのは簡単で、単にそのウィンドウを一度マウスの左ボタンでクリックするだけです。アクティブなウィンドウは常に画面上ではもっとも上に来ています。先のウィンドウの上下関係を変える例では、アクティブなウィンドウも右側のものに切り替わっているというわけです。

2.6.3 メニュー

ウィンドウ環境で指示を与える方法として、「メニュー」によるものがあります。つまり一覧から自分の望みのものを選ぶというやり方です。勿論一覧の中に自分の望みの指示が含まれていなければいけません。メニューの形はいろいろあって、そのすべてについて網羅的に紹介することは出来ません。ただ、すべてに共通のやり方としては、項目ないしは絵柄をマウスでクリックすればそこから一覧が湧いて出て、その中から自分の望みの指示をマウスでクリックすることによって選択します。ウィンドウシステムによっては項目ないしは絵柄をプレスすればそこから一覧が湧いて出て、その中から自分の望みの指示の項目までドラッグすることによって選択する場合もあります。

ここでは例として DEC-3300 のセッションマネージャに見られるメニューを挙げます。

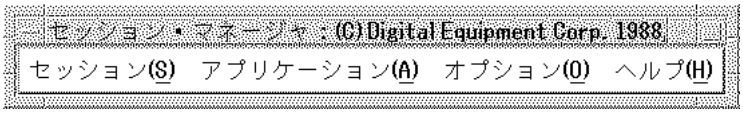

図 2.27: メニューの例：セッションマネージャ

このセッションマネージャには「セッション」「アプリケーション」「オプション」「ヘルプ」の 4 つのメニューが用意されています。例えば「セッション」の部分をマウスの左ボタンでクリックすると以下のような状態になります。

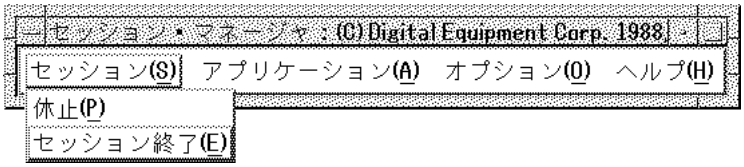

図 2.28: メニューの例：セッションマネージャのセッションメニュー

ここで例えば「セッション終了」をマウスの左ボタンでクリックすると、それはつまり「セッションメニューからセッション終了を選択し、実行する」ということを意味しています。

2.6.4　ボタン

ウィンドウ環境で指示を与える方法として「ボタン」[9]によるものがあります。つまりあるボタンを押すことによって、そこに割り当てられた動作を指示するという訳です。ボタンの形や配置はいろいろあって、そのすべてについて網羅的に紹介することは出来ません。

ここでは例として DEC-3300 のセッションマネージャに見られるボタンを挙げます。

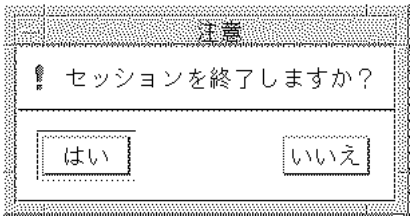

図 2.29: ボタンの例：セッションマネージャのセッション終了確認

このウィンドウには「はい」と「いいえ」の二つのボタンがあります。ここで利用者はそのどちらを実行するかをマウスの左ボタンでクリックすることによってコンピュータに指示するというわけです。

ちなみにこのウィンドウのように、ユーザに強制的に指示を要求する小さなウィンドウのことを「ダイアログ」と呼ぶ場合もあります。

2.6.5　ウィンドウ環境のトラブル傾向と対策

ここでは良く起きるトラブルの傾向とその対策を示します。但しこれですべてのトラブルが解決する訳ではありませんし、本当にコンピュータが故障したのかも知れません。ですから対処は慎重に、ゆっくり確かめながら行なって下さい。

とにかくトラブルなんですけど！

これ以降をじっくり読んで、自分が該当しているトラブルがあるかどうか調べて下さい。どうにも当てはまらない場合はまず良く知っていそうな人に聞くのがいいでしょう。それでも解決しない場合は計算機センターの相談窓口まで連絡してください。気が短い人やパーソナルコンピュータを使い慣れている人の中には突然コンピュータの電源を切ったりする人もいますが、決してそれだけはしないで下さい。**UNIX コンピュータは非常に壊れやすく、突然電源を切ったりすると次に電源を入れても二度と立ち上がらなくなることが多々あります。**

キーボードをタイプしても、目的の場所に文字が入力されない

目的のウィンドウがアクティブになっていないのではないでしょうか。目的のウィンドウを一度左ボタンでクリックして下さい。

それでも直らない場合は C-q（コントロールキーを押しながら「Q」キーを押す）してみて下さい。

キーボードをタイプしたらカタカナもしくは変な文字が入力される

キーボード上の「かな」キーを押してしまったのではありませんか？もう一度「かな」キーを押すか「英数」キーを押すなどして解除して下さい。

[9]マウスのボタンと混同しないように書かなくては！

目的のウィンドウが他のウィンドウの下敷になって見えなくなってしまった

仕方がありません。上にあるウィンドウを移動させて下敷になっているウィンドウが見えるように場所を変えます。ひょっとするとウィンドウの背景（どのウィンドウの上でもないところ）にマウスポインタを持っていって右ボタンをプレスすれば「次のウィンドウ」「前のウィンドウ」などというメニューが現れるかも知れません。それらを使って奥にあるウィンドウを前に出してくることも出来ます。

第 3 章

UNIX それから

ここでは計算機センターが管理している UNIX 環境を例に取りながら、比較的一般的な UNIX 環境の利用方法を説明します。ここでの説明は網羅的なものではなく、部分を取り上げて曖昧に説明しています。これは本文の読者のコンピュータそのものについての知識のハードルを高くしすぎないためで済ませるためです。読者が UNIX 、つまりコンピュータの利用に慣れて行くにしたがって自分でマニュアル、書籍、ネットワークなどから情報を常に取り込んで理解することが大切です。

3.1 基礎知識をもう一度

3.1.1 login

UNIX コンピュータを利用する為には、初めに誰がこの UNIX コンピュータをこれから利用するのかを、その都度確認する login と呼ばれる作業をしなければなりません。本文では既に login が済んでいるものとして説明を続けます。また、ターミナルが起動され、コマンドを入力できる状態になっていることを前提にしています。機種ごとの login の方法、ターミナルの起動の方法については第 2 章の「UNIX はいかが？」をご覧下さい。

ところで login するのは良いのですが、ときどき logout しない（もしくはセッションを終了しない）人がいます。logout しないと他の人がそのコンピュータを使えないばかりか、逆に他人に悪用されてしまいますから、しばらくしてまた戻ってくる場合でも席を外す際は logout するべきです。

3.1.2 キー表記

キーボードから様々な文字列を打ち込むことを「タイピング」と呼んでいます。「abc」と打ち込むことを「abc とタイプする」「abc と入力する」などと表現することもあります。UNIX では通常のアルファベットや数字以外に、様々なキーがあります。例えばアルファベットの A と書かれたキーを押せば、小文字の「a」が入力されるでしょう。もしも大文字の「A」をタイプしたければ「シフトキー」を押しながら A キーを押すことになります。これらのことは既に第 2 章の 2.2.1 で説明しました。ですからこれ以降は「aBc」とタイプする、と書けば「アルファベットの A キーを押して、シフトキーを押しながら B キーを押して、C キーを押す」というように解釈してください。同様にして、以下のような記述で様々なタイピングを表現します。

<Return>	リターンキーを意味します。
<Delete>	削除キーを意味します。一般的には Delete キーに当たります。
<Space>	空白キーを意味します。
<Tab>	タブキーを意味します。一般的にはアルファベットの Q の左にあります。
<ESC>	エスケープキーを意味します。一般的には数字の 1 キーの左にあります。
<ESC> X	エスケープキーを押してから X キーを押すことを意味します。
<Control> X	コントロールキーを押しながら X キーを押すことを意味します。
C-X	コントロールキーを押しながら X キーを押すことを意味します。
M-X	メタキー（一般的には Escape キーに同じ）を押してから X キーを押すことを意味します。

3.1.3 カーソル

タイピングしている最中に、次にキーを押したらどこにその文字が入力されて表示されるかを判り易くするために表示する目印のことを「カーソル (cursor)」と呼んでいます。形は状況に応じて様々で、丁度一文字分の大きさの■（黒い長方形）だったり、|（縦棒）だったりします。他の文字と見分けが付き易いように点滅している場合もあります。

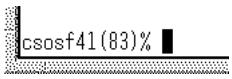

図 3.1: カーソルの例（黒い長方形）

3.2 コマンド

3.2.1 コマンドって何だ？

コンピュータと言うのは機能の集合体だと考える事が出来ます。これからあなたが利用しようと思っているコンピュータの中には星の数ほどの（いやそんなに無いな、山ほどの）機能があり、その中から利用者は自分の希望の機能を選んで実行させるのです。山ほどの機能にはそれぞれ固有の名前が付けられており、利用者は機能をその名前で選ぶと言うわけです。逆に言うと機能の名前を知らなければどんなに便利な機能がコンピュータの中にあったとしても利用できませんし、名前を間違えて指定すれば、望みの機能とは違う機能が働き出します。この「機能にそれぞれ付けられている固有の名前」をコマンドと呼んでいます。山ほどあるコマンドを利用者が覚え易いように、その機能を連想し易い名前がコマンド名として付けられています。

あるコマンドを入力するとは、そのコマンドに対応する機能を実行するように指示するという事です。

3.2.2 プロンプト

コンピュータの利用と言うのは実際にはこのコマンド実行の繰返しだと言えます。コンピュータが「コマンドをどうぞ」と言うメッセージを表示し、利用者がコマンドを入力する。コンピュータはコマンドの結果をメッセージとして表示し、「次のコマンドをどうぞ」と、またメッセージを表示する。そして利用者が再びコマンドを入力する、という具合です。見方を変えればコンピュータと利用者はメッセージとコマンドで「対話」しているようにも見えます。このコンピュータが利用者との対話のために用いている「次のコマンドは何ですか？」という催促メッセージのことを一般に「プロンプト (prompt)」と呼んでいます。

cc 環境での一般的なプロンプトは `csosf01(81)%` などと表示されます。DEC-3300 で login した直後に見える DECterm ウィンドウの左隅に見つけることが出来るでしょう。

3.2.3 簡単なコマンド

date コマンド

ここで一つ非常に簡単なコマンドを実行してみましょう。`date` コマンドです。プロンプトに続いて `date <Return>`とタイプしてください。以下のように時間が表示されるでしょう。つまり `date` は日付と時間を表示すると言う機能のコマンドです。

以下に示す例は実際にあなたが試してみたときとは結果が違うでしょう。この例を試したときと、今とでは状況が違っているからです。これ以降に挙げる例も、全く同じ結果が表示されるとは限らないことに注意してください。

```
csosf01(81)% date
1994 年 02 月 27 日（日）20 時 05 分 03 秒
csosf01(82)%
```

もしも `date` とタイプしたのにその通りに入力できない様な場合は 第 2 章の 2.6.5 をチェックして下さい。

もしも以下の様に `Command not found` などというメッセージが表示されてしまった場合はコマンドの打ち間違いです。もう一度、今度は正確にコマンドをタイプして下さい。

```
csosf01(81)% data
data: Command not found
csosf01(82)%
```

この正しくない操作を利用者が実行しようとしたためにコンピュータの方から「それは駄目だったよ」という意味のメッセージが返ってくる時があります。このようなメッセージを一般に「エラーメッセージ」と呼んでいます。

hostname コマンド

hostname コマンドで自分が使っているコンピュータのホスト名を得られます。

```
csosf01(82)% hostname
csosf01
csosf01(83)%
```

上記の例から分かるように、cc 環境ではホスト名はプロンプトに含まれていつでも表示されています。

3.2.4 引数とオプションのあるコマンド

finger コマンド

finger コマンドで、利用者の情報を得ることが出来ます。finger に続いて調べたいキーワードをタイプして<Return>です。ここでは yasuda さんについての情報を表示させてみましょう。

```
csosf01(41)% finger yasuda
Login name: yasuda                      In real life: Yutaka Yasuda
Directory: /NF/home/syokuin0/yasuda     Shell: /bin/csh
On since Mar  6 21:04:04
   on ttyp2 from ds10.kyoto-su.ac.jp
No Plan.

Login name: hayato                      In real life: YASUDA Hayato
Directory: /NF/home/g930/hayato         Shell: /bin/csh
Never logged in.
No Plan.
csosf01(42)%
```

例から判るように二件の情報が表示されました。一つはユーザ名 yasuda さんのもので、もう一つはユーザ名 hayato さんのものです。hayato さんはユーザ名ではなくフルネームの部分に指定したキーワードがマッチしたから表示されたのですね。

finger ではコマンドに続いてキーワードを指定しましたね。この様にコマンドの後ろにそのコマンドが実行するために必要な情報を付ける場合があります。このコマンドの後ろに付けるものを「引数（ひきすう）」と呼んでいます。パラメータなどと呼ぶ場合もあります。引数は常にコマンドの後ろに書きます。逆に言うとコマンドが必ず一番前に来ます。

また、コマンドの処理内容を少し変える指示を与えることが出来る場合もあります。例えば finger コマンドの表示は少し長いので、これを短くするために-s という指示をコマンドの後ろ、キーワードの前に与えます。

```
csosf01(43)% finger -s yasuda
```

```
Login       Name               TTY Idle    When            Office
yasuda   Yutaka Yasuda          p2      Sun 22:29
hayato   YASUDA Hayato             < .   .   .   . >
csosf01(44)%
```

しかしやはりユーザ名 yasuda さんの情報だけが見たいと思っています。finger コマンドにはキーワードのマッチングをユーザ名だけで行う指示を与えることも出来ます。先ほどの -s という指示に今度は -m という指示を加えます。

```
csosf01(44)% finger -s -m yasuda
Login       Name               TTY Idle    When            Office
yasuda   Yutaka Yasuda          p2      Sun 22:29
csosf01(45)%
```

なるほどユーザ名 hayato さんの情報が表示されなくなりましたね。

この -s や -m などのように、コマンドの処理内容を少し変化させるような指示を「オプション」もしくは「コマンドオプション」と呼んでいます。オプションは常にコマンドの後ろに書きます。逆に言うとコマンドが必ず一番前に来て、それからオプション、その後にその他の引数という並びになります。オプションは大抵 - 記号（引き算記号、ハイフン）に導かれています。また、オプションが複数並ぶ場合については -s -m と並べて書くことも出来ますし、縮めて -sm と書くことも出来る場合があります。つまり上記の例だと finger -sm yasuda と書いても同じ意味です。

3.2.5 対話的なコマンドとそのサブコマンド

bc コマンド

bc コマンドで、簡単な四則演算電卓を利用することが出来ます。bc <Return>でカーソルが画面の左端で止まります。そこで四則演算の式を一行入力して<Return>すれば、すぐ次の行に結果が表示されます。プロンプトは特に表示されませんが、そこでまた式を入力すればまたその結果が表示されますから計算は何度でも繰り返して行えます。

```
csosf01(65)% bc
1 + 2 * ( 3 + 4 )
15
```

この状態では、コンピュータは四則演算の式を計算する bc コマンドの最中で、それ以外の利用者の指示は受け付けられません。式以外、例えば date のような通常のコマンドをこの状態でタイプしても以下のようにエラーメッセージを表示するだけです。

```
date
syntax error on line 1 (null)
```

計算を止めてコマンドプロンプトの状態に戻りたければ quit <Return>です。

```
quit
csosf01(66)%
```

bc などのように、コマンドの中には実行すると更にユーザからの指示を待つような（つまり「対話的な」）動きをするものもあります。bc における quit などのようにコマンドの処理中にユーザがそのコマンドに対して更に細かい指示を与えるコマンドの事を「サブコマンド」と呼んだりします。

3.2.6 コマンドの使い方を調べる

さて、コマンドと一口に言っても、どうやらその機能と名前を覚えるだけでは済みそうにないことが判りますね。コマンドをどれか一つ取ってみても、それぞれ独自のオプション、引数、もしくはサブコマンドなどがあり、どのコマンドにどんなオプション等があったかを覚えないと実際困りそうです。でもそんなものいつでも全部覚えておけるほどみんな暇ではありません。その為にコマンドの正確な機能、使い方を教えてくれるコマンドがあります。man[1]コマンドです。例えば先ほど取り上げた finger コマンドのオプションなどを忘れてしまった場合にはすかさず man finger <Return>です。

```
csosf01(79)% man finger
finger(1)                                                    finger(1)
NAME
  finger, f - Displays user information
SYNOPSIS
  finger [-bfhilmpqsw] [user ...]

  The finger command displays information about the users in the passwd file.
```

というわけです。このマニュアル表示は大抵一画面では収まり切りませんから、一ページごとに一旦停止します。<Space>で一ページ分送ります。<Return>で一行分送ります。この表示が送られて行く様子を「スクロール」と呼んでいます。全てを表示し切ったら man コマンドは終了します。一旦停止している状態で、q キーを押すと最後まで表示せずにその状態で man コマンドの表示を終了させることが出来ます。h キーか、?キーのいずれかを押せば、どのようなキーを押せば一旦停止後にどのような操作が効くかを表示してくれます[2]。

man によるマニュアルの書式

man コマンドが表示する内容を良く見てみると、幾つかの項目に分けて説明してあるのが判るでしょう。finger の場合だと **NAME, SYNOPSIS, DESCRIPTION, FLAGS, EXAMPLES, FILES, RELATED INFORMATION** という具合です。上記の項目分けは、全ての UNIX において全く同じではなく、幾らか違う部分もありますが大抵は共通の形式で書かれています。以下にその項目の意味を書いておきます。

- **NAME, 名前**
  コマンドの名前とその概要。
- **SYNOPSIS, 形式**
  コマンドの形式。コマンドに適用できるオプション、引数などを列挙します。
  ここでの表記にはルールがあって、例えば finger の例を以下に挙げると、
  ```
  finger [-bfhilmpqsw] [user ...]
  ```
  この [ ] に囲まれた部分は「なくてもいいよ」つまり省略可能だという事を意味しています。特に上記の例のように [ ] に多くのオプション文字がくくられていた場合は、その中のどれでもピックアップして同時に与えていいよ、という意味です。また、 ... は、その直前のものを繰り返して書いてもいいよという意味です。つまり finger ではキーワードを複数書けるのですね。コマンドにオプションや引数をつけてタイプする場合、ここに表示された順番に注意して下さい。

[1]「マニュアル (manual)」のつもりで man です。
[2]つまりこれらは「表示一旦停止機能」のサブコマンドと言うわけです。

- **FLAG, OPTION, フラグ, オプション**
  それぞれのオプションの働きについて詳しく書いてあります。あるオプションとは相反する指示だから、これとこのオプションは同時に指定してはいけないよ、などということも書いてあります[3]。

- **DESCRIPTION, 機能説明**
  コマンドの詳細説明。コマンドの機能が詳しく書かれています。ここにオプションの説明を含めている UNIX もあります。

- **FILES, ファイル**
  コマンドに関係するファイル[4]の名前が列挙されます。

- **SEE ALSO, RELATED INFORMATION, 関連項目**
  コマンドに関連する項目。深い関係のあるコマンドなどが列挙されます。ここの項目は再び man コマンドで参照できますから、このコマンドのマニュアルだけを読んで良く判らない場合はここを追い掛けて行くのがお勧めです。

- **BUGS, バグ**
  コマンド使用上の制限事項。コマンドの動きがどうもおかしいという時は注意してみましょう。

コマンド名を調べる

コマンドの名前が判らないのだけれど、このような機能を持ったコマンドを探したい、と言う時にも man コマンドは有効です。キーワードでコマンドを検索するオプションとして-k オプションがあります。man -k password などとすれば password に関係のあるコマンドなどの一覧が表示されます。結構沢山出ますが一行で一つのコマンドを紹介してくれています。一番目の項目がコマンド名、その次の括弧に囲まれた数字がマニュアルの分類番号で、残りがコメントです。

```
csosf01(86)% man -k password
conflict (8)            - search for alias/password conflicts
lock (1)                - Requests and verifies a user password
passwd (4)              - Password files
passwd, chfn, chsh (1)  - Changes password file information
popwrd (8)              - Sets password for a POP subscriber
printpw  (8)            - Outputs the contents of the password database
pwck, grpck (8)         - Checks the password and group files for inconsistencies
yppasswd (1)            - change password in Network Information Service (NIS)
yppasswdd, rpc.yppasswdd (8)    - server daemon for modifying the Network Inform
ation Service (NIS) password file
csosf01(87)%
```

このような感じですね。上記の括弧に囲まれたマニュアルの分類番号の一覧を以下に示しておきます。この分類のことをセクションと呼んでいます。

---

[3] こういうのを排他的なオプションなどと表現することもあります。
[4] 後述。3.4 参照

セクション番号	分類
1	ユーザコマンド（一般利用者の為のコマンド）
2	システムコール（プログラム言語から利用します）
3	関数（プログラム言語から利用します）
4,5	各種ファイルフォーマット
6	ゲームとデモ
8	保守用コマンド（システム管理者が利用します）

ところでこのセクション番号の割当てですが、UNIX によってちょっと違いがあります。セクション 1, 2, 3 位まではどの機種でも同じなのですが、4, 5 あたりについては上記の表は余り当てにならないことに注意して下さい。

さて、例に挙げた password キーワードでのマニュアル検索ではセクション 1 とセクション 4 と両方に passwd という項目がありましたね。ここで `man passwd` とすると常に前の方だけ、つまりセクション 1 の方についてだけが表示されます。このような状況でセクション 4 の passwd について知りたい場合は、`man` コマンドでセクション番号を明示してやります。ここでちょっと気にしなければいけないのはセクション番号の指定の仕方が UNIX によってまちまちだと言うことです。以下に cc 環境の代表的な機種である DEC-3300 (OSF/1) の場合と SPARCcenter2000(Solaris2) の場合をそれぞれ示します。

機種	OS 名称	コマンド記述
DEC-3300	OSF/1	`man 4 passwd`
SPARCcenter2000	Solaris2	`man -s 4 passwd`

3.2.7 UNIX によるコマンドの違い

UNIX は一種類ではなく各メーカーからたくさんの種類の UNIX コンピュータが出荷されています。UNIX はそれぞれのメーカーで独自に改造され、コマンドも少しづつ動きが違います。違う UNIX を採用している場合には、コマンド名は同じでもオプションが違うと言う事は良くあることです。先述の `man` コマンドもその例です。

cc 環境は DEC-3300(OSF/1) , SPARCcenter2000(Solaris2) , NeXTStation(NEXTSTEP) の 3 種類の UNIX マシンの混成です。このドキュメントは一般的な UNIX なら大抵あてはまるようにして書いていますが、今後例示されるコマンドのうちのいくらかはその通りでは動作しない可能性があります。おかしいな、と思ったらすぐ `man` コマンドで確認する習慣を身につけましょう。

3.2.8 トラブルからの脱出

UNIX を操作していて、どうにもおかしな状態になってしまって困ることがあります。もう一度始めからやり直したいんだけど、今どういう状況なのか良く判らないなあどうしようどうしよう、、

とにかくコマンドを中断して最初からやり直したい

C-`c` を試してください。何度か C-`c` するとうまくプロンプトに戻る場合があります。

でもキー入力が全然受け付けられていないようなんですけど

C-`q` を試してください。C-`s` でキー入力をロックしてしまう場合が時々あります。C-`s` の解除が C-`q` です。C-`q` の後でならキー入力が効くのでは無いでしょうか。

正確な表現をすると C-s でロックされているのはキー入力では無く、画面表示です。つまりキー入力は受け付けられているのだけれども、その結果の表示がロックされているので、あたかもキー入力が効いていないかのように見えるのです。注意して見れば C-q の後に今までタイプしていた分と、その結果表示が一気に画面に表示されるはずです。

それでも C-c が効いてないようなんです

ええい、仕方がありません。C-z を試してください。それで Suspended などと表示されてシェルのプロンプトが表示されたらしめたものです。その状態ですぐさま kill %% とやってください。そのあとは普通にコマンドが打ち込める状態になっているのでは無いでしょうか。

画面表示がどうにもおかしくなっているんです

Emacs 利用中であれば C-l を試してください。<Control>と英字の L です。それ以外のコマンドなどで画面が乱れている場合はとりあえず C-c などしてシェルのプロンプトまで戻り、そこで tset コマンドを試してください。

それで戻らなければ login し直すのが早いですね。

でもやっぱりどうにもならないんです

仕方がありません。計算機センター相談窓口まで連絡下さい。

3.3 シェル

今までコマンドの例を挙げてきました。その時、コンピュータそれ自体が利用者からのコマンドを受け付けて逐一実行してくれているように書いてきましたが、じつはそれは正確な表現ではありません。いままで「csosf01(81)%」などのプロンプトを表示して、利用者からのコマンドを受け付けてくれていたのは「シェル」と呼ばれるプログラムだったのです。

コンピュータはいろいろなプログラムを実行できます。逆に言うと、コンピュータが実行できるのはプログラムだけです。利用者のキーボード入力からコマンドを受け付けてそれに対応するプログラム[5]を実行するための仕掛けがシェルなのです。勿論シェルもコンピュータの中ではプログラムで実現されていますから、シェルプログラムなどとも呼ばれます。とにかく利用者がキーボードからのコマンドでコンピュータを操作するときに、利用者とコンピュータを仲介してくれるプログラムなのだと考えてください。

参考

シェルは一種類ではなく、何種類もあります。一般的に UNIX でシェルと言えば csh[6]がポピュラーなのですが、cc 環境では tcsh[7] と呼ばれるシェルを標準的に採用しています。tcsh は csh の拡張版のようなもので、 基本的な動作は csh と同じだと考えてください。世間一般で売られている csh のための書籍に書いてある事項は殆どそのまま tcsh にも適用できますので、安心して csh の参考書を使って貰って結構です。

3.3.1 コマンド入力時の編集

シェルには便利な機能があって、コマンドの入力時に利用者のタイピングを手助けしてくれたりします。今までタイピングで間違えたときには、<Delete>で一文字づつ戻って間違えたところから打ち直してくれと書いてきました。確かにこれが確実な方法なのですが、しかし例えば fonger -sm yasuda とタイプし終ったところで「あっ、finger が fonger になってる！」と気が付いた場合、最初の f 以外全て打ち直しになって悲しい思いをすることになります[8]。

しかし安心してください。ここでカーソルキーを紹介しましょう。カーソルキーとはリターンキーの少し右辺りに配置してある矢印キーのことです。カーソルについては既に 3.1.3 で説明しましたが、このカーソルが左矢印（←）を一つ押すことによって一つ左に移動します。

上記の悲しい fonger の例だと、左矢印キーを十数回押して n までたどり着き、そこで <Delete>を押して o を消し、それから i を押します。これでめでたく finger -sm yasuda が完成しましたね。ここで元気良く <Return>とすれば完成したコマンドで実行が行なわれます。

カーソルは右にも動きます。例えば今まさに finger に直したのに、今回は yasuda さんではなく yasuo さんを検索する筈だった事に気が付いたとします。今度は右矢印キー（→）を押して yasuda の最後の a までたどり着き、そこで <Delete><Delete>で da を削除します。後は o をタイプして <Return>です。

カーソルを左右に動かしたりする編集キーには他にも幾つかあります。以下に一覧を載せておきます。

---

[5] つまりそれがコマンドの本体なのですね。

[6] 「しーしぇる」と読んでください。

[7] 「てぃーしーしぇる」と読んでください。

[8] 特にこのような打ち間違いをするのは初心者のうちが多いので、タイピングに慣れていない初心者にとっては非常に悲しいものです。

キー	アクション
左矢印（←）	一文字分左へ
C-b	左矢印に同じ
右矢印（→）	一文字分右へ
C-f	右矢印に同じ
C-a	コマンド行先頭（左端）へ
C-e	コマンド行末尾（右端）へ
<Delete>	カーソル位置の直前の一文字を消去
C-k	カーソル位置から末尾までを消去
C-u	コマンド行全てを消去

これでもう悲しい思いをせずに済みますね。

3.3.2 ヒストリ

コマンドを何度もタイプしていると、非常に良く似たコマンドを何度も繰り返したり場合によっては何度も同じコマンドを繰り返している事に気が付くでしょう。幾つか前に打ち込んだあのコマンドをもう一度！と思うこともあるでしょう。もっと悲しい場面としては非常に苦労してタイプした長いコマンドが、実はタイプミスを含んでいてもう一度全部タイプし直さなくてはならない、と言う場合です。

しかし安心してください。そんな時に便利なのがヒストリ（ history ）機能です。一つ前のコマンドを呼び戻すにはカーソルキーの上矢印（↑）キーを一度押します。そこで現れた一つ前のコマンドも、左矢印キーや右矢印キーでカーソルを移動させながら編集し直して実行することが出来ます。二つ前のコマンドを呼び戻すには上矢印キーをもう一度（つまり二度）押します。上矢印を押しすぎて、目標のコマンドより戻りすぎた場合は下矢印を押します。

キー	アクション
上矢印（↑）	一つ前のコマンド
C-p	上矢印に同じ
下矢印（↓）	一つ次のコマンド
C-f	下矢印に同じ

ここで history コマンドを試してください。一体どれだけのコマンドを覚えているかが判ると思います。上下の矢印キーによって、このヒストリの中を上下することが出来ると言うわけです。

```
csosf01(86)% history
    7  21:20   goto label
    8  21:20   w | grep tubo
..... （中略）
    82  21:48   finger yasuda
    83  21:48   date
    84  21:48   finger -m yasuda
    85  21:48   finger -sm yasuda
    86  21:48   history
csosf01(87)%
```

これでまた一つ悲しい思いをせずに済むようになりましたね。

3.3.3 イベント

また、history のリストを見ると、左に番号があるのが判ります。この番号はプロンプトの括弧の中に出てくる数字に一致しています。この番号で、何番目をもう一度実行せよ、という指示も出来ます。例えば 84 番目のコマンドを（もしくは 84 番目のイベントを）もう一度実行したいと言うときは !84 です。

```
csosf01(88)% !84
finger -m yasuda
Login name: yasuda                      In real life: Yutaka Yasuda
Directory: /NF/home/syokuin0/yasuda     Shell: /usr/local/bin/tcsh
On since Mar  8 21:47:43 on pts/2 from ds10.kyoto-su.ac.jp
No unread mail
No Plan.
csosf01(89)%
```

84 番目のコマンド finger -m yasuda が念のために !84 のすぐ次の行に表示されているのが判りますね。

また番号ではなく、一番最近に実行した ○×△ で始まるコマンドをもう一度実行すると言う指示も可能です。上記の例の状態で、!d とすると 83 番目の date にマッチして、それが実行されます。

```
csosf01(89)% !d
date
1994 年 02 月 28 日（月）19 時 08 分 25 秒
csosf01(90)%
```

83 番目のコマンド date が念のために !d のすぐ次の行に表示されているのが判りますね。コマンド名のマッチングの為に ! に続けて指定する文字は何文字でも構いません。書いた文字数の分だけでマッチングし、一番最近のコマンドから逆にさかのぼって一番最初にヒットした（適合した）コマンドが実行されます。もしもさかのぼってヒットするコマンドがなかった場合は「そんなイベントはない」と良う意味のエラーメッセージが表示されます。実際「イベント」という、この機能にまつわる名前を見るのはこのとき位です。

これで随分幸せになりましたね。(^_^)

3.4 ファイル

ファイルとは何かと言うことを本質的に理解できるようになるにはコンピュータの構造を知るところから始めなくてはなりません。これはファイルがコンピュータが発展する過程での歴史的経緯から発生し、現在に至っているからです[9]。

ここではファイルとは何か、と言う事については余り言及せず、その仕掛けを利用するとこんな事が出来る、というところを説明します。

3.4.1 でもやっぱりファイルって何？

気になる人の為に少し中途半端なものになるのを覚悟で「ファイル」とは何か説明しましょう。気にならない人は読み飛ばして下さい。

身のまわりの電化製品を見渡すと、ディスクやテープなどの「記録媒体」などと呼ばれるものを容易に見つけることが出来ると思います。例えば音楽を聞くのに使っているＣＤ（コンパクトディスク）や、ビデオテープなどです。前者はキラキラ光る面に溝を刻んで、後者は茶色の鉄粉が塗ってあるテープに磁石で印を付けながら、とにかく「なにか」を記録していきます。UNIX コンピュータにもこれと同じように「なにか」を記録できる「ディスク」が幾つも付いています。きっとあなたも今までにワープロ用のフロッピーや、音楽用の MiniDisk など、コンピュータ用のディスクの類似品を目にしたことがあると思います。このディスクの中にいろんなものを記録していくわけですが、コンピュータのディスクはＣＤみたいに交換できない上に、UNIX コンピュータは大勢の人が使っているので、整理して「もの」を配置しないとどこに何を記録したのか判らなくなってしまいます[10]。ＣＤだってアルバム一枚に幾つもの曲を入れていますね。そうしないと 5 曲目だけ取り出すなんていう事が出来なくて不便です。

コンピュータのディスクの中にものを記録する場合、記録はそれぞれ他の記録と混同しないように分割して残され、他の記録と区別するためにそれぞれ名前が付けられています。このそれぞれ名前を付けられたひとかたまりの情報の集まりを「ファイル」と呼んでいます。つまりコンピュータのディスクの中にはそれぞれ名前を付けられたファイルが幾つも記録されているということです。

3.4.2 ファイルの一覧を見る

UNIX に限らず現在市場に出ている殆どのコンピュータはファイルという形で情報（データなど）を格納しています。丁度様々な記録を紙に書いて、ファイル（file , 書類差し）に入れて保存するようなものです。あなたの机に文書整理用のファイルがいくつも置いてあるように[11]、コンピュータの中のあなたの記録場所にも幾つもファイルを置くことが出来ます。幾つも作ることが出来るので、それぞれを区別するために名前が付けられています。「ファイル名」などと表現します。

ファイル名には実際には殆ど全ての文字が利用できます。しかし様々な理由から、ファイル名として利用する文字としては、アルファベット大文字、小文字、数字、_（アンダースコア）、-（ハイフン）、.（ピリオド）、,（コンマ）、:（コロン）、;（セミコロン）、#（シャープ）、~（オーバライン）程度にするべきです。これら以外の記号文字、漢字などはファイル名としては利用しない方が無難です。また、ファイル名の長さは 255 文字が最大です。

`ls`[12]コマンドで、今あるファイルの名前の一覧をチェックすることが出来ます。

```
csosf01(82)% ls
```

[9] などと書いていますが、実はこの文章を書いている人が良く理解していないから説明できないのです。済みません。
[10] 長らく使っている留守録用のビデオテープのように
[11] ない人も、まああると思ってください。
[12] 字が判りにくいかも知れませんが、英小文字の L と S です。list を縮めたつもりなのです

```
Apps            Library         Mail            jsykojin.dic
csosf01(83)%
```

つまり「Apps」「Library」「Mail」「jsykojin.dic」という名前の 4 つのファイル[13]があると言うことです。

3.4.3 試しにファイルを作ってみましょう

ファイルの一つの利用方法として、あなたが実行したコマンドの記録を取る方法を紹介します[14]。`script` ファイル名 `<Return>`で、ファイル名のファイルにコマンドの実行結果が記録できます。ここでは例として時刻、実行しているマシンの UNIX の種類、ある月のカレンダーを表示させるコマンドを実行した記録を取ります。ファイル名として、とりあえず `test` を指定してみます。`script` コマンドの終了は `exit` です。つまり今回のサンプルでのタイピングは、

`script test <Return> date <Return> uname <Return> cal 7 1999 <Return> exit <Return>`

となります。

```
csosf01(83)% script test
Script started, file is test
csosf01(81)% date
1994 年 03 月 08 日（火）23 時 22 分 51 秒
csosf01(82)% uname
OSF1
csosf01(83)% cal 7 1999
          7 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
csosf01(84)% exit
csosf01(85)% Script done, file is test
```

`test` という名前のファイルが出来た事を `ls` コマンドで確認しましょう。

```
csosf01(84)% ls
Apps            Library         Mail            jsykojin.dic  test
csosf01(84)%
```

3.4.4 ファイルの内容を見る

`cat` **ファイル名** `<Return>`で、出来たファイルの内容を表示する事が出来ます。

```
csosf01(85)% cat test
Script started on Tue Mar 08 23:22:47 1994
csosf01(81)% date
```

[13] cc 環境では作った覚えがなくても、上記の 4 つのファイル程度は既にホームディレクトリ（後述）に作られています。
[14] 学生のレポート提出などに有効でしょう。

```
1994年03月08日（火）23時22分51秒
csosf01(82)% uname
OSF1
csosf01(83)% cal 7 1999
          7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
csosf01(84)% exit
csosf01(86)%
script done on Tue Mar 08 23:23:19 1994
```

このように情報をファイルにする事によって、いつでも取り出せる形で保存する事が出来るのです。一般的には、ファイルは明らかに消すと言う操作をしない限り消えることはありません。logout して、再び login しても、コンピュータの電源を切っても残っています。

ファイルの内容が長い場合

`cat` コマンドでファイルの内容を表示させた場合、その内容が長いと一画面に入り切らずにどんどんスクロールして行ってしまいます。そのような場合には `more` コマンドで一画面ごとに表示を一旦停止させる事が出来ます。`more ファイル名 <Return>`です。

一旦停止している時に、以下のキーで様々なアクションを指示出来ます。

キー	アクション
`<Space>`	一画面分スクロール
`<Return>`	一行分スクロール
f	一画面分スキップして一画面分スクロール
b	二画面分戻って一画面分スクロール
h	どのキーを押せばどんなアクションをするかを教えてくれる

3.4.5 ファイル名を変える

ファイル名は変更することが出来ます。UNIX においてファイル名を変更すると言う作業は、実はファイルを別のところへ移動すると言うことを意味します。つまり前のファイルは消えてなくなって、新しい名前で生まれ変わるという事です。コマンドは `mv`[15] です。引数が最低二つ必要で、最初の引数が元のファイル名、最後の引数が移動先のファイルです。例えば以下の例ではとりあえず `test` と付けたファイル名を `log` という名前に変えています。これはつまり `test` というファイルを `log` というファイルへ移動する、という事です。

```
csosf01(86)% mv test log
```

`test` という名前のファイルがなくなって `log` という名前のファイルが出来た事を `ls` コマンドで確認しましょう。

[15] move を縮めたつもりなのです

```
csosf01(86)% ls
Apps            Library         Mail            jsykojin.dic   log
csosf01(87)%
```

3.4.6 ファイルの複写

ファイルはその内容をそっくりそのままに複写することが出来ます。コマンドは cp[16] です。引数が最低二つ必要で、最初の引数が元のファイル名、最後の引数が複写先のファイルです。例えば以下の例では log ファイルを log2 という名前で新しく作り、内容はそっくり log ファイルから引き写しています。先の mv との違いは、元のファイルがそのまま残る (cp) か、それとも消える (mv) かです。

```
csosf01(87)% cp log log2
csosf01(88)%
```

log2 という名前のファイルが増えた事を ls コマンドで確認しましょう。

```
csosf01(88)% ls
Apps            Mail            log
Library         jsykojin.dic    log2
csosf01(89)%
```

3.4.7 ファイルの消去

ファイルは消去することが出来ます。逆に消去しない限りいつまでもそこに残っています。ファイルの置き場所は容量的には限りがあり、しかも複数の人で同じ置き場所を使っている場合が多いので、不要なファイルは削除するように心がけましょう。コマンドは rm[17] です。引数としてファイル名を与えます。例えば以下の例では log2 ファイルを削除しています。

```
csosf01(89)% rm log2
csosf01(90)%
```

log2 という名前のファイルが消えた事を ls コマンドで確認しましょう。

```
csosf01(88)% ls
Apps            Library         Mail            jsykojin.dic   log
csosf01(89)%
```

[16] copy を縮めたつもりなのです
[17] remove を縮めたつもりなのです

3.5 ファイルを編集する

今まで扱ってきたファイルは、先ほど `script` コマンドで作成したものですね。ファイルは、このようにして作成するばかりでなく、利用者が自分の好きなようにその内容を変更したり、情報を追加したり出来ます。例えば先の `script` コマンドで作成したファイルも、レポートのつもりであれば自分の学部、学科、氏名を先頭に入れたいものです。

そこでファイルを自由に編集する方法を紹介します。`emacs` です。ここでは Emacs[18] を使ってファイルを編集する作業を簡単に説明します。Emacs は全く多機能なソフトウェアで非常に多くの機能があります[19]が、ここではそれらのほとんどを説明しません。

3.5.1 Emacs での作業の流れ

初めに Emacs でファイルを編集するときの作業の流れを示しておきます。

- Emacs の起動
- ファイル名の指定
- 編集
- ファイルの保存
- Emacs の終了

Emacs ではコントロールキーを押しながらの作業が非常に多くなります。念のため今一度説明しておきます。

コントロールキーを押しながら x キーを押すことをここでは C-`x` と表記します。

エスケープキーを押してから x キーを押すことをここでは M-`x` と表記します。

3.5.2 Emacs の起動

Emacs を利用する環境には二通りあります。

A. X ウィンドウ環境が目の前にある場合。計算機センターが管理しているコンピュータ環境における具体例としては DEC-3300 の前に座って利用している場合です。

B. X ウィンドウ環境が目の前にない場合。計算機センターが管理しているコンピュータ環境における具体例としては、パソコンが置いてある情報処理教室から SPARCcenter2000 を利用している場合などです。

いずれの場合も Emacs でファイルを編集する場合のコマンド名は、`emacs` ですが、それぞれの場合に応じてすこし作法が変わります。以下に Emacs を起動するところを別々に説明します。注意深く読んでください。図 3.2 に、最初にうまく Emacs が起動できた場合の画面表示を示しておきます。

Emacs は起動すると常に最初の時点で画面内にメッセージを書いてくれています。いつでも同じメッセージですので、特に気にしないで下さい。

Emacs の画面を良く見ると、下から二行目に黒く色が反転した行がありますね。Emacs は、この行を境にして、画面をそれより上の数十行の部分とそれより下の一行の部分に分けて使います。上から順番に以

[18] 「いーまっくす」と読んでください。

[19] それを全て紹介した本の厚さは軽く 2 センチありますね。

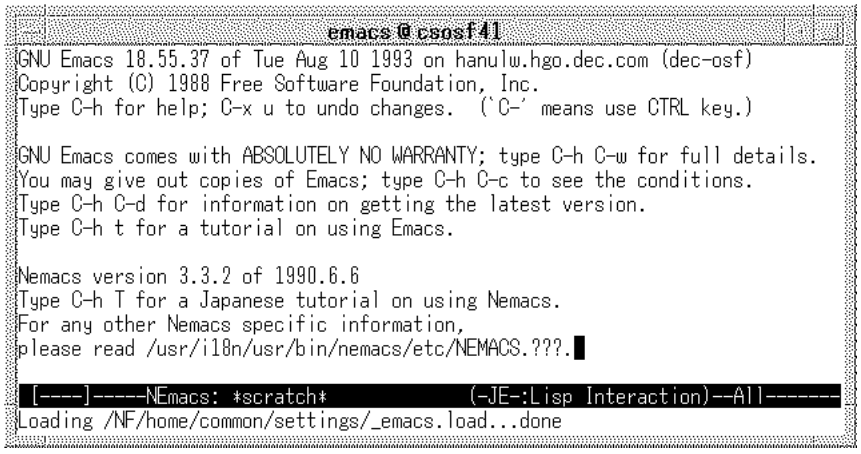

図 3.2: Emacs の起動直後

下のように呼ばれています。この呼び名は時々説明に現れますので、覚えてください。

ウィンドウ	編集するファイルの内容が表示される。(X ウィンドウなどのウィンドウと区別するために Emacs のウィンドウ、テキストウィンドウなどと表現することもあります。)
モードライン	編集しているファイルなどについての情報が表示される。常に白黒反転して表示される。
エコーライン	Emacs からのメッセージなどが表示される

3.5.3 A. の場合 : X ウィンドウ環境での Emacs の起動とエラー対策

emacs & <Return>で Emacs を起動します。

```
csosf01(85)% emacs &
[1] 9503
csosf01(86)%
```

新たなウィンドウが一つ開いて、図 3.2 のような画面表示になると思います。もしもそうならない場合は、まず <Return>を一度押して、エラーメッセージがあるかどうかをチェックして下さい。エラーメッセージを見て以下のいずれの症状が発生しているかをよくチェックして、その対処を施し、もう一度 emacs & コマンドを試みてください。

エラーメッセージ : Please set the environment variable TERM; see tset(1).

このエラーメッセージが表示された場合は以下のコマンドを実行し、再度 emacs & <Return>です。

```
csosf01(86)% setenv DISPLAY unix:0.0
```

エラーメッセージ : Xlib: connection to "unix:0.0" refused by server 他

```
Xlib:  connection to "unix:0.0" refused by server
```

```
Xlib:  Client is not authorized to connect to Server
X server not responding.  Check your DISPLAY environment variable.
```

上記のようなエラーメッセージが出た場合はちょっと問題です。本当にあなたが利用している環境は X ウィンドウ環境ですか？あながた該当するのは B. のケースではないですか？一度確認してください。B. のケースであれば B. のケースでのこのエラーの欄を参照してください。

もし本当に X ウィンドウ環境でこのようなことが起きるのなら、以下のコマンドを実行し、再度 `emacs & <Return>`です。「‘」バッククオートに気を付けてください。

```
csosf01(86)% setenv DISPLAY ‘hostname‘:0.0
```

エラーメッセージ : X server not responding. Check your DISPLAY environment variable.

上記のようなエラーメッセージが出た場合はちょっと問題です。本当にあなたが利用している環境は X ウィンドウ環境ですか？あながた該当するのは B. のケースではないですか？一度確認してください。B. のケースであれば B. のケースでのこのエラーの欄を参照してください。

3.5.4 B. の場合 : 非 X ウィンドウ環境での Emacs の起動とエラー対策

`emacs <Return>`で、Emacs を起動します。

```
csosf01(85)% emacs
```

これで図 3.2 のような画面表示になると思います。もしもこうならない場合は、以下のいずれの症状が発生しているかをよくチェックして、その対処を施し、もう一度 `emacs` コマンドを試みてください。

エラーメッセージ : Please set the environment variable TERM; see tset(1).

このエラーメッセージが表示された場合は以下のコマンドを実行し、再度 `emacs <Return>`です。

```
csosf01(86)% set term=vt100
```

エラーメッセージ : Xlib: connection to ”unix:0.0” refused by server 他

```
Xlib:  connection to "unix:0.0" refused by server
Xlib:  Client is not authorized to connect to Server
X server not responding.  Check your DISPLAY environment variable.
```

上記のエラーメッセージが表示された場合は以下のコマンドを実行し、再度 `emacs <Return>`です。上記の `"unix:0.0"` の部分が多少違っていたとしても対処は同じです。

```
csosf01(86)% unsetenv DISPLAY
```

エラーメッセージ : X server not responding. Check your DISPLAY environment variable.

上記のエラーメッセージが表示された場合は以下のコマンドを実行し、再度 `emacs <Return>`です。上記の `"unix:0.0"` の部分が多少違っていたとしても対処は同じです。

```
csosf01(86)% unsetenv DISPLAY
```

3.5.5 ファイル名の指定

Emacs が無事に起動できたら、まずは編集するファイルの名前を指定しましょう。指定した名前のファイルが存在しない場合、Emacs はその名前で新たにファイルを作成します。指定したファイルが既存のものならば、Emacs はそのファイルを読み込みます。

起動した直後の状態では、モードラインの左の部分に `*scratch*` と表示されているでしょう。ちょっと覚えておいてくださいね。

ファイルを指定するためには C-x C-f です。まずはじめに C-x を押してください[20]。数秒待つとエコーラインに `C-x-` と現れるでしょう[21]。ここで更に追い打ちを掛けるように C-f です。するとエコーラインには以下のように表示されますね。

```
Find File: ~/
```

この状態で編集したいファイル名をタイプし、そして `<Return>`です。今回は先ほど作成した `log` ファイルを編集する事にして、ファイル名のところに `log` とタイプしましょう。`<Return>`で `log` ファイルの内容が Emacs のテキストウィンドウ部分に読み込まれましたね。

さて、先ほどちょっと覚えておいてと言っていたモードラインの左の部分に `log` と表示されているのが判りますか？つまりモードラインのこの位置は、今現在編集しているファイルの名前を常に表示しているのです。最初に `*scratch*` だったのはまだ何のファイルも編集していないよと言うことを示していたのです。

3.5.6 編集

いよいよファイルの編集です。テキストウィンドウには目標のファイルが読み込まれていますね。この状態で、カーソルは恐らくテキストウィンドウの左上端だと思われます。この状態でタイピングすれば、カーソルのある位置にタイプした文字が入力されます。タイプした文字を消したい場合は `<Delete>`です。

カーソルの移動

カーソルは以下の方法でテキストウィンドウの任意の位置に移動することが出来ます。移動した先でタイプすればその文字がカーソルのある位置に入力されます。

左矢印 (←),C-b	カーソルを左に一文字移動
右矢印 (→),C-f	カーソルを右に一文字移動
上矢印 (↑),C-p	カーソルを上に一文字移動
下矢印 (↓),C-n	カーソルを下に一文字移動
C-a	カーソルを行の先頭に移動
C-e	カーソルを行の末尾に移動
M-b	カーソルを左の単語に移動
M-f	カーソルを右の単語に移動
M-a	カーソルを文の先頭に移動
M-e	カーソルを文の末尾に移動

[20] C-x の後で `<Return>`などしてはいけません。

[21] 実はこのエコーラインの表示を待つ必要はありません。慣れたら待たずに次のキーを押して下さい。

画面の移動

カーソルをどんどん下に動かして行けばいつかテキストウィンドウの下端に到達します。そこで更にカーソルを下に移動させれば画面が一ページ弱スクロールします。カーソルキー以外にも画面を移動させる方法としては、以下のキーがあります。

C-v　画面を下に（つまり次の画面に）移動
M-v　画面を上に（つまり前の画面に）移動
M-<　文頭（つまりファイルの先頭に）移動
M->　文末（つまりファイルの最後に）移動
C-l　（<Control>と英字の L です。）カーソルのある行を画面中央に持ってくるように画面を移動

行の移動

Emacs でしばらく編集していると、ある行の前後関係を入れ換えたい、もしくはある行を別の行を数行以上またいだ別の位置に移動したいと思うことがあるでしょう。このような場合は以下のステップを追うことで行の移動として実現できます。

1. 移動したい行を C-k で削除する。
2. 移動先にカーソルを移動させる。
3. C-y で 1. により削除した行を複写する。

つまり C-k を一度押すことによってカーソルの位置より右の一行分を削除することが出来ます。C-k を続けて何度か押すことによって更に下の行もまとめて削除することが出来ます。注意しなければならないのは、この削除の最中にカーソルを移動したり、何かほかの操作をしてはならないと言うことです。あくまで連続した C-k の繰返しとして実行する必要があります。

移動する行の削除が済んだら、今度は移動先の位置にカーソルを移動させます。その後で C-y とすると、先ほど削除した行がその場所に割り込むように複写されます。

このようにしておこなう行の移動は、一行でも複数行でも構いません。C-y は何度でも行えますから、行の複製としても応用できます。C-k をやったすぐ後でカーソルを全く動かさずに C-y を実行すれば削除する前の状況に戻りますから、それからカーソルを移動してまた C-y をすれば、行を違う場所に複写することにも使えます。様々な応用が利く方法ですのでぜひ覚えてください。何度か失敗するかも知れませんから、機会を見てじっくり練習するのがお勧めです。

3.5.7　ファイルへの保存

カーソルを自由に動かして自分の思うようにファイルを編集したら、ここで保存をしましょう。ファイルは Emacs に読み込まれて Emacs の中で編集されているだけで、元のファイルは全く編集されずにそのまま残っています。Emacs の中の編集された結果を元のファイルに書き込むことによって、編集の結果が反映されると言う仕掛けです。この種の操作を一般的に「保存」「セーブ (save)」と呼んでいます。

ファイルへの保存は C-x　C-s　です [22]。保存がうまく行けばエコーラインに

```
Wrote /NF/home/syokuin0/yasuda/log
```

などと表示されます。

[22] くどいですが <Return>キーなど押さずにコントロールキーを押しながら x s と押します。

3.5.8 Emacs の終了

さて、保存も済んだら Emacs を終了したいところですが、再び 3.5.2 の場合分けに戻ります。

A. の場合だと、Emacs は別の X のウィンドウとして起動されていますから、もともと Emacs を起動したシェルが動いているウィンドウは別に残っており、そこでまた別のコマンドを実行できますから、特に Emacs を終了する必要はないでしょう。シェルのウィンドウをクリックしてそちらをアクティブにするだけで良いですね。

B. の場合だと、Emacs を終らないとシェルのプロンプトが現れず[23]、次のコマンドが実行できませんから Emacs を終らなければなりません。

いずれにしても Emacs を終了するには C-x C-c です。もしもまだ保存していないファイルを編集中に Emacs を終了しようとした場合は、以下のようなメッセージがエコーラインに現れます。

```
Save file /NF/home/syokuin0/yasuda/log? (y or n)
```

編集中のファイルを保存して Emacs を終了する場合は y をタイプします。それでシェルのプロンプトが現れるでしょう。

保存しない場合は n をタイプします。この場合、Emacs は念のためにもう一度以下のような確認の問い合わせをします。

```
Modified buffers exist; exit anyway? (yes or no)
```

今度は yes とタイプします。これで編集中の内容はファイルに保存されず、シェルのプロンプトが現れるでしょう。

3.5.9 Emacs もっともっと

ここまでで非常に簡単に Emacs の使い方を紹介してきました。でもこれだけの機能で日常的にファイルを編集するのはやはり不便があると思います。

実際、Emacs にはもっともっとさまざまな機能があります。ここではさらに深く Emacs の使い方を知りたい人のためにいくつかの方法を紹介しましょう。

その前にちょっとしたテクニック

Emacs は非常に多くの<Control>キーや <ESC> キーを利用した機能があります。これらの機能を実行しようとして、キー操作を間違えてしまったり、操作の途中でおかしくなってしまった場合、下手をするとどんどんと深い失敗の谷に落ちて行くときがあります[24]。そういう事にならないためには、やはり「あぶない」と思ったときにはすぐに作業を中断するのが得策です。Emacs では作業の中断は一般的に C-g です。何が置きてもこれさえ覚えておけばもうそれ以上失敗の傷を広げることはありません。覚えておくといいでしょう。

もっと詳しいドキュメント

第 5 章「UNIX もっともっと」の 5.3 に、より詳しい Emacs の使い方についての説明がありますのでそちらを参照して下さい。また、付録の参考文献にも幾らか挙げておきますので、そちらも参照して下さい。

[23] 実はそんな事はなくて Emacs の中からシェルを呼び出すことも出来るのですが、ここでは説明しません。
[24] これが実は結構怖いんです。

漢字をタイプしたい

今まではアルファベットをタイプする方法しか説明しませんでしたから、普通の人が日常的な言葉をファイルに書き込むような事は出来ませんね。でも安心して下さい。Emacs を利用してかな漢字変換もできます。3.9 に、Emacs 上でのかな漢字変換機能についての説明がありますのでそちらを参照して下さい。

チュートリアル

Emacs には自己学習の為の機能がついています。図 3.2 に出ている表示を良く読めば判るように、Emacs を起動した状態で C-h T [25] とすれば以下のような画面表示になると思います。

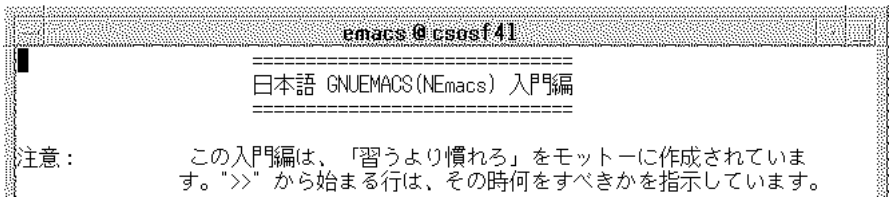

図 3.3: Emacs 入門（部分）

この入門編を読みながらそこに書かれている通りに操作をしていけば、Emacs の殆どの機能について知ることが出来ます。この入門編を終るには（勿論終り方も入門編に書いてありますが）3.5.8 を読んで下さい。入門編を始めた時のモードラインを見れば判るのですが、NEMACS.tut という名前のファイルを編集している事になっています。その為、終了作業をすると以下のように保存するかどうかを問い合わせてくる場合があります。

```
Save file /NF/home/syokuin0/yasuda/NEMACS.tut? (y or n)
```

ここで 3.5.8 の記述にしたがって保存しておくとこのファイルが残ってしまいます。特に必要でない限りはここでは保存しないのがいいでしょう。チュートリアルの最中に保存したとしても、もう要らないと思ったらこまめに `rm NEMACS.tut <Return>`で消去するのがおすすめです。

[25] T は大文字の T です。つまりシフトキーを押しながらアルファベットの T キーを押します。

3.6 印刷

UNIX 環境では様々なものをプリンタから印刷する事が出来ます。cc 環境でも、幾つかプリンタを用意しており、これは cc 環境を利用出来るコンピュータからならどこからでも誰でも利用出来るものです。

3.6.1 どんなプリンタがあるか

cc 環境から利用出来るプリンタは以下の通りです。プリンタは複数あるので、それぞれ名前が付けてあります。

プリンタ名	設置場所
ccpr01	計算機科学研究所 2 階ミニコン室（白くて小さな方）
ccpr02	計算機科学研究所 2 階ミニコン室（茶色の大きな方）
cspr01	2 号館 4 階 21 情報処理教室
clpr01	3 号館 2 階 31 情報処理教室
cepr01	5 号館 1 階 51 情報処理教室
c1kpr01	第一研究室棟 2 階共同利用室
c9pr01	9 号館

プリンタを利用する最初のときには、まずそのプリンタのある場所まで行って、実際どのプリンタに出力されるのか確認しておく事をお勧めします。上の表は恒久的なものではなく、様々な要因でプリンタは配置換えをしたり名前を変更したりされる可能性があります。

プリンタの名前はそれぞれのプリンタ自体に貼り付けてありますから、まずプリンタのところに行って、プリンタの名前を確認しておくことがトラブルを起こさないためには重要です。

3.6.2 ファイルの印刷

lpr -P プリンタ名 ファイル名 <Return>とすれば指定のプリンタに指定のファイルの内容を印刷する事が出来ます。

```
csosf01(81)% lpr -Pcspr01 log
```

-P オプションに続くプリンタ名の指定は、上記の例のようにくっつけて書いてください。例えば -P cspr01 などとしてはいけません。第二の引数となるファイル名とプリンタ名の間には一つ以上の空白を開けてください。

注意

上記の方法でファイルを印刷する場合、どんなファイルでも印刷出来るとは限りません。cat コマンドなどで内容が確認出来るような文字ばかりのファイルに限ります。それ以外の、cat したら画面に変な文字がいっぱい表示されるようなファイルは印刷しないでください。プリンタが止まってしまったり無駄に数百ページ印刷されたりします。

3.6.3 印刷状況をチェックする

lpq -P プリンタ名 <Return>で指定のプリンタの現在の状況の確認が出来ます。プリンタの状態を表すメッセージが何行か帰ってきます。

プリンタがすいている状態

以下はプリンタに何も出力待ちのものが無い場合の例です。

```
csosf01(82)% lpq -Pcspr01
csosf01: Tue Mar  8 13:48:02 1994:
no entries
csosf01(83)%
```

上記のようではなく、単に no entries メッセージだけが返ってくる場合もあります。

プリンタがなにかを印刷している状態

以下のようなメッセージが帰ってきたら、それはプリンタが何か印刷している最中か、もしくは印刷のための準備中だと言う事です。

```
csosf01(88)% lpq -Pcspr01
csosf01: Fri Mar 11 15:29:59 1994:
cspr01 is ready and printing
Rank   Owner      Job  Files                                 Total Size
active tanaka     43   sample.ps                             152 bytes
1st    yasuda     46   test.text                             8 bytes
csosf01(89)%
```

上記の例では tanaka さんの sample.ps と言うファイルが現在印刷中 (active) で、その次 (1st) の yasuda さんの test.text というファイルが印刷待ちだという事です。

ユーザ名の右に出ている 43 や 46 の数字は印刷要求それぞれに割り当てられた番号で、ジョブ番号と呼ばれています。

3.6.4 印刷の取消し

一旦 lpr コマンドでプリンタに流し込んだ出力要求を取り消して、印刷しないようにできます。**lprm -P プリンタ名 ジョブ番号 <Return>**です。一つ上の例の、yasuda さんは慌て者で、間違って別のファイルを印刷するようにコマンドを実行してしまいました[26]。この yasuda さんの印刷要求に付けられたジョブ番号は 46 です。これを取り消す例を示します。

（自分以外の印刷要求は決して取り消すことは出来ません。）

```
csosf01(99)% lprm -Pcspr01 46
benkei.kyoto-su.ac.jp: dfA046csosf01 dequeued
benkei.kyoto-su.ac.jp: cfA046csosf01.kyoto-su.ac.jp dequeued
csosf01(100)%
```

上記のメッセージは例で、実際これとはかなり違ったメッセージが表示されるかも知れません。重要なのは dequeued で、このメッセージが表示されればまず間違いなく印刷要求は取り消されています。念のためにもういちど lpq -Pcspr01 <Return>などして印刷要求が消えていることを確認するのがいいでしょう。

[26] こんな人が紙を無駄遣いするんですね。

3.6.5　利用上の注意

ちょっとひとこと

最近プリンタの紙の無駄遣いや古紙の散乱が目立ちます。プリンタに印刷しようと思ったら、紙は使い切っているわ、周りは古紙だらけで汚いわ、、では困ります。印刷するなと言うつもりはありませんが、慣れるに従ってお互い無駄な印刷はしなくて済ませる様にし、また積極的に散乱しているプリント結果の整理整頓をお願いします。

プリンタのトラブル

プリンタを使っていると、時々紙詰まり（ジャムと呼ばれる）や印刷が薄いなどのトラブルが発生します。この種のトラブルが発生したら計算機センターまで連絡下さい。また、計算機センターが配備している計算機運用補助員と呼ばれる学生が各情報処理教室を回っていますから、彼らに頼むのも良いでしょう。紙やトナー（インクのようなものです）の補給も彼らが行います。

大量の印刷をする場合

大量の印刷をする場合は他の利用者の迷惑にならないように、印刷要求の少ないときに行ってください。また、紙の補給を行わなければならない可能性のあるくらいページ数の多い印刷をしている場合は極力プリンタの見える位置で作業してください。

我々が利用しているプリンタは低速です。殆どのプリンタは毎分 6 ページ以下の印刷しか出来ません。これはつまり 60 ページのマニュアルを印刷するには 10 分以上掛かるという事です。他の人が何か印刷しようとしても、あなたのマニュアルを印刷し終わるまで 10 分も待たなくてはならないかも知れません。この点に注意して利用者みんなでうまくプリンタを共用しましょう。

ところで大量の印刷については、多少部屋が遠くても計算機科学研究所 3 階ミニコン室に設置してある ccpr02 を利用するのがお勧めです。このプリンタは最大毎分 20 ページの印刷が可能ですので、通常の 1/3 以下の時間で印刷が終了します。

3.7 状況の変化

cc 環境は全く固定的な環境では無く、常に変化しています。例えば先に挙げたプリンタの配置や名前は、新しいプリンタの導入や利用者の要求に応じて変化して行きます。このドキュメントは印刷物ですから、印刷した時点で固定されてしまいます。最近に起こった変化は吸収していない場合もあるでしょう。

それを解決するために cc 環境では ccinfo というコマンドを用意しています。これは cc 環境で配布されているドキュメントや様々な情報の最新のものを利用者が簡単に取り出せる事を目標に設定されました。

ccinfo<Return>で起動できます。以下のようなガイドメニューが表示されるでしょう。この ccinfo コマンドを起動した直後のメニューをトップメニューと呼んでいます。例えばプリンタの情報は「1. cc 環境の設備について」以下にあります。

```
csosf01(125)% ccinfo
----------------------------------------
ここでは一般的な情報の検索が可能です。
まずは分野を選んでください。
        1 cc 環境の設備について
        2 初心者向けネットワーク機能紹介
        3 ネットワークサービスに関する情報
        4 そのほかの話題
    番号を入力してください( 0 で終了します) :
```

ccinfo コマンドは基本的にはメニューに表示される項目について、その番号をタイプすることによって選択する様になっています。番号として 0（ゼロ）を入力すると一つ前のメニューに戻ります。トップメニューで 0 を入力すると ccinfo コマンドそのものを終了します。

ccinfo コマンドのメニュー内容は常に更新され、その機能も常に変化していくでしょう。しかしどのように変化したとしても、基本的にはメニューの番号を選ぶか、問い合わせに対して y または n などで返事をする事によって操作が進行するように作られています。

様々な情報が掲載されていますので、一度覗いてみることをお勧めします。

3.8　ファイルの階層構造

これまでで一般的なファイルの操作について説明してきました。ところで UNIX ではファイルは階層化されています。ここでは階層化されたファイルの概念と扱い方を説明します。

3.8.1　ディレクトリ

コンピュータをしばらく使っていると結構ファイルが増えて来て、そのうちどのファイルが何のためのものだったのか判らなくなるものです。`ls` コマンドで一覧を見たら一画面では収まり切らなくなったりして大変な状況の人も出てくるでしょう。そうならないために例えばファイルの名前を長くして、その名前を見ればファイルの内容の想像が付くようにするなど利用者は色々な対策を取ったりします。しかしそういう手法で全てが解決するわけでもありませんね。

第 2 章の 3.4 で、UNIX が扱うファイルはあなたの机の上に並ぶファイルのようなものだと言いました。もしもあなたの机の上にファイルが非常にたくさん並びはじめ、ファイルの背表紙にちょっと長めの名前を書いておくくらいでは目的のファイルがどこにあるのかすぐに探せなくなってしまったとしましょう。あなたはどうしますか？

多くの人は「整理が必要だ」と感じるでしょう。ファイルを分類し、分類ごとに大きな区分を作ってファイルを束にして置いておけば良いと言うわけです。分類とは物事を階層化して整理すると言うことです。まず大分類があって、それから中分類、更に必要なら小分類、と言う感じですね。以下に日常的な机の上の分類の例を示します。四角で分類、丸でファイルを表しています。

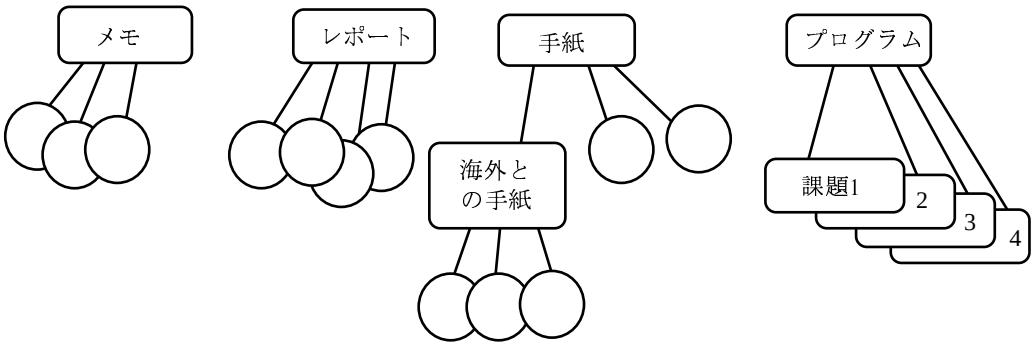

図 3.4: 階層化された分類の例 (1)

UNIX でもファイルを階層化して整理することが可能です。以下に先の例に合うような階層化を行ったファイルの配置の例を示します。

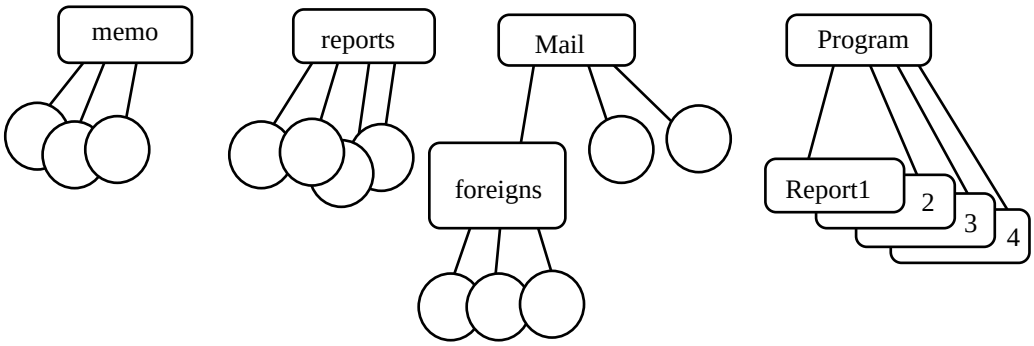

図 3.5: 階層化された分類の例 (2)

しかし実は UNIX コンピュータは yasuda さん一人のものではなくて、数多くの人が同時に使っていると言うことを忘れてはいけません。コンピュータにとっては、「 yasuda のファイル」という分類がまず最初に既に存在しているのです。

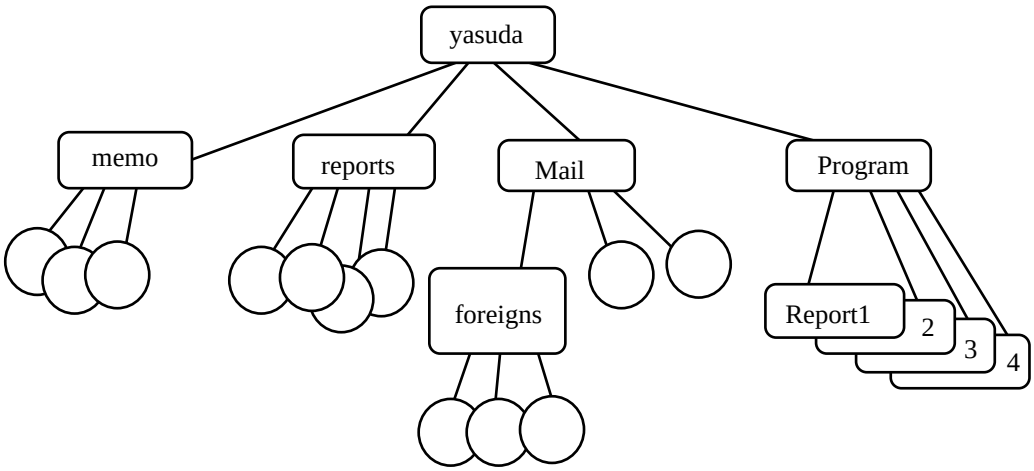

図 3.6: 階層化された分類の例 (3)

そして「yasuda のファイル」は以下のようにもっと大きな分類の下に配置されているのです。

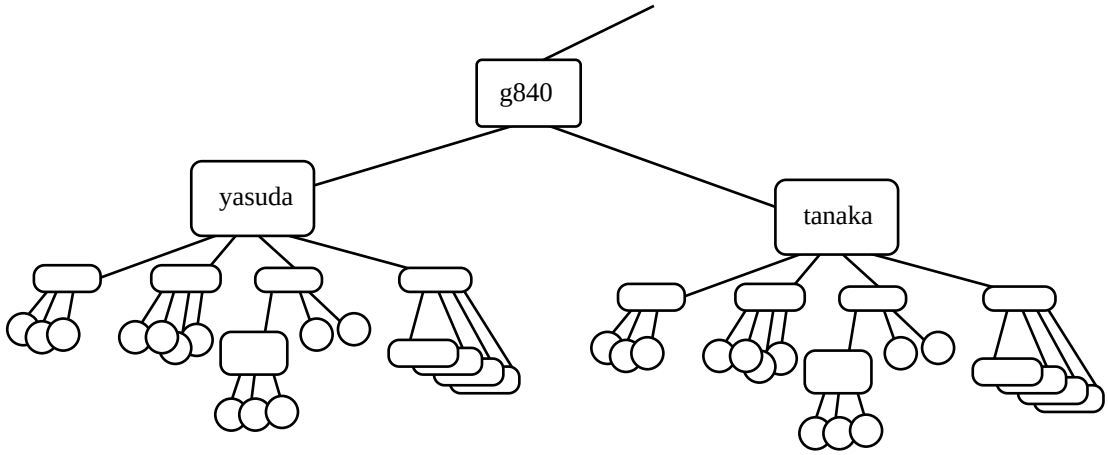

図 3.7: 階層化された分類の例 (4)

上の例での「yasuda のファイル」より一つ大きい分類の名前は「g840」です。これは「ある年度に入学してきた学生のファイル」という意味の分類です。これよりもう少し上の存在として「全ての利用者のファイル」というものが有り、更にたどって行くと最終的に「全てのファイル」という最大の分類に到達します。

先ほどからの例では四角で表してきた、ファイルの分類なるものの存在を UNIX では「ディレクトリ[27]」と呼んでいます。

UNIX では全てのファイルはたどって行けば「全てのファイル」を意味する唯一で最大の分類に到達します。この分類の根本（もしくは頂上）なるものの存在を「ルートディレクトリ[28]」もしくは「トップディレクトリ」などと呼んでいます。

「yasuda のファイル」は実は yasuda という名前が付いたディレクトリなのです。この、利用者ごとに割り当てられた利用者自身のためのディレクトリを「ホームディレクトリ」と呼んでいます。ホームディレクトリは常に利用者のユーザ名が付けられて、利用者登録の際にシステム管理者[29]の手によって作成されます。利用者は自分のホームディレクトリ以下に自分のファイルもしくはディレクトリを作り、保存することが出来ます。

いくつかのディレクトリを挙げましたが、いずれにしてもディレクトリには違い有りません。

UNIX のディレクトリとファイルの構造を見ると、それが木の根の構造のように見えると思います。また上下をひっくり返せばルートディレクトリを根にしてディレクトリの分類過程が枝のように、そしてファイルが葉のように見えるかも知れません。このようなイメージで表せる構造のことをコンピュータの世界では「木構造」「ツリー (tree) 構造」と呼んでいます。ディレクトリの木構造をディレクトリ・ツリーなどと呼んだりします。

ところで UNIX ではディレクトリはファイルの一種として扱われます。（扱われる、のです。普通のファイルとディレクトリはその意味に違いがあります。）そのため「ディレクトリファイル」などと表現されるときもあります。ディレクトリもファイルと同じ様に名前を付けてそれを他のものと区別して扱います。名前のルール（名前に利用できる文字、文字数の制限など）は普通のファイルと同じです。

[27] directory：住所録？何故この呼び名を採用したのか、私は知りません。（筆者）
[28] root directory：根、ですね。
[29] cc 環境におけるシステム管理者は計算機センターです

3.8.2 ツリー構造におけるファイル名の表記

ファイルにはファイル名が有ることは説明しました。しかしUNIXのディレクトリ・ツリーの中で名前によってそのファイルを指定するにはツリー構造を含めて表現できる方法が必要ですね。

絶対パスによるファイル名の表記

例えばユーザ名 yasuda さんのホームディレクトリの名前は `yasuda` です。このディレクトリファイルを、UNIX コンピュータ全体のツリー構造の中では「`/NF/home/g840/yasuda`」などと表現します[30]。これが正にディレクトリ・ツリーの中で絶対的な位置と名前を表す表記法です。

先頭の「`/`」はルートディレクトリを表しています。それ以降の「`/`」はディレクトリ構造の区切り、つまり図 3.7 でのディレクトリとディレクトリもしくはファイルを結ぶ「線」に相当します。「`/`」にはさまれた名前は全て途中に存在するディレクトリの名前であり、最後の名前はディレクトリ、もしくはファイルの名前です。

こうして表現することによってディレクトリ・ツリーの中で、名前によって完全にファイルが指示できるようになります。これを「絶対パスによるファイル名の表記」などと呼んでいます。絶対パスによる表記の場合、その表記はルートディレクトリを起点にして、たどって行くディレクトリの道のり (path : パス) を表現していると看做せます。

相対パスによるファイル名の表記

でも絶対パスによってしかファイルの名前が表現できないとしたらこれは非常に不便なことです。例えば yasuda さんが自分のホームディレクトリの直下にある log というファイルを old-log という名前に変えたいと思ったときに、こんな風にコマンドを書かなくてはいけません。

```
csosf01(82)% mv /NF/home/g840/yasuda/log /NF/home/g840/yasuda/old-log
```

いやこれはたまりません。これでは単に長い名前をファイルに付けているようなもので、何のためにディレクトリと言う概念を導入したのか判りません。

そこでファイルを表現するのに、ディレクトリ・ツリーの前半部分をタイプしなくても済むように覚えておいて、ツリーの残りの部分だけ表現すればいいような表記方法があります。そのためにUNIXのシェルは、常にディレクトリ・ツリーのどれか一つのディレクトリに注目しています。そのディレクトリまでのツリーの記述は省略可能となるわけです。例えば先の例の `mv` において、今注目しているディレクトリがホームディレクトリだったとすると、以下のようにコマンドを短く書くことが出来ます。

```
csosf01(82)% mv log old-log
```

この「今注目しているディレクトリ」もしくは「今省略可能であるディレクトリ」を「カレントディレクトリ[31]」もしくは「ワーキングディレクトリ」と呼んでいます。カレントディレクトリはコマンドによって変更することが出来ます。(後述)

`pwd` コマンドでカレントディレクトリを確認する事が出来ます。

```
csosf01(81)% pwd
/NF/home/g840/yasuda
csosf01(82)%
```

[30] あなたのホームディレクトリはあなたのユーザ名が使われているはずです。実際のあなたのホームディレクトリの名前が知りたければ login 直後に `pwd` コマンドで確認できます。

[31] current directory : 現在のディレクトリ

先の例の、省略された mv コマンドのファイルに関する表記では、この/NF/home/g840/yasuda が省略されていた[32]と言うわけです。

ところで、login 直後のカレントディレクトリは常にホームディレクトリです。つまり今まで 第 2 章 の 3.4 などで試してきたファイルは、この省略された表記法によって表現された、あなたのホームディレクトリ直下にあったファイルだったのですね。ls コマンドなどでその一覧が表示されていたのも、あなたのホームディレクトリ以下の内容だったというわけです。

こうして表現することによってディレクトリ・ツリーの中で、簡単な表記によってファイルが指示できるようになります。これを「相対パスによるファイル名の表記」などと呼んでいます。相対パスによる表記の場合、その表記はカレントディレクトリを起点にして、たどって行くディレクトリの道のり (path : パス) を表現していると看做せます。

パスによる表記でもう少し

言い遅れましたが「絶対的な表記」と「相対的な表記」の区別は、その表記の先頭が「/」であるか否かで判断されます。

また、パス中には以下の記号が利用できます。

記号	意味
.	カレントディレクトリを意味します 例えば./sample と書けば、カレントディレクトリにある sample というファイルを意味します。つまり単に sample と書いたのと同義です。(しかしそう書いたのでは駄目な場合もあるのです。)
..	一つ上のディレクトリを意味します。 例えば../sample と書けば、カレントディレクトリの一つ上のディレクトリにある sample というファイルを意味します。 例えば../../sample と書けば、カレントディレクトリの二つ上のディレクトリにある sample というファイルを意味します。

また、シェルからコマンドの引数としてファイルを記述するとき、パスの先頭であれば以下のような書き方も出来ます。(但し sh では駄目です。tcsh か csh で有効です。)

記号	意味
~	自分のホームディレクトリを意味します
~username	ユーザ名 username のホームディレクトリを意味します

3.8.3 ディレクトリの扱い

ディレクトリの作成

ディレクトリを作成するには mkdir[33]コマンドを利用します。書式は以下の通りです。

mkdir ディレクトリ名...

ディレクトリの消去

ディレクトリを消去するには rmdir[34]コマンドを利用します。書式は以下の通りです。

[32] 厳密には pwd の結果の最後にもう一つ/ を付けないといけませんね。
[33] make directory の略なのです。
[34] remove directory の略なのです。

`rmdir ディレクトリ名...`

ディレクトリの消去は、そのディレクトリより下にディレクトリまたはファイルが含まれていては出来ません。消去したいディレクトリ以下のファイルまたはディレクトリを `rm` または `rmdir` コマンドで予め消しておいてください。

カレントディレクトリの表示

カレントディレクトリを表示するには `pwd`[35] コマンドを利用します。書式は以下の通りです。

`pwd`

カレントディレクトリの変更（移動）

カレントディレクトリを変更するには `cd`[36] コマンドを利用します。書式は以下の通りです。

`cd [ディレクトリ名]`

login した直後はカレントディレクトリは常にホームディレクトリです。それから cd コマンドでどこのディレクトリに移動しても、単に `cd<Return>`（つまり引数であるディレクトリ名を省略）とすると、常にホームディレクトリに移動します。

3.8.4 ディレクトリを意識したコマンドの書き方

今まで紹介してきたファイルを扱うコマンドは、そのほとんどがディレクトリに対しても適用できます。これ以降に以下の図の状況を例に取って説明します。

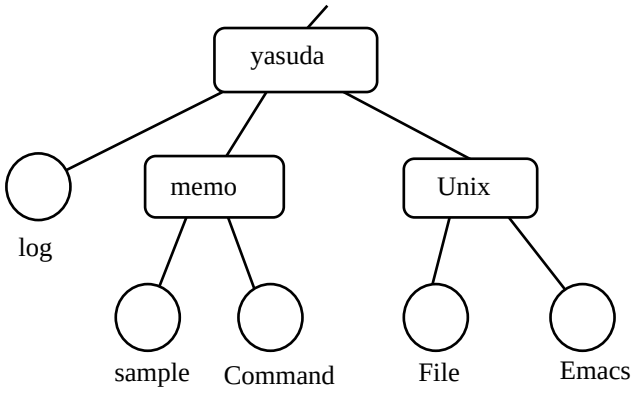

図 3.8: ディレクトリの例

まずは練習材料を作る

まず初めに練習のために図 3.8 の状況を作るのがおすすめです。以下にその手順を示します。今のカレントディレクトリはホームディレクトリではないかも知れませんから念のために最初に `cd<Return>` してホームディレクトリに移動しておきましょう。

[35] print working directory の略なのです。
[36] change directory の略なのです。

```
csosf01(84)% cd
csosf01(84)% mkdir memo Unix
csosf01(84)% ls
Apps          Mail          jsykojin.dic  memo
Library       Unix          log
csosf01(84)%
```

ファイルのコピー

cp コマンドには以下の 3 通りの書き方があります。

1. `cp [-i] file1 file2`
2. `cp [-i] file...  dir`
3. `cp -r dir1 dir2`

-i オプション[37]を与えると、コピーする際に同名のファイルが既に存在し、コピーすることによって上書きされて元の内容がなくなってしまうような場合に実行してよいかどうか問い合わせを行う。これに y と答えると実行し、それ以外の入力であれば実行しない。

1. の書き方では、file1 は file2 に単にコピーされます。

2. の書き方では、(もし複数書けば複数の) file は、dir のすぐ下に元のファイル名でコピーされます。

3. の書き方では、dir1 以下のファイルを全て含めてディレクトリごと dir2 のすぐ下に元のファイル名、ディレクトリ名でコピーされます。

例えば log ファイルを memo ディレクトリ以下に同じく log という名前でコピーする場合、以下のようなさまざまな表現が出来ます。

カレントディレクトリ	書き方	コマンド記述
yasuda	1.	`cp log memo/log`
yasuda	2.	`cp log memo`
memo	1.	`cp ../log log`
memo	1.	`cp ../log ./log`
memo	2.	`cp ../log .`

例えば sample ファイルを Unix ディレクトリ以下に同じく sample という名前でコピーする場合、以下のようなさまざまな表現が出来ます。

カレントディレクトリ	書き方	コマンド記述
yasuda	1.	`cp memo/sample Unix/log`
yasuda	2.	`cp memo/sample Unix`
memo	1.	`cp sample ../Unix/sample`
memo	2.	`cp sample ../Unix`

[37] inquiry 問い合わせ、の積りでしょうか

例えば memo ディレクトリをツリーごと Unix ディレクトリ以下にコピーする場合、以下のようになります。(今度は書き方は 3. しかありません。)

カレントディレクトリ	コマンド記述
yasuda	`cp -r memo Unix`
memo	`cp -r ../memo ../Unix` (どういうわけか `cp -r .  ../Unix` は駄目なのです)
Unix	`cp -r ../memo .`

この結果、ディレクトリ・ツリーは以下のようになります。

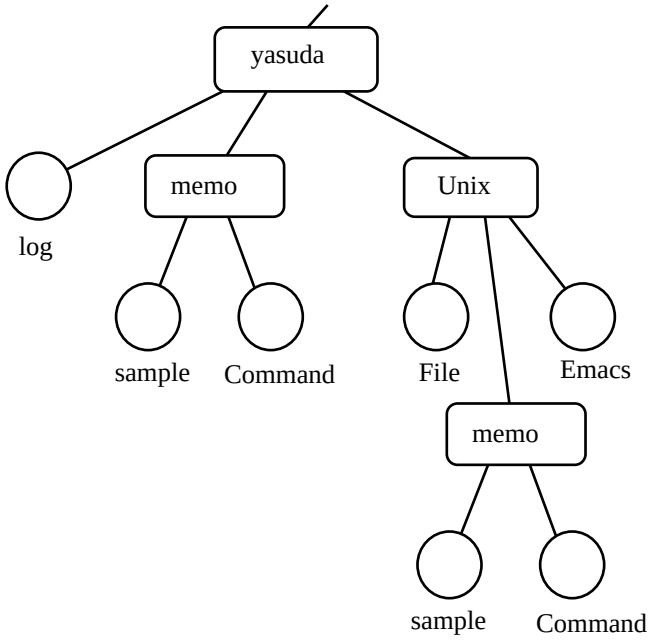

図 3.9: コピー後のディレクトリ・ツリー

ファイルの移動

mv コマンドには以下の 3 通りの書き方があります。

1. `mv [-if] file1 file2`
2. `mv [-if] file...  dir`
3. `mv -r dir1 dir2`

-i オプション[38]を与えると、移動する際に同名のファイルが既に存在し、移動することによって上書きされて元の内容がなくなってしまうような場合に実行してよいかどうか問い合わせを行う。これに y と答えると実行し、それ以外の入力であれば実行しない。

[38] inquiry 問い合わせ、の積りでしょうか

-f オプション[39]を与えると上書きされて元の内容がなくなってしまうような場合でも問い合わせせず実行する。

1. の書き方では、file1 は file2 に単に移動されます。

2. の書き方では、(もし複数書けば複数の) file は、dir のすぐ下に元のファイル名で移動されます。

3. の書き方では、dir1 以下のファイルを全て含めてディレクトリごと dir2 のすぐ下に元のファイル名、ディレクトリ名で移動されます。

mv コマンドはすぐ判るように cp コマンドと全く同じ記述方法が出来ます。働きもほとんど同じで、単に元のファイルが残る (cp) か残らない (mv) かだけです。cp コマンドと同じですので、ここでは例を挙げません。

最後の 3. の書き方で memo ディレクトリを Unix ディレクトリにディレクトリ・ツリーごと移動した場合は、図 3.9 に挙げた例の左側の元の memo ディレクトリと、それ以下の sample と Comand ファイルがなくなった状態になります。

ファイルの消去

rm コマンドは以下の書き方をします。

```
rm [-ifr] file...
```

-i オプション[40]を与えると、消去する際に場合に実行してよいかどうか問い合わせを行う。これに y と答えると実行し、それ以外の入力であれば実行しない。

-f オプション[41]を与えると問い合わせせず実行する。

-r オプション[42]を与えて file の部分がディレクトリだった場合は file 以下のファイルを全て含めてディレクトリ・ツリーごと消去する。

特に rm コマンドに-r を指定してディレクトリ・ツリーごと消去するという例は非常に有用です。なにしろディレクトリの消去である rmdir コマンドは、そのディレクトリ以下の内容が空になっていなければいけないのです。そのような場合は rm -r で一発消去ができます。

[39] force 強制、の積りでしょうか
[40] inquiry 問い合わせ、の積りでしょうか
[41] force 強制、の積りでしょうか
[42] recursive 回帰的、の積りでしょうか

3.9 EGG：Emacs での漢字の入力

いままでタイピングと言えばアルファベットのタイピングについてのみ説明してきました。しかしやはりひらがなや漢字をファイルの中に書きたいものです。ここでは Emacs を用いたひらがなや漢字などのタイピングの方法について説明します。

3.9.1 かな漢字変換

アルファベットの場合タイピングは簡単です。つまりキーボード上のキーに書いてある文字が、そのキーを押すことによって入力されるのです。ひらがなのタイピングについては何とかこの方法で済ませられるかもしれませんが、漢字についてはそういうわけには行きません。漢字は数万字（日常的に使う漢字だけでも数千字）あって、とても数千のキーを並べるわけには行かない[43]からです。

そこで「かな漢字変換」による漢字の入力の登場です。つまりまず「かな」を目標の漢字の読みとして入力することによって、それを漢字に変換しようと言うアイディアです。最近市販されているワープロは殆ど全てこの方式を採用しています。UNIX コンピュータでもこの方法で漢字をタイプします。しかも「かな」の入力はローマ字からの変換です。つまり「かな」すらキーボードには載っていない（もしくは載っていても使わない）ので、まずアルファベットのキーを利用してローマ字で「かな」を目的の「漢字」の読みとしてタイプし、それを変換するのです。

3.9.2 Wnn と EGG

Wnn[44] は 京都大学数理解析研究所、オムロン株式会社、株式会社アステックの 3 者によって開発されたかな漢字変換システムです。cc 環境ではこの Wnn を標準的な漢字変換システムとして採用しています。

EGG[45]は電子技術総合研究所の戸村哲氏が中心となって開発した、Emacs 上で Wnn を利用するためのシステムです。cc 環境ではこの EGG を Emacs 上での標準的なかな漢字変換システムとして採用しています。

EGG のモード切り替え

さて、Emacs 上で漢字をタイプするためには、まず EGG をローマ字かなモードにします。Emacs が起動されている状態で、C-\ です。（キーボードによっては C-\ の代わりに、C-¥ かもしれません。）モードラインの左端に注目してください。以下のようになるのが判ると思います。

```
[ aあ]-----NEmacs: *scratch*            (-EE-:Lisp Interaction)--All----------
```

この状態から元に戻るためには、再び C-\ です。モードラインが元に戻りましたね。C-\ でモードラインの左端がくるくる変わるのを確認してください。

```
[----]-----NEmacs: *scratch*            (-EE-:Lisp Interaction)--All----------
```

モードラインの左端が [----] となっている状態を EGG の「透過モード」と呼んでいます。（透過モードと言うのは、つまり今までどうりアルファベットのタイピングがそのまま行える状態です。）[aあ] となっている状態を「ローマ字かなモード」と呼んでいます。ローマ字かなモードにしておくと、タイプしたアルファベットはまず EGG に受けとられ、そこでローマ字として解釈されてひらがなが表示されます。それから漢字変換に関するキーを操作して目的の漢字かなまじり文へと変換するのです。

[43] 昔の漢字タイプライタは正にその通り数千のキーを並べていましたけどね。コンピュータではその方法は採用していません。
[44]「うんぬ」と読みます。名前の由来は「私の名前は中野です」の略から来ているそうです。
[45]「えっぐ」と読みます。名前の由来は「たくさんまたせてごめんなさい」の略「たまご」から来ているそうです。

ためしに変換

ローマ字かなモードにして、アルファベットで「watasinonamaehanakanodesu」とタイプしてください。打ち込んだアルファベットが縦棒にはさまれながら次々とひらがなに変換されて行くのが判るでしょう。

図 3.10: EGG のフェンスモード

この縦棒を EGG ではフェンスと呼び、変換途上のかな文字がフェンスにはさまれている状態をフェンスモードと呼んでいます。

フェンスの中の文字を編集するには普通の Emacs の編集のときと同じくカーソルキー（←や→）と<Delete>キーです。カーソルキーが効かない場合は C-b でカーソル左、C-f でカーソル右と同じ動きをします。

ローマ字が全てかなに変換されたら<Space>キーを押すことによって今度はかなを漢字に変換する作業が始まります。

注意

あなたがそのコンピュータで初めてかな漢字変換を行うときは、以下のような問い合わせが行われます。

図 3.11: ユーザ辞書作成の問い合わせ

これらの質問には全て yes <Return> と答えてください[46]。

再び注意

あなたがその Emacs で初めてかな漢字変換を行うときは、エコーラインに「ホスト local の WNN を起動しました」などというメッセージが表示されて、かな漢字変換が開始されるのに若干時間が掛かるかも知れません。

かな漢字変換が始まると、モードラインの左端が以下のようになります。

```
[漢字]--**-NEmacs: *scratch*              (-EE-:Lisp Interaction)--All----------
```

フェンスの中は以下のようになっているでしょう。このモードラインの左端が [漢字] となっている状態を「漢字変換モード」と呼んでいます。

[46] 結構何回も yes と答えなければなりません。今試してみたら合計 10 回も必要でした！

図 3.12: EGG の漢字変換モード

ここで<Return>とすれば現在表示されている漢字かなまじり文の候補「私の 名前は 中野です」が採用されて、Emacs にタイプされます。フェンスがなくなって、漢字が Emacs の普通の操作で編集できる状態になりましたね。

この作業を「確定」と呼んでいます。これでまず一つ漢字の文章がタイプされました。一安心です。

再変換

一度変換するだけで自分の目的の漢字に変換してくれれば良いのですが、先ほどのようにうまく行くことばかりでありません。そこで、EGG が最初に挙げてくれた漢字の候補を再変換しなければなりません。一度でうまく行かない例として、今度は「nanasinokakasihaeraihito」とタイプして <Return>で変換してください。恐らく以下のようになるでしょう。

|名 なし 野 化 貸しは 偉い 人|

惜しいですね。ここでは最終的に「名なしのカカシは偉い人」に変換し直す例をあげます。

かな漢字変換では変換は文節単位に行なわれます。はじめはローマ字でタイプされた一連のひらがなの列から EGG が自分で文節の区切りを見つけて変換します。上記の例ではその結果「名 なし」「野」「化」「貸しは」「偉い 人」と文節の区切りを付けました。EGG の漢字変換モードでは文節の区切りを一つの空白で表します。「名なし」と「野」などの間にそれぞれ一つの空白があるのが判りますね。

ここでやらなければならないのは「野」を「の」に変換し直して、「化」の文節を伸ばして「かかしは」までにし、それから「カカシは」に変換し直すという二つの作業です。それぞれについて見て行きます。

次の候補を上げる

今、カーソルは「名なし」の上にあると思います。再変換したいのは、その次の文節である「野」ですから、まずカーソルを次の文節に移動します。

フェンスの中の文節を移動するには普通の Emacs の編集のときと同じくカーソルキー（←や→）です。カーソルキーが効かない場合は C-b でカーソル左、C-f でカーソル右と同じ動きをします。→もしくは C-f で「野」の上にカーソルがやってきたら、そこで<Space>を押すことによって次の候補が表示されます。何度か<Space>を押しているうちにひらがなの「の」が現れると思います。あまり急いで<Space>を押しすぎると肝心の「の」が現れても勢いでもう一度<Space>を押してしまう時もあるでしょう。そういう時は C-p で一つ前の候補を再表示する事が出来ます。

文節の区切りを変える

「の」がうまく行ったら今度は「化」の文節に C-f でカーソルを移動して、今度は文節の区切りそのものを訂正しなくてはなりません。文節の区切りを伸ばすには C-o です。「欠かしは」となるまで何度か押して文節の区切りを伸ばします。再び何度か C-o を押しているうちに、あまり急いで C-o を押しすぎると肝心の「欠かしは」になっても勢いでもう一度 C-o を押してしまう時もあるでしょう。そういう時は C-i で文節を短くする事が出来ます。

文節が目標の「欠かしは」になったら（「欠かしは」が自分の目的の漢字では無いので）「カカシは」が現れるまで繰り返して<Space>を押します。目的の漢字が現れたら<Return>で確定、です。

3.9.3 まとめ

以下にキー操作のまとめを示しておきます。

ローマ字かなモードでのキー操作

<Delete>	タイプミスの修正、一文字削除
←または C-b	カーソル一文字左
→または C-f	カーソル一文字右
<Space>	かな漢字変換の開始

漢字変換モードでのキー操作

←または C-b	カーソル一文節左
→または C-f	カーソル一文節右
C-o	文節を伸ばす
C-i	文節を縮める
<Space>	次候補の表示
C-p	前候補の表示
M-s	候補の一覧をエコーラインに表示する。数字による選択かカーソルキー（もしくは C-b ,C-f ,C-p ,C-n ）による選択を行い<Return>で確定。 候補一覧表時から抜けて元に戻るには C-g 。

3.9.4 ローマ字入力のヒント

記号を入力したいのです

大抵の記号に関しては「○」は「まる」などのように、その記号の読みが登録されていますから、それで変換出来ます。

「たんい」（もしくは「んお？」など）のように「ん」の次に母音が来るのです

「ん」を単独でタイプするために N (大文字) と n' が用意されていますので、それを利用してください。「taNi」もしくは「tan'i」とタイプすれば「たんい」とローマ字変換されます。

「あっ」（もしくは「んぁ？」など）のような小さい「っ」「ぁ」を入力したいのです

小さい「ぁぃぅぇぉ」などは、x を前に付けてタイプします。「xa」とタイプすれば「ぁ」とローマ字変換されます。「xtu」で「っ」、「xyu」で「ゅ」です。

第 4 章

ネットワークの世界へようこそ

京都産業大学内では多くのコンピュータが導入され、教員や学生に提供されています。これらを使って行く上で、利用者同士の情報交換は非常に有用です。今や殆ど全てのコンピュータは電線で結ばれ、ネットワークと呼ばれる構造によってコンピュータ同士の間で通信が可能です。つまりコンピュータの利用者の間でお互いのメッセージを交換することが出来るのです。

典型的なメッセージ交換の形として「電子メイル」「ニュース」などの方法が存在します。これらの機能を総称してネットワークサービスなどと呼んだりします。ここでは、それぞれのネットワークサービスがどのようなものか説明します。

4.1 ネットワークサービス紹介

ここでは「電子メイル」と「ニュース」サービスがどのようなものかを説明します。すでに知識と経験のある人は 4.3 まで読み飛ばして下さい。但し 4.2.2 については読んでおく方が良いかも知れません。

4.1.1 電子メイルって何？

電子メイルは通常の郵便システムと似た機能を持ちます。大きな相違は紙や物を運ぶのではなく、文字をデータ化して相手に届けるという点です。つまり通常の葉書や封書による郵便は、実際には紙としての葉書や物としての封書を届けるのが第一の目的では無く、そこに書かれた内容、つまり文字を相手に届けることが本質です。しかし実際には紙や物を届けることによって実現しているというのが郵政省の郵便システムと言えます。

それに対して電子メイルでは、一般的に文書（整形された文字の集合）の内容をコンピュータに入力する事によってデータ化し、またコンピュータの機能を使って相手に送り付けます。受け取った側でも、データ化された文書を、受け取った側に読めるようなかたちでコンピュータによって表示します。そうして送り手が書いた内容が受け手に伝わる、と言う仕掛けです。

仕掛けはともあれ、一般の郵便も電子メイルも誰かが書いた文章をある特定の相手に届けると言う本質は変わりませんね。

後の 4.1.5 でもっと具体的な利点などを紹介します。

4.1.2 ニュースって何？

電子メイルが個人と個人の間のメッセージ交換であったのに対して、ニュースは個人が大勢に対してメッセージをやり取りできるシステムです。

ニュースではあたかも掲示板に個人がメッセージを書くようなもので、大勢の人が（もしも興味があれば）そのメッセージを読み、そして自分の意見を再びそれぞれ大勢の人に見えるように書き込みます。こうすることによって、広く一般に向けて公開の議論が可能になると言うわけです。

一般の掲示板は本当の木の板にチョークか何かで書くのでしょうが、ネットワーク上のニュースシステムでは、コンピュータを介する事によってメッセージをデータ化して行います。木の板の文字かデータかはともあれ、個人が書いたメッセージを多くの人が覗いて公開の返事を書くという本質はかわりませんね。

後の 4.1.6 でもっと具体的な利点などを紹介します。

4.1.3 京都産業大学のネットワーク

京都産業大学のコンピュータは、その多くが学内のネットワークに接続されています。計算機センターが管理しているコンピュータの多くも学内ネットワークに接続され、また上記のネットワークサービスが受けられます。

京都産業大学のネットワークは全世界を覆うネットワークに接続されています。この巨大なネットワークのことを Internet と呼んでいます[1]。京都産業大学のコンピュータの多くは、つまり Internet につながれていると言うことです。そして京都産業大学で行われているネットワークサービスの多くはこの Internet で行われているものと同じものです。それ故に京都産業大学の電子メイルサービス、ニュースサービスは、世界中の電子メイルサービス、ニュースサービスと通じ合えると言う訳です。

今やあなたも京都産業大学内のコンピュータを介して世界中の人と電子メイルを交換することが出来ます。また、あなたは学内のコンピュータを介して世界中の人とニュースによって意見を交換することが出来ます。

4.1.4 Internet とは？

Internetは世界中のネットワークを相互接続したもので、学術研究のための実験ネットワークです。Internetには主体となる運営組織は具体的には存在しません。また中心となるコンピュータもどこにも存在しません。世界中にあるネットワークを接続して、お互いの好意で協調して運用しています。商用のいわゆるパソコンネットとはこれらの点で異なります。このネットワークの特徴として、主に以下のような特徴が挙げられます。

- 各組織（ドメインとも言います）の組織内ネットワーク同士を結合することにより、メイルやニュース等のサービスを提供しています。サービスについては次節でもう少し説明します。
- 各 Internet 参加組織間（および組織内）のリンクの管理の多くは各組織のボランティアによって行なわれています。（全てでは無い）
- 各組織のネットワーク（特にメイルサービス）の管理者をポストマスタといいます。
- 国内での Internet の利用は非営利目的な利用に限られます。営利目的に用いることは一切禁止されています。たとえば、Internet での商売、商品の宣伝、リクルート活動など。ただし海外では一概にそうとは言えません。それはその国の法律が決定するでしょう。
- 国内では Internet に参加している組織は、大学などの教育機関、企業、国立の研究機関などで、'94 年 3 月現在約 1400 組織です。全世界だと、数万組織は接続しているでしょうか？

[1] なんだか話が SF チックになってきましたね。

これ以上の情報を得るたには、コンピュータ系の雑誌やInternet newsの記事等から自分で勉強するように努力してみて下さい。図書館にも置いてある各種のコンピュータ雑誌も推薦出来ます。Internet newsについては4.1.6で説明します。

4.1.5 Internet mailサービスってどんなもの？

Internetで行われている電子メイルサービスを利用すると、世界中に散らばる数万組織の人達に対して手紙を出すことが出来ます。しかも大学などの研究機関が主にネットワークに参加していますから、我々大学関係者などが連絡をとりたい相手もこの電子メイルサービスを利用している可能性は結構高いのです。

この電子メイルサービスは郵政省が行なっているメイルサービス（郵便）に比べると以下の点で優れています。

- 手紙が速く相手の手元に到着する
  たとえ相手がアメリカでもオーストラリアでもイギリスでも、大抵の場合（控え目に見積っても）数分間以内に相手の手元に手紙が届きます。実際アメリカなどへは郵便では10日くらいかかったりします。

- 基本的に24時間営業、年中無休である
  夜にアメリカ宛に出したメイルは恐らく向こう時間の朝に到着するでしょう。日曜日は配達してくれなかったり、年賀状シーズンに停滞したりすることはありません。

- 郵便ポストまで歩いて行かなくて済む
  京都産業大学のネットワークに接続できる端末がありさえすれば、そこから全ての操作が出来ます。自宅から作業をする事も出来ます。

- 紙がたまらないで済む
  頻繁に手紙のやり取りをしていると、結構大きさのまちまちな保存しにくい紙がたまるものです。しかし電子メイルでは手紙はコンピュータの中に残り場所を取りません。そして、消さない限りいつでも取り出して読み返せます。更に、計算機の中に入っている限りキーワードで検索可能です。紙をめくって斜め読みする必要はありません。

逆に、以下の点では劣っています。

- 書留郵便がない
  メイル配送は先に述べた通りまだボランティアベースで行なわれている部分があります。出したメイルを必ず（何があっても）相手に届けるようなサービスは行なわれていません。

- 自分がネットワークに加入していないといけない
  しかもコンピュータを使わなければなりません。しかし最近ではコンピュータもずいぶん使いやすくなりました。

- 相手がネットワークに加入していないといけない
  しかもコンピュータを使っていてくれなければなりません。しかし最近では結構大学関係者はこう言うものに参加しています。

電子メイルは世の中の殆どの便利なものと同じ様に、決して万能ではありません。しかし、ある局面では他の何ものにも代えられない位役に立つことがあるのです。

4.1.6 Internet news サービスってどんなもの？

Internet news は最近普及しつつある NIFTY-serve など、つまり商用のいわゆるパソコンネットの電子掲示板 (BBS) システムに似ています。つまりたくさんの人が掲示板に自分の言いたいことを書き込むと言うものです。この掲示板はまた多くの人に見られていますから、読んでいる人が書き込まれたことに反論したり、意見を添えたりします。

具体的には「私は今度行われる教育改革には反対だ。」「いや、私はこの点で賛成する。」「私も賛成だ。」と言った議論や、「先日若狭に釣りに行ってきました。今は鯛が好調です。」「私も行ってきました。少し沖合いに出るとハマチが来ます。餌は生き餌がいいようです。」「私も行きたいのですが、舟を紹介してください。」と言った趣味の話などが行われています。

また、何かについて困っているときに質問を書き込めば、誰か親切な人が解決法を教えてくれるかも知れません。そんな都合の良い話は無いって？いいえ、これは冗談ではありません。ネットワークの向こうには何千人もの非常に親切でお節介な人が暮らしているのです！[2]

要は多くの人が見ているところに書き込むわけですから、勢い情報交換の場となるのです。しかもこれは世界中の人が参加しています。つまり我々は海外の情報も居ながらにして読むことが出来るのです[3]。非常にたくさんの人が参加して、いつも多くの書き込みがありますから興味のある話題ごとにグループ化されています。

また、京都産業大学の内部向けに幾つかのニュースグループが用意されています。「sandai.」が頭に付くニュースグループです。外部の情報には興味がない人も（せめてここだけでも）是非覗かれることをお勧めします。有用な情報の広報などはここで良く行われます。

Internet news は生まれが USENET と呼ばれる大学間で始まった実験ネットワークです。その歴史的な経緯でニュースシステム上で用いられる用語が少し一般のパソコンネットの電子掲示板システムとは違っています。

共通の興味によってくくられる話題は決められた場所に書き込むことになっていますが、これをパソコンネットでは SIG もしくはボード、フォーラムなどと呼んでいます。Internet news ではこれをニュースグループと呼んでいます。

ニュースシステムにユーザが書き込んだメッセージを Internet news では記事（アーティクル）と呼ぶ場合があります。

記事を書き込むことを Internet news では投稿（ポスト）と呼んでいます。

4.1.7 ネットワークでの暮らし方

ネットワークサービスと共にコンピュータを利用するのは非常に快適なものです。ですが、その環境で快適に暮らすにはある程度ルールを心得ておくことが重要です。ネットワークサービスは機械によって提供されていますが、相手をしているのは機械ではなく、人間であることをとかく忘れがちです。

第 1 章の「はじめに」などでも述べているように、京都産業大学のコンピュータ環境には一般社会と同じ様に規則、慣習、道徳があり、そして法律も適用されます。京都産業大学のネットワークは Internet の一部でもあります。京都産業大学のコンピュータ利用環境よりもっと大きな利用環境である Internet にも勿論様々な規則、慣習があります。お互いに協調して暮らして行けるように心に留めておきましょう。

[2] 嘘だと思うならコンピュータ関係のニュースグループを少し覗いてみてください。

[3] もちろん海外のニュースは英語が殆どです。でも日本国内のニュースは日本語 (漢字) で流れていますから、英語を読みたくない人も御安心下さい。「fj.」が頭に付くニュースグループが漢字のものです。

4.2 電子メイル準備体操

4.2.1 Internet mail アドレスについて

Internet でサービスされているメイルサービス[4]でも一般的な郵便システムと同じ様に、宛先の住所を明記する必要があります。Internet は世界中でサービスされているので、世界で一意に決まる[5]住所がメイルサービスを受ける人それぞれに必要です。すなわちそれが住所であり、Internet mail サービスがアメリカ生まれであることからアドレスと呼びます。メイルサービスのためのアドレスですから、メイルアドレスと言うとその意味合いがもっとはっきりするでしょう。

現在のところ、メイルサービスがコンピュータを介して行われているものであるため、メイルアドレスはそれぞれのコンピュータの利用者に割り当てられます。逆に言えばメイルサービスを利用するためには、メイルサービスが利用可能なコンピュータの利用者とならなくてはいけません。

京都産業大学の幾つかのコンピュータは、Internet mail サービスが利用可能です。それらのコンピュータの利用者は Internet mail サービスを利用することが出来るでしょう。つまり Internet mail サービスを受けている世界中に数万台（もっとかな？）存在するコンピュータのユーザそれぞれとメイルを交換することが出来るという事になります。

京都産業大学のメイルサービスが利用出来るマシンにおけるメイルアドレスは一般的には以下のような書式となっています。

```
foo@bar.kyoto-su.ac.jp
```

アドレスの表記そのものは西欧式になっており、右側に大きな区分、左側に小さな区分が書かれています。ピリオド (.) で区切られており、右側からそれぞれの区分の意味を以下に示します。

`jp`	Internet mail アドレス最大の区分で、国を示しています。
`ac`	jp 以下に存在する中区分であり、教学関係であることを示す。他に co が一般企業、or がその他の組織などとして定義されています。
`kyoto-su`	京都産業大学を示しています。
`bar`	ここには何が来るか一概には言えません。京都産業大学内で、既に決められている宛先に配送されることを示しています。大学のネットワーク管理者が決定した区分のようなものだと考えてください。
`@`	これより左はユーザ名であることを示しています。
`foo`	foo（仮称）というユーザ名宛てに配送されることを示しています。

`kyoto-su.ac.jp` は、京都産業大学のネットワークを示すアドレスであり、世界中を覆う Intenret で一意な名前です。ネットワーク環境の中で、このように階層付けされて一意に確保されているような名前をドメインなどと呼んだりします。`kyoto-su.ac.jp` より左の `foo@bar` は京都産業大学内で一意に保たれるように管理されています。あなたが電子メイルを利用するときはあなたが利用者として登録されているコンピュータの管理者に、自分のメイルアドレスがどのようなものかを確認することが重要です。

4.2.2 計算機センター管理のコンピュータのメイルアドレス

ここでは計算機センターが管理しているコンピュータのメイルアドレスに限定して説明します。ここで説明するルールは普遍的なものでは無いので、他のマシンのユーザのメイルアドレスを類推する役には立

[4] これ以降単にメイルと言ったら電子メイルを差します
[5] 一意に決まる、とは「間違いなくたった一つのものに特定出来る」と言う意味です。

たないでしょう。他のマシンのメイルアドレスについては、それらのマシンの管理者に問い合わせる必要がある事に注意してください。

計算機センターが管理し、産業大学の教員、学生に提供しているコンピュータのうち、メイルサービスが利用可能なのは以下のマシンです。ホスト名とはネットワーク上の各コンピュータの名前です。以降各マシンはホスト名で表現します。

機種名	ホスト名	
SPARCcenter2000	cc2000	計算機センター 1 階に設置の Sun 社製コンピュータ。
DEC-3300	csosf01〜41	2 号館 4 階 21 情報処理教室に 41 台設置の DEC 社製コンピュータ。
NeXTstation	ccns001〜015	計算機科学研究所 3 階 C3 情報処理教室に 15 台設置の NeXT 社製コンピュータ。
DEC-3500	ksuvx1	2 号館 1 階に設置の DEC 社製コンピュータ。

以上のマシンでメイルを利用する場合、メイルアドレスはそれぞれ以下の通りとなります。

ホスト名	メイルアドレス
cc2000 および csosf01〜41	`username@cc.kyoto-su.ac.jp`
ccns001〜ccns015	`username@ccnext.kyoto-su.ac.jp`
ksuvx1	`username@ksuvx1.kyoto-su.ac.jp`

上記のうち@より左の `username` には各ユーザのユーザ名を書きます。例えば神山太郎さんは taro というユーザ名で cc 環境にユーザ登録されているとすると、 cc2000 や csosf シリーズのマシンでメイルを扱うためのメイルアドレスは

```
taro@cc.kyoto-su.ac.jp
```

となります。また、太郎さんは ccns シリーズも使っています。ユーザ登録情報については cc2000, csosf シリーズと ccns シリーズのマシン群は全て共通なのですが、メイルアドレスについては ccns シリーズだけ別個になります。太郎さんが ccns シリーズのマシンでメイルを扱うためのメイルアドレスは

```
taro@ccnext.kyoto-su.ac.jp
```

です。

ところで太郎さんは上記のように二種類のメイルアドレスを持っています。複数のマシンにユーザ登録しているから複数のメイルアドレスを持つようになるわけで、これは自然なことです。例えば太郎さんが二箇所に部屋を借りているような状態だと想像すれば良いでしょう。しかし太郎さんにメイルを送る場合には、上記のうちのどれに送るのが妥当かは送る前に太郎さんに電話なり手紙なりで確認するべきです。つまり太郎さんは二箇所に部屋を借りていて、二つのメイルボックスを持っているが、そのどちらを毎日チェックしているかは誰にも想像出来ないからです。これについては次の「相手のメイルアドレス」にもっと詳しく書きます。

ところでよくよく考えてみると太郎さんは cc2000, csosf01〜41 マシン群及び ccns001〜ccns015 マシン群と合計すると数十台に及ぶ非常に多くのマシンに登録されていることになりますね。しかしそんなにたくさんメイルボックスがあっては面倒なので、cc2000 と csosf01〜41 マシン群はメイルボックスを共有しており、`username@cc.kyoto-su.ac.jp` 宛てに送られたメイルは、この共用のメイルボックスに配送されます。故に cc2000 及び csosf01〜41 のいずれのコンピュータを用いてメイルボックスをチェックしても、配

送されてきたメイルを読むことが出来ます。どのコンピュータからメイルボックスをチェックしたとしても、結果的には共通の、たった一つのメイルボックスを見ている事になるからです。

また、ccns001〜ccns015 マシン群もメイルボックスを共有しており、`username@ccnext.kyoto-su.ac.jp` 宛てに送られたメイルは、この共用メイルボックスに配送されます。つまり計算機科学研究所 3 階に並んでいる 15 台の NeXT のどれに座っても `username@ccnext.kyoto-su.ac.jp` 宛てに送られてきたメイルを見ることが出来ます。

共用メイルボックスを cc2000, csosf01〜41 マシン群と、ccns001〜ccns015 マシン群とに分けている理由は ccns マシン群が扱う NeXT メイル[6]を cc2000, csosf マシン群が扱えないからです。

4.2.3 相手のメイルアドレス

いざメイルを誰かに送ろうとした場合、相手のメイルアドレスが必要になるでしょう。相手のアドレスを調べるには、その相手に聞くしか方法がありません。例えば神山太郎さんが上記の計算機センター管理のコンピュータ、cc2000 や csosf01〜41 に hanako でユーザ登録されている神山花子さんにメイルを出そうとした場合、`hanako@cc.kyoto-su.ac.jp` 宛てにいきなり出すのは好ましくないと言えます。コンピュータシステムは正直に cc2000 及び csosf01〜41 コンピュータの為の hanako ユーザの共用メイルボックスに太郎さんのメイルを配送するでしょうが、花子さんが本当に cc2000 などのマシンでメイルボックスをチェックしてくれるとは限らないからです。ひょっとしたら花子さんは電子メイルなど全く使っていないかもしれません。ひょっとしたら、花子さんは他のコンピュータにも何らかの名前で登録されておりメイルはもっぱらそこで利用しているかもしれませんね。

では太郎さんが花子さんにメイルを送る場合はどうすれば良いのでしょうか？結論は「君に今後メイルを送りたいのだけれど、いったいどこに送れば良いの？[7]」と最初の一回目に（勿論メイル以外の方法で）聞くことです。馬鹿馬鹿しいようですがこれは非常に重要なことです。

4.2.4 自分のメイルアドレス

自分のメイルアドレスが何であるかは自分がユーザ登録されているコンピュータの管理者に確認するのがいいでしょう。但し計算機センターが管理しているコンピュータについてのメイルアドレスに関しては先に述べた通りです。

ある人にメイルを送って欲しいと思った場合は、どこ宛てに送って欲しいか、すなわち自分が日常的にチェックしているメイルアドレスを相手にはっきり通知することが大切です。これは先に書いた相手のメイルアドレスをはっきり聞く、という事の裏返しです。同じく非常に重要なことです。

4.2.5 さあ、本番！

準備体操はこのくらいにしておきましょう。実際にメイルを読んだり書いたりするのは現在ではコンピュータを操作すると言う事にほかなりません。即ち普遍的に「こうすればメイルを読み、書く事が出来る」という方法はなく、そのコンピュータ独自の操作方法を修得する事になります。

次に計算機センターが管理している cc 環境の UNIX コンピュータを利用してメイルを読み書きする方法について説明します。

---

[6] NeXT メイルとは絵や音や様々なデータを簡単な操作でメイルに含ませて送ることが出来るシステムで、現在この機能は NeXT コンピュータでしか利用出来ません。cc 環境では ccns001〜ccns015 までのマシンで NeXT メイルが扱えるという事です。この使い方などについては NeXT のマニュアルを参照してください。

[7] 当然関西弁でも可

4.3 MHE：Emacs による電子メイルの読み書き

MHE[8] は Emacs を利用して電子メイルを読み書きする機能を提供します。
Emacs と共に働きますから、Emacs の操作方法についてある程度理解していることを前提に説明します。

4.3.1 はじめに

ここでは以下の流れに従ってメイルを扱う方法を説明します。

- メイルを読む
- メイルを書く
- 来たメイルの返事を書く
- メイルの整理

この流れの通り、まずメイルを読む方法を説明したいところなのですが、ちょっと問題があります。つまり恐らくあなたはまだ誰からもメイルを送ってもらっていないので、メイルを読む練習をするにも、読むべきメイルが届いていないだろうと言うことです。
そこでまず練習のために、自分自身にめがけて実験メイルを送るコマンドを紹介します。`mailself` コマンドです。

```
csosf01(81)% mailself
csosf01(81)%
```

`mailself` コマンドは実行に数秒以上掛かります。うまく実行出来た場合は、上記のように何もメッセージを表示せずに終了し、プロンプトが返ってくるでしょう。これで一通、新しいメイルがあなた宛に届いているはずです。次の節からは、そのメイルを読むことで練習して行きましょう。
MHE の全ての操作は Emacs 上で行います。さあ、`emacs` コマンドで Emacs を起動して下さい。

[8] 名前の由来は「Emacs front end to the MH mail system」から来ています。mh についてはここでは説明しません。`man mh` でマニュアルが用意されていますが、特に理解する必要はありません。

4.3.2 メイルを読む

メイルを読むためには、Emacs が起動されている状態で M-**x mh-rmail <Return>**とします。

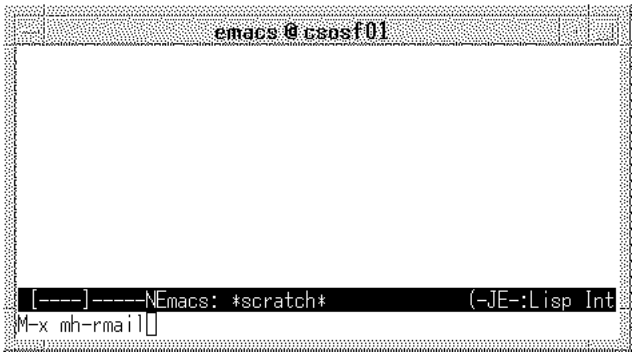

図 4.1: mh-rmail の起動

新規にメイルが届いている場合はこれで以下のような画面表示となるでしょう。

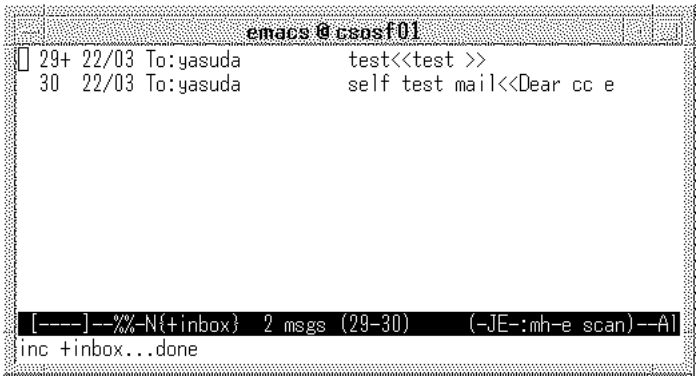

図 4.2: メイルの一覧表示

到着したメイルが一覧として一通一行の形で表示されているでしょう。カーソルは一覧表示の左側にあるはずです。ここでカーソルを上下させて、自分が読みたいメイルの行の左にカーソルを移動させます。カーソルの上下は Emacs 上でのファイルの編集の際のカーソルの上下と同じです。つまり上（一つ前の行）に移動したいときは上矢印（↑）もしくは C-**p** キー、下に移動したいときは下矢印（↓）もしくは C-**n** キーです。M-< や M-> で一番先頭や末尾の行への移動が出来ます。

自分が読みたいメイルの行の左にカーソルを移動させて「．（ピリオド）」を押せば、そのメイルの内容が Emacs のウィンドウを二分割して下半分に表示されると思います。

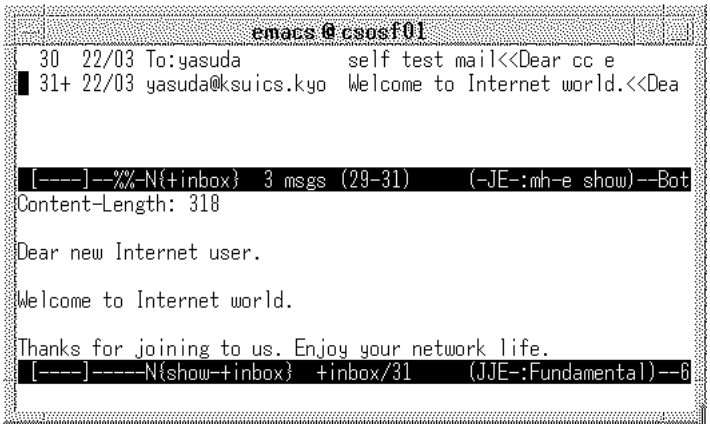

図 4.3: メイルの内容の表示

一つのメイルの行が長すぎて Emacs のウィンドウにおさまり切らないときは <Space>キーで一画面分スクロールさせる事が出来ます。スクロールして過ぎてしまった部分の内容を巻き戻してみたい場合は <Delete>キーです。

用語説明

まず一覧に現れる情報について少し説明しておきます。以下に一覧表示の例を挙げます。

```
19+ 03/17 Yutaka Yasuda      Welcome to Internet world. <<Dear friends,
```

左から順番に項目別に説明して行きます。

19	順番に振られる番号です。番号が新しければ新しいほど昔の、古いメイルです。
+	現在このメイルに注目しているよと言う印です。
03/17	日付です。
Yutaka Yasuda	メイルの送り元、つまり誰から来たかを示しています。 mailself コマンドで送り付けたテストメイルであれば恐らくここは To:に続いてあなたのユーザ名が来るでしょう。
Welcome to Internet world.	メイルの表題です。
<<	ここから内容の一部だよと言う印です。
Dear friends,	恐らくまだ続きがあるはずですが、これがそのメイルの本文の先頭部分です。表題だけでメイルの内容の想像が付かないときなどは重宝します。

上記の例での Welcome to Internet world.、つまり表題はメイルの世界では Subject と呼ばれ、本文とは少し区別して扱われます。つまりメイルは主に「宛名」「送り元」「Subject 」「本文」の 4 つの部分からなっていると言うわけです。

メイルの一覧表示とはつまりこの主たる 4 つの部分のうち、宛名を除いた三つを並べているという事です。この一覧が並んでいる状態を、フォルダモードと呼んでいます。

さて、今度はメイルの内容を読んでいる時に表示される部分の初めの方に、必ず付いてくる `To:` などに導かれた数行に注目してください。これらはメイルのシステムが付けたもので、そのメイルの補助的な情報が記録されています。一般の郵便で言うと表書きや消印の情報に相当します。この部分をメイルのヘッダと呼んでいます。以下にメイルヘッダの例を挙げます。

```
To: yasuda
Subject: Re: NOMIKAI again
Date: Mon, 21 Mar 1994 12:53:31 +0900
Cc: tanaka, ryo, omatsu
From: Tanaka Terunori <tanaka>
```

以下に順番にそれぞれの行ごとに説明して行きます。

`To:`	誰宛に送られた手紙かを示しています。ここでは `yasuda` さん宛ですね。
`Subject:`	表題です。これはこのメイルを書いた人が付けたものです。既に説明しましたね。
`Date:`	このメイルが書かれた日付です。
`Cc:`	Carbon Copy を意味しています。Carbon Copy とは複製を意味していて、このメイルの複製を誰宛に同時に送ったかを示しています。ここでは `tanaka, ryo, omatsu` の 3 人に送っています。
`From:`	誰から送られてきたかを示しています。ここでは `tanaka` さんです。`Tanaka Terunori` は `tanaka` というユーザ名の人のフルネームを示しています。

次のメイルを読む

図 4.3のようにメイルの内容が表示されている状態で、(もしあったとして) 次 (もしくは一つ前) のメイルを読むには幾つかやり方があります。

- `q` キーを押して一覧だけが表示されている図 4.2 の状態に戻り、そこでカーソルを一つ下 (もしくは一つ上) に移動して、そこでもう一度「`.` (ピリオド)」で内容を読む。
- 図 4.3の状態のまま、カーソルを一つ下 (もしくは一つ上) に移動して、そこでもう一度「`.` (ピリオド)」で内容を読む。
- 図 4.3の状態のまま、`n` キーを押して次のメイル (もしくは `p` キーを押して一つ前) の内容をすぐに表示させる。

MHE を終る、再起動する

メイルを読み終って、普通の Emacs の操作に戻りたいと思ったときは `q` キーを押します。これで MHE を起動する前、つまり図 4.2 の状態に戻ります。Emacs を終りたい場合はいつも通りに C-`x` C-`c` です。

再びメイルを読みたいと思ったときは単にもう一度 M-`x` `mh-rmail<Return>`とするだけです。但し、先ほど全てメイルを読んでしまっていたとすると、`mh-rmail` は起動されたらまだ一度も読んでいないメイルの一覧を表示しようとしますから「読んでいないメイルは一つも無いよ」と言ってメイルを一行も表示しないでしょう。このような場合、前に読んでしまったメイルをもう一度読み返したい場合は M-`r` です。

4.3.3 メイルを書く

さて、メイルを読むことが出来るようになったら今度はメイルを書いて送ってみましょう。一番良いのは誰か知人に相手になって貰うことですが、それが出来ないようならまず自分自身宛に送って、その結果をチェックするのがよいでしょう。

新たにメイルを書いて発信するには Emacs が起動されている状態で、M-x `mh-smail<Return>`とするか、`mh-rmail` を実行してメイルの一覧が表示されている図 4.2 の状態で `s` もしくは `m` とします。

するとまずエコーラインに `To:`と表示され、宛先を問うて来ます。ここでメイルを送る相手のメイルアドレス（4.2.3で説明したことに注意して下さい）をタイプします。`<Return>`すると今度は `Cc:`と表示され、Carbon Copy が必要かどうかを聞いてきます。Carbon Copy とは複写のことで、そのメイルの複写を控えとして送ることを差します。もしもあなたがそのメイルの Carbon Copy をどこかに（例えば自分自身にでも）送りたいのであれば `Cc:` の問いかけに対してメイルアドレスをタイプすることで答えます。アドレスは幾つでも並べて書けますので、複数の相手に Carbon Copy を送ることも可能です。並べて書くときはアドレスとアドレスの間に「,（カンマ）」で区切りを入れてやらなければならないことに注意してください。もしも Carbon Copy が必要無ければ単に`<Return>`してください。

今度は `Subject:`と表題を聞いてきますので、何かわかりやすい表題をタイプしてください。Subject には漢字やかなはつかわず、アルファベットと数字程度で表現してください[9]。

以上の `To:, Cc:, Subject:` をタイプし終ると画面表示が変わり、手紙の内容を書くウィンドウが用意されます。

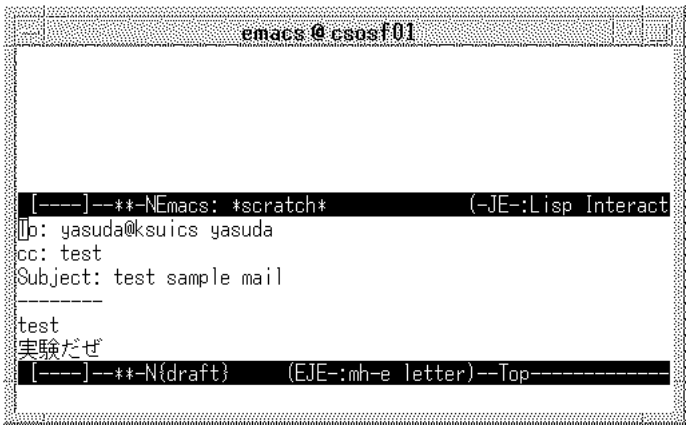

図 4.4: メイルの内容を書く

この状態をレターモードと呼んでいます。`To:, Cc:, Subject:` のすぐ下に`--------`という行がありますが、これ以降に本文を書いてください。この`---`行は MHE システムが必要とするもので、削除してはいけません。この状態で普通に Emacs によってファイルの内容を編集するときと同じ様にメイルの本文を編集する作業が出来ます。Emacs の操作に習熟すれば、メイルの内容としてどこかのファイルの内容を取り込んだり、さまざまな応用が利くようになるでしょう。

そうやってメイルの本文を書き終ったら、C-c C-c でメイルが発信されます。反対にメイルを書いている途中で、そのメイルを出したくなくなった場合は C-c C-q とします。するとエコーラインに以下のような確認の為の質問をしてきますので、y と答えてください。

[9] 最近 Subject にも漢字が利用できる場合がありますが、これは相手が Subject に漢字を適用できるシステムを持っているか、居ないかに依存しますからそれが確認できない限りは漢字は使わない方が無難です。因みに現在の cc 環境の MHE は漢字の Subject には対応していません。

```
Kill draft message? (y or n)
```

これで、そのメイルを放棄することが出来ます。

4.3.4 来たメイルの返事を書く

来たメイルに対する返事を書くには、返事を書こうとしているメイルを図 4.3 のような状態で読んでいるときに a キーを押します。すると、そのメイルに関係している人の誰に対しての返事を書くのかをエコーラインに以下のようなメッセージを表示して質問してきます。

```
Reply to whom:
```

この質問に対しては以下のいずれかで返事をします。

返答	意味
from	そのメイルの差出人、つまり From: に書かれている人に返事を送る。
<Return>	from に同じ
to	そのメイルの差出人、つまり From: に書かれている人を To:に、そのメイルの To: に書かれている人を Cc: に指定する。
cc	そのメイルの差出人、つまり From: に書かれている人を To:に、そのメイルの To: 及び From: に書かれている人を Cc: に指定する。
all	そのメイルの差出人、つまり From: に書かれている人を To:に、そのメイルの To: 及び Cc:及び From: に書かれている人を Cc: に指定する。

上記の表のいずれかによって返答すると図 4.4に非常に良く似た状態になり、返事のメイルの本文が書けるようになります。(これもレターモードです。)

電子メイルでの返事の作法

普通の手紙とは違って、電子メイルでは返事を書くときに前回の手紙の内容を引用することが簡単にできます。

返事を書こうとしているときに C-c C-y とすれば、そこに元となるメイルの内容が引用されます。引用の際は自動的に各行初めの部分に「この行は引用だよ」と言うことを示すために「>」記号が付加されます。

この作業ではメイルヘッダも本文も全ての部分が引用されますので、かなり不要な部分も含まれている可能性があることに注意してください。こうやって引用された元々のメイルのうち、今回のあなたのメイルでその返答となる部分以外の行を消去し、以下の例のように引用と返答とを交互に混ぜながら返事を書くと判りやすいメイルとなることでしょう。

> 先日は飲み会お疲れ様でした。
いえいえ。ありがとうございました。

> そこで話しておりました原稿の件ですが、酒の上の事とは言え、冗談と言う
> わけでもなく、本当にお願いしたいと思います。
ぎょぎょっ。やはり本気だったのですね。(^_^;)
ちょっと怖い気もしますが、了解しました。引き受けさせていただきます。

このような感じでしょうか。

4.3.5　メイルの整理

メイルをしばらく利用していると、メイルがたくさん溜ってきて邪魔になります。不要なメイルは削除することが出来ます。図 4.2 の状態で不要になったメイルの行の左にカーソルを移動し、ここで「.（ピリオド）」ならそのメイルの内容が読める、と言う状況でピリオドの代わりに d をタイプします。するとメイルの一覧表示の番号のすぐ右に D がマークとして表示されます。この段階は単に削除するべきメイルにマークを付けているだけで、実際の削除はまだ行われていません。どんどん削除したいメイルにマークを付け続けて行くことが出来ます。マークを付け間違えた場合は、D マークの行にカーソルを移動して今度は u キーをタイプすれば D マークが消え、マークが外すことができます。

マークされたメイルを実際に削除するには x もしくは e キーです。問い合わせなどはなく、すぐに削除を実行します。

4.3.6　メイルが来ているかどうか確認する

ずーっと Emacs を利用している人ならば、時々隙を見て M-x mh-rmail<Return>してメイルを確認すれば良いのですが、メイルが来ているかどうかを確認するためにわざわざ Emacs を起動するのが面倒だと言う人のために、メイルが来ているかどうかを簡単に確認する方法を紹介します。from コマンドです。from<Return>です。以下に例を示します。

```
csosf01(88)% from
From yasuda Tue Mar 22 18:51:30 1994
From yasuda Tue Mar 22 18:51:51 1994
csosf01(89)%
```

上記のように from コマンドは未読のメイルがあれば、一通あたり一行で「誰からか」「いつ届いたか」を表示します。もしも未読のメイルが一つもなければ from コマンドは何も表示せずに以下のように終了します。

```
csosf01(88)% from
csosf01(89)%
```

from コマンドで未読メイルが見つかった場合だけ、Emacs を起動して MHE で読めば良い、と言う事ですね。

4.3.7　メイルの実体はどこに？

MHE は各ユーザのホームディレクトリのすぐ下に Mail というディレクトリを用意し、メイルをそこに保存しています。普通に操作をしていれば、MHE は全て inbox というところにメイルを溜め込むのですが、その実体は Mail ディレクトリの下にある inbox というディレクトリです。そこに一通のメイルを一つのファイルとして保存しています。ファイル名には順番に付けられた番号が使われています。

具体的には~/Mail/inbox/1 などという名前でメイルが残っているはずです。自分で確かめてみると良いでしょう。

4.3.8　トラブルからの脱出

MHE を起動しようとすると.mh-profile がない、というエラーメッセージが表示されるんですけど

Cannot find MH profile /NF/home/..(略)../.mh_profile というメッセージがエコーラインに表示され、MHE が起動できない場合があるかも知れません。このような場合は一度 Emacs を終了してから、

おまじないとして inc コマンドを一度だけ実行してください。何か問い合わせをしてくるかも知れませんが、その時は y と答えてください。それでもう一度 Emacs を起動し、それから MHE を起動してみてください。

4.3.9 MHE もっともっと

ここに紹介したのは MHE の機能のうちの基本的なものだけです。MHE にはもっと様々な機能がありますが、それについてはここでは説明しません。附録に参考文献を挙げておきますので、それらを参照して下さい。

4.4 メイルを書くときの注意

ここでは操作方法では無く、メイルの中身の書き方について説明します。

初めてメイルを出すときに

始めのうちはいきなり学外や海外にメイルを出さずに、学内でメイルの練習をしてから送って下さい。練習の相手が見つからない場合は計算機センターに相談してください。

あまり大きなファイルを送らないこと

メイルによってファイルを転送することも出来ますが、、、

- どうしてもメイルによってファイル転送する場合は目安として 50 キロバイト[10]/メイル以下にしましょう。あまり大きなファイルを送るのは、参加組織に迷惑がかかります。
- どうしてもメイルによって大きなファイルを送らなければならないなら、1 メガバイト くらいまでなら分割して送っても大丈夫かもしれません。それ以上ならフロッピーやテープで送りましょう。

そのメイルは相手が読めるものですか？

現在のメイルがコンピュータによって処理されている限り、自分が利用しているコンピュータによって処理出来た文書が、相手が利用しているコンピュータによって復元出来るとは限らない事に注意しましょう。

- 漢字を含むメイルが必ず相手に読めるとは限りません。相手はひょっとしたら漢字が表示出来ないコンピュータを使ってメイルを読んでいるかもしれません。相手が漢字を読めるかどうか、まず最初に確認するのがよいでしょう。
- 基本的に Internet は JIS コード漢字を利用していますが、あなたが Internet を利用する窓口になるコンピュータによって漢字コードはまちまちです。たとえば計算機センターが管理している cc2000, csosf01〜41, ccns001〜015 に限っては標準の漢字コードとして EUC コードを採用していますが、メイルは JIS コードに変換されて送出されるように設定されています。どの漢字コードを適用すれば良いのか判らない場合はホストコンピュータの管理者に確認してください。
- Subject に漢字は使わないで下さい。

[10] バイトというのはコンピュータ上の資源の大きさを表す単位です。50 キロバイトと言えば漢字にして 25000 字、つまり原稿用紙 100 ページ強辺りとなります

- 使ってはならない文字として、半角カナ（カナ・キーを押して入力したもの）やメーカー独自の文字（(株),I,II,III,IV,V 等の文字が 1 文字で表されているもの、丸の中に文字が書いてあるものなど）があります。また、使わない方が良い文字としては、罫線素片等があります。

- NeXT メイルを送るときは相手が NeXT メイルをそのメイルアドレスで扱えることを確認してください。NeXT メイルは音や絵をメイルに含ませる事が出来ますが、相手も NeXT コンピュータを利用している必要が有ります。NeXT メイル特有の注意事項については別紙に詳しく説明が有りますので、NeXT コンピュータでメイルを利用される方は一度読んでください。

メイルの信頼性について

送ったメイルは相手に確実に届くことも、内容の完全な秘匿性も保証されていません。到着の確実性が要求される場合には、相手にそのメイルが届いたら折り返し届いた旨をメイルしてもらうように頼みましょう。ある程度の秘匿性が必要ならば暗号化するかまたは直接手渡しして下さい。

一行の長さについて

メイル配送を行うコンピュータのシステム自体は一行 255 文字までサポートしています。しかし、そのメイルの読み易さや返事を書く時に内容を参照することも考えて、漢字で約 35 文字以下、アルファベット（半角文字）で約 70 文字以下にするのが良いでしょう。

海外へのメイルについて

国内以上にアドレスや送るメイルの大きさの注意が必要です。また、海外のネットワークにはそれぞれのネットでの取り決めがあるので、それに従うようにしましょう。

メイルアドレスに工夫が必要な場合もあります。一般的には宛先のアドレスは `user@aaa.bbb.ccc` のように `.jp` で終らないものの、国内と同様の書式です。(`user` には相手のユーザ名を、`aaa,bbb,ccc` 等には相手のアドレスを入れます)

著作権および責任の所在

書いたメイルに関する責任は、書いた人の属する組織にあるのではなく、書いた人自身ににあります。また、著作権はそのメイルを書いた人にあります。

違法行為の禁止

例えば、メイルでのソフトの違法な流通などをやってはいけません。

4.5 GNUS：Emacs によるニュースの読み書き

GNUS は Emacs を利用してニュースを読み書きする機能を提供します。
Emacs と共に働きますから、Emacs の操作方法についてある程度理解していることを前提に説明します。

4.5.1 はじめに

ここでは以下の流れにしたがってニュースを扱う方法を説明します。

- GNUS の起動
- ニュースの記事を読む
- ニュースグループの選択
- ニュースの記事を投稿する
- ニュースの記事にフォローする
- 投稿した記事のキャンセル
- ニュースの記事にメイルで返事をする
- 古い記事を読み返す
- ニュースの記事の保存

GNUS の全ての操作は Emacs 上で行います。さあ、`emacs` コマンドで Emacs を起動してください。

4.5.2 GNUS の起動

ニュースを読むために、まず GNUS を起動します。Emacs が起動されている状態で M-`x gnus<Return>` とします。

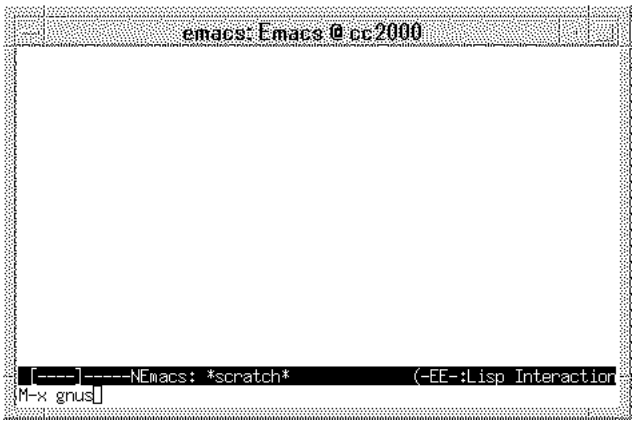

図 4.5: GNUS の起動

最初の GNUS の起動には数分以上かかる場合があります。これは全ての情報をゼロから構築しているためで、次回からは僅かの時間で起動できるようになるでしょう。以下のような状態になるまで、しばらくお待ちください。

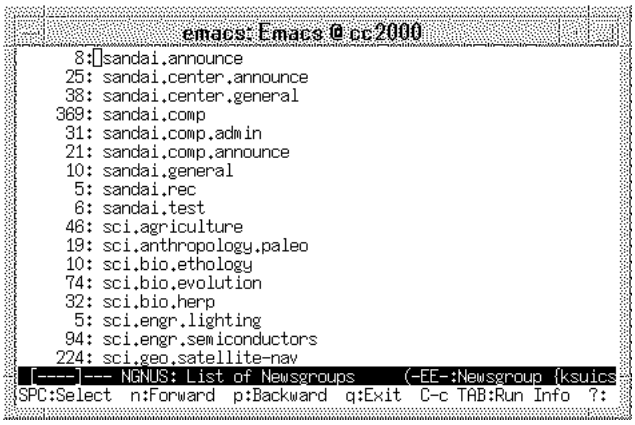

図 4.6: ニュースグループ一覧

4.5.3 記事を読む

図 4.6 のように、一つのニュースグループが一行で表わされ、一覧として表示されているでしょう。カーソルは一覧表示のニュースグループの名前の左側にあるはずです。ここでカーソルを上下させて、自分が読みたいニュースグループの行にカーソルを移動させます。カーソルの上下は Emacs 上でのファイルの編集の際のカーソルの上下と同じです。つまり上（一つ前の行）に移動したいときは上矢印（↑）もしくは C-p キー、下に移動したいときは下矢印（↓）もしくは C-n キーです。M-< や M-> で一番先頭や末尾の行への移動が出来ます。

このニュースグループの一覧が表示されている状態を「グループモード」と呼んでいます。

グループモードで、自分が読みたいニュースグループの行にカーソルを移動させて<Space>キーを押せば、そのニュースグループの記事の表題一覧と、そのなかの記事の内容を表示できる以下のような状態へと移行します。

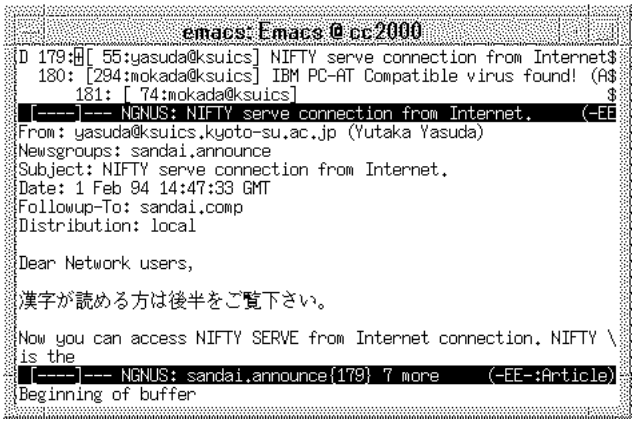

図 4.7: 記事の内容表示

この状態では Emacs のウィンドウが二分割され、上半分に先ほど選んだニュースグループの記事の表題の一覧が、下半分にそのニュースグループの先頭の記事の内容が表示されています。下半分に表示されている記事の内容が長すぎて Emacs のウィンドウに収まり切らないときは<Space>キーで一画面分スクロールさせることが出来ます。スクロールして過ぎてしまった部分の内容を巻き戻してみたい場合は<Delete>キーです。

次の記事を読む

今度は、ある特定のニュースグループの中の、一つの記事が一行で表わされ、一覧として表示されている訳です。カーソルは一覧表示の番号の右にあるはずです。ここでカーソルを上下させて、自分が読みたいニュースグループの行にカーソルを移動させます。カーソルの上下は Emacs 上でのファイルの編集の際のカーソルの上下と同じです。つまり上（一つ前の行）に移動したいときは上矢印（↑）もしくは C-p キー、下に移動したいときは下矢印（↓）もしくは C-n キーです。M-< や M-> で一番先頭や末尾の行への移動が出来ます。

自分が読みたい記事の行にカーソルを移動させて<Space>キーを押せば、その記事の内容が Emacs のウィンドウの下半分に表示されます。

また、記事の内容を読んでいるときに<Space>キーを押し続けて行けば、一つの記事を読み終った段階で自動的に次の記事に移動します。記事を読んでいる途中で n キーを押せば、今読んでいる記事の次の未読記事を表示します。p キーを押せば、今読んでいる記事の前の未読記事を表示します。

この、あるニュースグループを選択して、記事を次々と読んで行ける状態を「記事モード」もしくは「アーティクルモード」と呼んでいます。

q キーで記事モードから抜けてグループモードへ戻ります。

GNUS を終了する、再起動する

記事を読み終って、普通の Emacs の操作に戻りたいと思ったときはグループモードで q キーを押します。すると、エコーラインに以下のような表示が現れ本当に GNUS を終了するかどうか聞いてきます。

```
Are you sure you want to quit reading news? (y or n)
```

ここで y キーを押してやると GNUS を起動する前、つまり図 4.5の状態に戻ります。Emacs を終りたい場合はいつも通りに C-x C-c です。

再びニュースを読みたいと思ったときは単に M-x gnus<Return>とするだけです。

4.5.4 ニュースグループを選ぶ

グループモードでは、何も指示しなければうんざりするほどたくさんのニュースグループを表示します。もう読まないと決めてしまったニュースグループはこの一覧に表示させないようにすることができます。図 4.6 のようなグループモードでカーソルを読みたくないニュースグループの行に移動し、そこで u キーを押します。するとその行の左端に U マークが付きます。U マークを付け間違ってしまった時は、もう一度その行で u キーを押すことによって外すことができます。どんどん U マークを付けていったら l（英小文字の L）キーを押してみて下さい。今まで U マークがついていた行が表示されなくなりましたね。これで次から GNUS を使ってニュースを読む時に、全く読まないニュースグループの一覧が出てこなくなって随分楽になると言うわけです。ちなみに u は Unsubscribed の略で「購読しない」と言うことを意味しています。

Unsubscribed されているグループを再び表示させたい場合は L（今度は英大文字の L）キーを押して下さい。あるとき Unsubscribed してしまったニュースグループを、再び読みたいと思った時は、この L キーによってすべてのニュースグループを表示させた後で、目的のニュースグループについている U マークを再度 u キーによって外してやります。

4.5.5 記事を投稿する（けどちょっと待てよ）

せっかくニュースシステムが稼働しているのです。ただ読むだけで全く投稿しないのも面白くありません。ここは一つ何か書き込んでみましょう。

投稿する前にちょっと考えること

記事を投稿するのは GNUS で勿論可能なのですが、ニュースを読みはじめて間もなくの頃はとにかく読むのに徹して投稿は控えた方がいいと思います。これはネットワークニュースと言うものを良く理解しない内に「的外れな」「失礼な」もしくは「迷惑な」記事を投稿する危険があるからです。

ちょっと否定的に書きましたが基本的にはニュースも含めてネットワークサービスは参加しないと意味がありません。人の書いたものを読むだけでは面白さも半減です。あまり遠慮しすぎる必要もありません。投稿そのものには賛成します。ニュースを盛り上げるためにもぜひ投稿してください。

さあ投稿してみましょう

さてもう前置きは良いでしょう。何か記事を投稿してみましょう。記事の投稿には二つのケースが考えられます。

1. 全く新しい話題を投稿する
2. 人の記事に対する意見やコメントを投稿する。

ニュースをいくらか読んでいるうちに、この二種類の記事に遭遇すると思います。誰かが質問の記事を投稿し、他の誰かがそれに答えたりしているでしょう[11]。

[11] え？見たことがない？それは余りに経験が少なすぎます。もうちょっとニュースを読み込んでみてから投稿してはいかがでしょうか？

まず先に 1. のケースについて説明します。次に 2. のケースについて説明します。2. のケースのように人の記事に対して意見を加えて投稿することをフォローすると呼んでいます。

いずれにしてもはじめて投稿する時は、まず練習をするというのがお勧めです。実験もしくは練習専用のニュースグループとして sandai.test が用意してあります。このグループならば、投稿に失敗しても全然問題ありません。他のニュースグループで失敗した場合は最悪の場合失敗した記事が世界中をかけめぐる事になってしまい、大変迷惑です。繰り返しますが、最初の投稿の際は sandai.test ニュースグループで練習して、操作に慣れるのがお勧めです。

4.5.6 記事を投稿する

記事を投稿するにはグループモードもしくは記事モードで a キーを押します。すると以下のようなメッセージをエコーラインに表示して確認をしてきますので y で答えます。

```
Are you sure you want to post to all of USENET? (y or n)
```

もしグループモードで a キーを押した場合は、以下のようなメッセージをエコーラインに表示してどのニュースグループに投稿するのか聞いてきます。記事モードで a キーを押した場合は、キーを押した時のニュースグループが投稿先のニュースグループとなりますので、聞いてくることはありません。

```
Newsgroup:
```

これにはポストする記事のニュースグループをタイプします。
すると今度は以下のようなメッセージをエコーラインに表示して投稿する記事の表題を聞いてきます。

```
Subject:
```

これには何かわかりやすい表題をタイプしてください。Subject には漢字やかなはつかわず、アルファベットと数字程度で表現してください[12]。
すると今度は以下のようなメッセージをエコーラインに表示して投稿する記事の配布範囲を聞いてきます。

```
Distribution:
```

この Distribution は、投稿する記事の配布範囲を示しています。以下のいずれかを答えるようにします。

配布範囲	ニュースグループ	意味
sandai	sandai.general など	京都産業大学内のローカルニュースグループ用。京都産業大学以外に配布されることはない。
fj	fj.jokes など	国内のニュースグループ用。fj など日本語のニュースの配送を受けている地域以外に配送されることはない。
world	comp.sys.sun など	国際ニュースグループ用。comp, sci, soc など世界じゅうに配布される。

ここまで答えると Emacs のウィンドウの状態が変わって、記事の内容を編集できるようになります。
ここで記事を書くわけです。`Newsgroups:`, `Subject:`, `Distribution:` のすぐ下に

[12] 最近 Subject にも漢字が利用できる場合がありますが、これは相手が Subject に漢字を適用できるシステムを持っているか、居ないかに依存しますからそれが確認できない限りは漢字は使わない方が無難です。因みに現在の cc 環境の GNUS は漢字の Subject には対応していません。

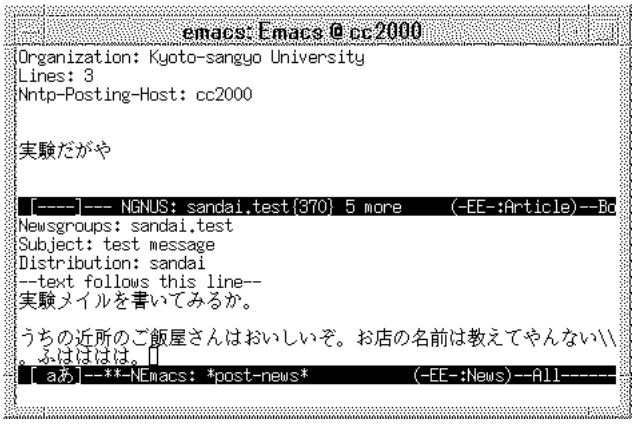

図 4.8: 記事を書く

```
--text follows this line--
```

という行がありますが、これ以降に本文を書いてください。この行は GNUS システムが必要とするもので、削除してはいけません。この状態で普通に Emacs によってファイルの内容を編集するときと同じ様にニュースの本文を編集する作業が出来ます。Emacs の操作に習熟すれば、記事の内容としてどこかのファイルの内容を取り込んだり、さまざまな応用が利くようになるでしょう。

そうやって記事の本文が書き終ったら、C-c C-c で記事が発信されます。

4.5.7 記事にフォローする

誰かが書いた記事を読んでいて、その記事に対して意見を添えたり、質問をしたり、疑問に答えたりするような記事を投稿する事をフォローと呼んでいます。フォローを行うには、その元となる記事を読んでいる状態で F キーを押します。

すると、単に新たな記事を書いて投稿するときと同じ様に以下のようなメッセージをエコーラインに表示し確認してきます。

```
Are you sure you want to followup to all of USENET? (y or n)
```

これにも y と答えてやります。すると今度は元記事の引用を行う際に、どのような記号を元記事の各行の前に付けるかを以下のように問うてきます。

```
Quoting marker (default: |) :
```

標準は「|」ですから、<Return>だけタイプすれば標準の記号が引用記号として使われます。例えばここで >><Return>とタイプすれば「>>」が引用記号として使われます。記事の引用としては一般的には「>」が多いようです。

引用の最初の部分には、その元記事の出所を示す機械的な情報が表示されます。例えば以下のようなものです。

```
In article <WORKER.94Mar22225450@cc2000.kyoto-su.ac.jp>
     worker@cc.kyoto-su.ac.jp (Environment Test Worker) writes:
```

これはオリジナルの記事の出所を示すものですから、残しておく方がよいでしょう。

あとはメイルのときにそうしたように、引用のうちから不要な部分を削除し、残した必要な部分に対するコメントを書き足して行きます。

そうやって記事の本文が書き終ったら、C-c C-c で記事が発信されます。

4.5.8 記事のキャンセル

自分で投稿した記事については、その取り消しが可能です。この作業を記事のキャンセルと呼んでいます。例えば投稿した後で、書いた記事の内容に誤りを発見した場合や、投稿するグループを大きく外してしまった時などにキャンセルしたくなると思います。しかしキャンセルは非常手段で、多くの Internet 上の人々に迷惑をかけてしまいます。何よりもまず不注意な記事の投稿をしないように心がけることが大切だということを忘れないでおいて下さい。

記事をキャンセルするには、図 4.7 の状態のように、キャンセルしたい記事を読んでいる状態で C キーを押します。すると以下のようなメッセージをエコーラインに表示して本当にキャンセルして良いかどうか確認してきます。

```
Do you really want to cancel this article? (yes or no)
```

ここで yes<Return> と答えることによって記事がキャンセルされます。

4.5.9 メイルで返事をする

ニュースの記事に対する返答というのは、同じく記事によって行なうフォローもあるが、時には記事の投稿者にメイルによって返答したくなる場合もあるでしょう。

記事に対してメイルで返事をするには、図 4.7 の状態のように、元の記事を読んでいる状態で r キーまたは R キーを押します。r と R の違いは元記事の内容の引用にあります。R は元記事の内容を引用してくれます。引用の際にはフォローの時と同じように引用記号を何にするか聞いてきます。いずれにしても以下のような状態になり、ここでメイルを書くわけです。

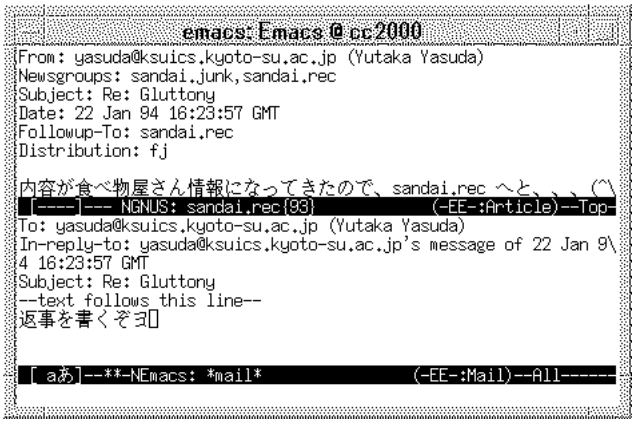

図 4.9: メイルで返事を書く

いつものように--text follows this line--の行から下にメッセージを書きます。この行を消してはいけません。内容の編集については普通の Emacs のつもりで操作が可能です。メイルの中身を書き終ったら C-c C-c でメイルを送信します。

4.5.10 古い記事を読み返す

GNUS は、未読記事が一つもないとそのニュースグループをグループモードでは表示しなくなりますし、記事モードでも普通では未読のものしか表示しません。一度読んでしまったニュースグループから、古い記事を再び読み返したい時は以下の手順を追うのがもっとも簡単です。

1. （もしあれば）そのニュースグループの記事を全部読んでしまう。未読の記事をゼロにする。
2. （もしグループリストに表示されていなければ）`L` キーでそのニュースグループを表示させる。
3. グループモードで未読ゼロのニュースグループを選択し、読む。

要するに GNUS は未読記事ゼロのニュースグループを読むという指示をしてやると、以前の記事を読み返せるように働いてくれるのです。残っている記事が非常に多いと、その中から最近のものをいくつ選んで表示するかを以下のようなメッセージで問い合わせてきます。

```
How many articles from sandai.general (default 197):
```

例では sandai.general ニュースグループの記事が結構多いので、そのうちいくら取り出すかを聞いています。どうやら全部で 197 記事あるようで、黙って `<Return>` すると 197 取り出してくれます。つまり自分が読み返したいと思う記事の数をタイプしてください。すると指定した数だけの既読記事が取り出され、図 4.7 のように表示されるでしょう。あとは今まで通りの操作で同じように扱えます。

4.5.11 記事の保存

ほとんど全てのニュースの記事は時間が経てばシステムの中から自動的に削除されてしまいます。つまり残しておきたい記事に関しては自分の手元に保存しておく必要があるのです。

記事を保存するには図 4.7の状態でメイルを読んでいる時に、`o` キーを押します。するとエコーラインに以下のようなメッセージを表示してどこに保存するかを聞いてきます。

```
Save article in folder [+sandai/test]? +
```

上の例ではたまたま sandai.test ニュースグループの記事を保存しようとしています。`+sandai/test` というフォルダーに保存すると言っていますので、ここではそれに従うことにします。単に `<Return>` です。

すると今度はエコーラインに以下のようなメッセージを表示して、指定のフォルダが存在しないが、作ってよいかどうかと聞いてきます。

```
Folder +sandai/test does not exist. Create it? (y or n)
```

仕方がないので、今度も言われた通りに `y` で答えます。するとようやく

```
Article saved in folder: +sandai/test
```

と言うメッセージが表示されて、記事が保存されました。

この保存された記事を正しく GNUS で読み返す方法もあるのですが、ここでは説明しません。その代わりにこの保存された記事の実体を説明します。

記事は各ユーザのホームディレクトリの下にある `Mail` というディレクトリの下に、更にニュースグループの名前を反映したディレクトリを作って保存されます。丁度メイルが MHE によって保存されるときの方法に似ていますね。上記の例では sandai.test というニュースグループのメッセージが一つ保存されまし

た。この様な場合は、あなたのホームディレクトリの下の Mail というディレクトリの下に更に sandai というディレクトリを作り、その更に下に test というディレクトリを作って、その更に下に 1 という名前のファイルとして先ほどの記事を保存しているのです。つまり~/Mail/sandai/test/1 という名前でファイルが出来ていると言うことです。自分で確認するのがよいでしょう。

4.5.12 記事を書く場合の注意

ここでは操作方法ではなく、記事の中身の書き方やその周辺の事柄について説明します。

どこのニュースグループに投稿するか？

ニュースグループは大きく分けて、3 つあります。

sandai.*	京都産業大学内のみのニュースグループ 配布範囲（以下 Distribution）は local を選択します。
fj.*	国内のニュースグループ Distribution は fj を選択します。
comp.*,rec.*,soc.*,sci.*など	海外のネットワーク（主にアメリカの USENET のニュースグループ） Distribution は world を選択します。

初めのうちは学内のニュースグループで練習をしてから学外のニュースグループを利用するようにしましょう。なお、上記の他にも海外のニュースグループがいくつかあります。

モデレータがいるニュースグループといないグループ

- モデレータ（そのニュースグループの管理人）がいないニュースグループは unmoderated と呼ばれています。すなわち、投稿すれば必ず全体に投稿されます。
- モデレータがいるニュースグループは moderated と呼ばれています。このニュースグループに投稿すると、記事はそのニュースグループを管理している人に自動的にメイルとして送られ、その人が有益であると考えた記事のみが全体に投稿されます。どのニュースグループが moderated かどうかは、fj に関しては、ときどき fj.guide.general にポストされる fj のニュースグループに関する記事を参照して下さい。

fj.test, sandai.test の使いかた

fj.test や sandai.test はポストやフォローのテストを行なうためのニュースグループです。普通に fj.test を用いると（Distribution:を特に指定しないと）、fj を購読している 1000 を越えるサイトに記事が流れることになります。その必要がない限り fj.test の利用は避け、できるだけローカルなテスト用ニュースグループ sandai.test で練習したりテストするようにしましょう。

記事の内容について

- 特にこのように書けというものはありませんが、自分の書いた記事は他人が読むことを考えて、読む人の立場に立って書きましょう。ときどき fj.questions.*などで、質問の内容がつかめないものがあります。

- ニュースのような文字による通信では、普段相手と会って話す時や電話で話す時と異なり、微妙なニュアンスが伝わりません。時々これに注意しなかったばかりに Internet で喧嘩をしていることがあります。注意しましょう。
- こう言ったことにならない為にも、しばらく人の投稿を見てネットワークニュース上でのマナーを学ぶのがいいと思います。喧嘩の元になるようなニュース上での「ひどい」記事は fj.misc というニュースグループなどでよく取り上げられますので参考にしてください。
- Subject の内容は、その記事の内容を反映しているものにすべきです。また、Subject には、漢字を使ってはいけません。
- 投稿の内容に関する責任は、書いた人の属する組織にあるのではなく、書いた人自身にあります。また、著作権はその記事を書いた人にあります。
- ニュースにおける誰かの発言は、発言者個人のものであって決して組織の代表としての発言ではないことに常に注意しておいてください。例えば某メーカーの社員が自社の製品について批判的なことを書いても、それは個人の感想に過ぎず、それ以上の意味はありません。その発言を取り上げてその会社の他の人に文句を言ったりするのは「反則」です。自分が買った製品が気に入らなくて「○×社の人説明して下さい」などと発言するのもナンセンスです。むしろこのような場合は「○×社の製品を買った人、良い解決法を考えましょう」とやる方が余程建設的です。
- 違法な内容の記事（歌詞のポストなどによる著作権の侵害、公道を時速○○○で走ったなど）をポストしてはいけません。当たり前の話ですが、反社会的な記事を投稿するべきではありません。それが問題として取り上げられ、Internet が社会から迫害されてしまう可能性だってあるのです。（現在の社会を構成する大部分の人にとって Internet なんてあっても無くてもどうでも良いものの一つなのです。）

フォローの方法

ニュースを読んでいるとわかりますが、相手の引用文の始めには「>」等をつけます。また、引用の前には、誰に対する引用なのかをはっきりさせます。引用の量はなるべく少なくなるようにして下さい。なお「>」で始まる行の行数は、自分がポストする記事全体の半分以下でなければなりません（ニュースリーダのデフォルトではこうなっています）。つまり引用大量、意見少量という事はするべきでない、という訳です。

signature のつけかた

signature はあなたの顔みたいなものです。個性的で印象に残るものにしましょう。行数があまり多くなると記事自体が読みにくくなってしまいます、少ない行数に情報をきれいにまとめるところが腕の見せ所です。目安としては 4 行以下がいいでしょう。

キャンセルについて

自分のポストした記事のみキャンセルを行なうことができます。ただし、キャンセルを行なうと、キャンセルを行なうためのコントロールメッセージが fj.*のニュースグループならば、fj のすべてのサイトに送られます。キャンセルを行なう時は十分慎重に行なって下さい。また、最初からキャンセルを行なわなくも済むようなポストをすべきです。

引用のマナー

自分のところに送られてきたメイルをニュースに引用する場合には、メイルの送り手の許可を得てからにしましょう。また、ニュースを引用する場合には、その引用した記事を特定できるような情報を付けて下さい。

第 5 章

UNIX もっともっと

ここでは計算機センターが管理している UNIX 環境を例に取りながら、より進んだ UNIX 環境の利用方法の説明と、さまざまな機能の紹介をします。ここでの説明は網羅的なものではなく、部分を取り上げて曖昧に説明しています。これは本文の読者のコンピュータそのものについての知識のハードルを高くしすぎないためで済ませるためです。読者が UNIX 、つまりコンピュータの利用に慣れて行くにしたがって自分でマニュアル、書籍、ネットワークなどから情報を常に取り込んで理解することが大切です。

5.1 ファイル名の補完

5.1.1 メタキャラクタ

今までファイルを指定するときにはファイル名を全てタイピングしていたと思います。でも、ファイル名を全部タイプしなくてももっと簡単にファイルを指定することが出来る場合があります。その為にファイル名を記述する部分に用いる文字として* と ? があります。これらの文字を UNIX ではメタキャラクタと呼んでいます。以下にそれぞれ詳しく説明します。

説明はカレントディレクトリ以下のファイルの状況が以下のようだと仮定して行います。

```
csosf01(88)% ls
bb      cc      log     log2    sample
csosf01(89)%
```

任意の文字列に適合するメタキャラクタ「*」

ここで「*」文字を紹介しましょう。例えば上記の例で log と log2 の両方のファイルの中身を見たい場合は、以下のようにすれば良いのです。

```
csosf01(92)% more l*
```

つまり l* とは「l で始まり、それ以降はどんな文字列でも構わないファイル」を列挙することを意味しています。その結果 more l* は more log log2 と解釈されて実行されます。

* が適用可能なのは何もファイル名の最後に限りません。

例えば上記の例で more *2 とすればそれは「どんな文字列で始まっても構わないから、最後が 2 で終わるファイル」と解釈され、log2 ファイルを差します。

o とすれば、それは「どんな文字列で始まっても構わないから、途中に o を含み、またどんな文字列で終わっても構わないファイル」と解釈され、log log2 の二つのファイルを差します。

* とすれば、それは「どんな文字列でも構わないファイル」と解釈され、bb cc log log2 sample の五つのファイルを差します。

* 文字はファイル名の記述のどこに幾つ現れても構いません。

また、適合する文字が無くても適合したとみなします。例えば log* とすれば「log で始まり、それ以降はどんな文字列でも構わないファイル」と解釈され、log log2 の二つのファイルを差します。

任意の一文字に適合するメタキャラクタ「?」

今度は「?」文字を紹介しましょう。例えば上記の例で bb と cc の両方のファイルの中身を見たい場合は、以下のようにすれば良いのです。

```
csosf01(92)% more ??
```

つまり?? とは「どんな文字でも構わないから、二つからなるファイル」を列挙することを意味しています。その結果 more ?? は more bb cc と解釈されて実行されます。

? が適用可能なのは何もファイル名の最後に限りません。例えば上記の例で more ?b とすればそれは「どんな文字でも構わないから一文字あって、次が b で終わるファイル」と解釈され、bb ファイルを差します。

? 文字はファイル名の記述のどこに幾つ現れても構いません。

ディレクトリに対してのメタキャラクタ

* と ? 文字はディレクトリに対しても適用可能です。例えば、ls */*2 などとすれば「カレントディレクトリ以下の全てのディレクトリの下にある、どのような文字列で始まっても構わないから最後は 2 で終わる名前を持つファイル」を見つけてその一覧を表示します。

5.1.2 対話的な補完

シェルからファイル名を記述しているとき、シェルが提供している対話的なファイル名の補完機能が幾つか利用できます。

再び説明はカレントディレクトリ以下のファイルの状況が以下のようだと仮定して行います。

```
csosf01(88)% ls
bb      cc      log     log2    sample
csosf01(89)%
```

ファイルの名前を途中までタイプしてくれる「<Tab>」

例えば more sample とタイプしたくて、more s までタイプしたとします。この状況で <Tab>キーを一度押せば more sample と、シェルの方で勝手にタイプしてくれて、sample から一つ離れた位置にカーソルが移動します。そこで<Return>を押せば良いと言うわけです。

つまりシェルは more s とタイプされた段階で、s に続くファイル名としてはこの状況では sample しか無いよ、と気を利かせてくれているのです。

今度は more log2 とタイプしたくて、more l までタイプしたとします。この状況で <Tab>キーを一度押せば more log と、シェルの方で勝手にタイプしてくれて、log に続く位置にカーソルが移動します。ベルが一度鳴るかも知れません。

今度はシェルは more l とタイプされた段階で、l に続くファイル名としてはこの状況では log と log2 しか無い事が判ります。だからとにかく一致している途中までタイプしてくれているのです。そこで残り不足している 2 をタイプして、<Return>を押せば良いと言うわけです。

ファイル名の一覧を表示する「C-d 」

<Tab>による補完を行って、カーソルがファイル名の記述のすぐとなりに来て（ベルが鳴る）時は、まだ残りがあるかも知れないよ、という意味でした。この段階で、では一体どのような名前のファイルが可能性として残っているのかを確認するには C-d キーを押します。そうすれば以下のように、まだ可能性のあるファイル名の一覧を挙げてくれます。

```
csosf01(86)% more log   ... ここで空白などあけずに<Control>-d
log   log2
csosf01(86)% more log
```

こうしてファイル名が長ったらしい場合でも、<Tab>キーと C-d を組み合わせて行けば、少ないタイピングで間違いなくそのファイルを指定することが出来ると言うわけです。

5.2 ファイルのアクセス権

誰か友達がファイルを作っていて、そのファイルをあなたが自分のホームディレクトリ以下にコピーしたいと思ったとします[1]。ファイルの階層構造を理解したあなたは早速以下のようなコマンドでコピーをしようとするでしょう。

```
csosf41(85)% cp ~tanaka/sample.tex ~
cp: /NF/home/g840/tanaka/sample.tex: Permission denied
csosf41(86)%
```

多くの場合は上の例のように失敗してしまいます。このエラーメッセージ「Permission denied」は UNIX を使っていると、時々みかけるエラーですね。これは「あなたにはアクセス権限が無いよ」ということを意味しています。

5.2.1 アクセス権

UNIX コンピュータはみんなで使うコンピュータであることは初めに説明しました。そのため、あなたのホームディレクトリのとなりには他人のホームディレクトリがあったりします。つまり誰でも他人のファイルの置き場所が大体わかっているわけです。これでは誰でも他人のファイルを覗いたり、書き込んだり、消去したりできてしまいます。

これは安全上（セキュリティ）の問題です。たった一人の人が専有して一台のコンピュータを使用し続けるのなら、そのコンピュータを自分の部屋に置いて、その部屋の入口に鍵を掛けておけば良いのです。でも cc 環境の UNIX のようにみんなで使うコンピュータは閉じ込めるわけにはいきませんからセキュリティを確保するためには何か別の対策が必要です。

そこでみんなで使うコンピュータの多くではアクセス権限という考え方を採り入れています。つまりコンピュータ上の資源にはそれぞれ誰が使えるかという情報が書いてあるのです。逆にその情報から洩れた人は使えないということです。コンピュータ関係の世界では、この「(資源を）使う」という事を「アクセスする」と表現することがあります。「アクセス権」とはつまり（資源に対する）「使用権限」と言う意味です。

そして、cc 環境では安全のため（もしくはプライバシーのため）に「利用者のホームディレクトリ以下のファイルは、利用者自身しかアクセスできない」という設定になっています。先の「Permission denied」はこの制限に引っかかったためのエラーメッセージだったのです。

5.2.2 UNIX におけるアクセス権

UNIX では全てのファイルに常にアクセス権が設定されています。ファイルが利用者によって作成されると、利用者がまずファイルの所有者となり、自動的にアクセス権が設定されます。アクセス権の設定は後で所有者によっていくらでも変更することが出来ます。

アクセス権は具体的には以下の 3 種類の項目に対してそれぞれ許可を与えることによって設定します。

[1] もちろんその友達の了解を得て、ですよ！勝手に人のファイルをコピーしてはいけません。

種類 (略号)	ファイルの種類	設定することによって許可されるアクセスの内容
読み出し (r)	ファイル	そのファイルの中身をとり出す
	ディレクトリ	そのディレクトリ以下のファイルの一覧を表示する
書き込み (w)	ファイル	そのファイルの中身を書き変える、もしくは消去する
	ディレクトリ	そのディレクトリ以下にファイルまたはディレクトリを新たに作る、もしくはそのディレクトリを消去する
実行 (x)	ファイル	そのファイルを実行する
	ディレクトリ	そのディレクトリ以下に `cd` コマンドで移動する

上記の 3 種類のアクセス権の設定項目は「所有者 (user)」と「グループ (group)」と「その他の人 (other)」のそれぞれの対象に対して別個に割り当てられます。つまりあるファイルに対して「所有者は読み書きできて、グループは読むだけで、その他の人は読むことすら出来ない」という設定が出来るのです。

所有者とはそのファイルを作った人です。グループについてはここでは説明しません。自分がどのグループに属しているかは `id` コマンドで確認することが出来ますので、参考にして下さい。学生は全員 student グループの一員です。教員は全員 teach グループの一員です。その他の人と言うのは所有者でもグループのメンバーでも無い利用者のことです。

5.2.3 アクセス権限を調べる

`ls -lg` コマンドで、ファイルのアクセス権限を含めた詳細な情報を得ることが出来ます。書式は以下の通りです。

`ls -lg [ファイル名...]`

以下に実行例を示します。

```
csosf41(82)% ls -lg
total 20
drwxr-xr-x   2 yasuda   student      512 08 月 29 日 1992  Apps
drwxr-xr-x  11 yasuda   student      512 12 月 17 日 15:08 Library
drwxr-xr-x   5 yasuda   student      512 11 月 02 日 17:25 Mail
drwxr-xr-x   2 yasuda   student      512 03 月 14 日 18:26 Unix
-rw-r--r--   1 yasuda   system      4096 03 月 08 日 23:22 jsykojin.dic
-rw-r--r--   1 yasuda   student      523 03 月 08 日 23:24 log
drwxr-xr-x   2 yasuda   student      512 03 月 14 日 18:26 memo
csosf41(83)%
```

これ以降に以下の一行を取りだして、詳細に説明します。

```
-rw-r--r--   1 yasuda   student      523 03 月 08 日 23:24 log
```

-rw-r--r--	ファイルの種類とアクセス権限についての情報。(さらに後述)
1	リンク数。ここでは説明しない。
yasuda	所有者
student	グループ
523	ファイルの大きさ。単位は Byte(バイト)。ここでは説明しない。
03 月 08 日 23:24	作成年月日
log	ファイル名

この、ロングフォーマットで得られる出力の最初の部分(-rw-r--r--)に注目して下さい。以下にそれぞれの桁について説明します。

桁位置 (例での値)	意味
1(-)	ファイルの種類を表す。d ならディレクトリ、- ならファイル。
2(r)	所有者に対する読みだし許可を表す。r なら許可、- なら禁止。
3(w)	所有者に対する書き込み許可を表す。w なら許可、- なら禁止。
4(-)	所有者に対する実行許可を表す。x なら許可、- なら禁止。
5(r)	グループに対する読みだし許可を表す。r なら許可、- なら禁止。
6(-)	グループに対する書き込み許可を表す。w なら許可、- なら禁止。
7(-)	グループに対する実行許可を表す。x なら許可、- なら禁止。
8(r)	その他に対する読みだし許可を表す。r なら許可、- なら禁止。
9(-)	その他に対する書き込み許可を表す。w なら許可、- なら禁止。
10(-)	その他に対する実行許可を表す。x なら許可、- なら禁止。

つまり以下のような表組を一列に引き延ばして書いたようなものです。

対象	読みだし	書き込み	実行
所有者	r	w	-
グループ	r	-	-
その他	r	-	-

5.2.4 アクセス権限を設定する

アクセス権限は所有者によって設定を変更することが出来ます。chmod[2] コマンドを利用します。書式は以下の通りです。

```
chmod mode file...
```

chmod コマンドは指定の file のアクセス権を mode の指定に従って変更します。mode は 3 つの部分からなる文字列で、**対象... オペレータ 内容** となっています。以下にそれぞれの部分に与え得る記号とその意味を説明します。

[2] change mode の略のつもりなのです

対象	u	所有者
	g	グループ
	o	その他の人
	a	全ての人
オペレータ	+	追加
	-	取消
設定内容	r	読みだし
	w	書き込み
	x	実行

以下に chmod の具体例を挙げます。

1. `chmod a+r log` 全ての利用者に読みだし権限を与える
2. `chmod ug+rw log` 所有者とグループメンバーに読み出しと書き込み権限を与える
3. `chmod go-rw log` 自分以外の利用者から読み書き出来ないようにする

+ と- オペレータは元のアクセス権に新たな設定を「付加する」ように働きます。つまり元の設定で、影響を受けない部分はそのまま残ります。例えば 1. の例では書き込み権限や実行権限などには影響を与えず、もとの log ファイルが持っていたはずの書き込みに関するアクセス権限の設定はそのまま残ります。

いろいろ試して、chmod の振舞いを理解するのがいいでしょう。

ちょっとマニアックな chmod の使い方

もっと直接的にアクセス権の設定をするために、chmod コマンドにはもう一つの mode 文字列の与え方があります。アクセス権限を表す rw-r--r-- などの文字列を以下のようにして二進数に見立てて計算するのです。

```
rw-r--r-- → rw-,r--,r--
110100100 → 110,100,100 → 6,4,4 → 644
```

つまり rwx の部分で 3 つに分け、それぞれで二進数のつもりで計算するのです。r が 2^2 の桁、w が 2^1 の桁、x が 2^0 の桁、と言う訳です。計算すると、rw-r--r-- は rw- , r-- , r-- と分割され、$2^2 * 1 + 2^1 * 1 + 2^0 * 0$, $2^2 * 1 + 2^1 * 0 + 2^0 * 0$, $2^2 * 1 + 2^1 * 0 + 2^0 * 0$ となり、最終的に 6,4,4 となります。コマンドとしては chmod 644 log で完全に rw-r--r-- を意味するアクセス権の設定が出来ます。

5.3 Emacs

Emacs[3]については現在作成中です。もうしばらくお待ち下さい。

5.3.1 基本の基本

起動と終了

画面

キーバインドとコマンド

5.3.2 基本操作

カーソルの移動

繰返し

ブロック移動

5.3.3 トラブルからの脱出

コマンドの中断

Control-G

アンドゥ

5.3.4 ヘルプ

マルチバッファ

モード

5.3.5 ファイルの扱い

ファイルの読み込み

ファイルの挿入

ファイルの保存

別のファイルへの保存

5.3.6 ウィンドウ

ウィンドウの分割

ウィンドウ間の移動

ウィンドウの境界線の移動

ウィンドウの消去

[3]「いーまっくす」と読んでください。

5.4 シェルよもう一度

5.4.1 シェル変数と環境変数

シェルには[4]シェル変数と環境変数と呼ばれる、コマンドなどの動作に影響を与えるものがあります。これを設定し直すことによって、もう少しあなたにとって都合の良い UNIX 環境が得られるかも知れません。ここではシェル変数や環境変数の本質については触れずに、その設定例やヒントを紹介します。本質的なことについては附録の参考文献などからシェルに関する記述を参照してください。

シェル変数の表示、設定

現在設定されているシェル変数に何が有るかを調べるには `set` コマンドを利用します。また、シェル変数の内容を変更するにも `set` コマンドを利用します。`set` コマンドには以下の 2 通りの書き方があります。

1. `set`
2. `set var = string`

1. の書き方の場合、現在設定されている全てのシェル変数を表示します。

2. の書き方の場合、シェル変数 `var` の内容を `string` に設定します。

また、`echo` コマンドを使って特定のシェル変数の内容を表示することも出来ます。`echo $var <Return>` とすると、`var` という名前のシェル変数の内容を表示します。シェル変数は慣例として英小文字と数字で構成されています。

例えば tcsh ではしばらく使わないでほうっておいた場合、自動的にそのシェルを終了する機能があります。これを制御しているのはシェル変数 `autologout` です。`echo $autologout<Return>`とする事によって、現在設定されている放置時間が分単位で確認できると思います。これは不用意に切り忘れてしまったような不要なシェルを放置しないための処置なのですが、この時間が短すぎて困る環境の人もいるでしょう。そのような人は、以下のようにして放置時間を延長できます。

```
csosf01(81)% set autologout=300
csosf01(82)%
```

また、例えばカレントディレクトリの情報はシェル変数 $cwd[5]に格納されていますので、`echo $cwd` などとして表示させることが出来ます。それからホームディレクトリの情報はシェル変数 $home に格納されています。これまた `echo $home` などとして表示させる事が出来ます。

`echo` コマンドなど、一般のコマンドでシェル変数を扱うときは常に `$` 記号がシェル変数の名前の前に付けます。`set` コマンドでシェル変数の名前を指定するときは `$` 記号を付けないことに注意してください。

環境変数の表示、設定

現在設定されている環境変数に何が有るかを調べるには `env` コマンド[6]を利用します。また、シェル変数の内容を変更するには `setenv` コマンドを利用します。

---

[4]特に csh と、それを模倣している tcsh には
[5]current working directory の略なのです
[6]UNIX の種類によっては setenv コマンドまたは printenv コマンドで行うというものもあります。

env

現在設定されている全ての環境変数を表示します。

setenv VAR string

環境変数 VAR の内容を string に設定します。

また、echo コマンドを使って特定の環境変数の内容を表示することも出来ます。echo $VAR <Return> とすると、VAR という名前の環境変数の内容を表示します。環境変数は慣例として英大文字と数字で構成されています。

例えばプリンタの操作をするときは lpr -Pcspr01 sample.tex などして常にプリンタ名を指定するようにしていました。環境変数 PRINTER が設定されている場合は、プリンタに関する各種コマンドは（ -P オプションを省略した場合）環境変数 PRINTER に指定されたプリンタに対して処理を実行しようとします。

```
csosf01(81)% setenv PRINTER cspr01
csosf01(82)% lpr sample.tex
csosf02(83)%
```

echo コマンドなど、一般のコマンドで環境変数を扱うときは常に $ 記号が環境変数の名前の前に付けます。setenv コマンドで環境変数の名前を指定するときは $ 記号を付けないことに注意してください。

5.4.2 隠れたファイル

UNIX ではファイル名が「. （ピリオド）」で始まるファイルは特に指定しないと見せないようにするという処置が施されています。これは . で始まるファイルは主に様々な環境設定に用いられると言う事を前提にしているからです。つまり逆にいうと、ファイルとして作らなければいけないのだけれど、出来れば余り目に付かない方が良いというようなファイルは . で始まるような名前が付けてあると言うことです。

この隠れたファイルも含めてファイルの一覧を見るには ls コマンドに -a オプション[7]を与えます。

ls -a<Return>です。一度試してください。気が付かないうちに非常に多くの隠しファイルが出来ているのが判りますね。

ls * などとすれば判りますが、これらのファイルはメタキャラクタの一部にも反応しません。もしもこれらのファイルをコマンドの引数として指定したければ単に cat .login などとそのまま . 付きでファイル名に書けば良いのです。

5.4.3 リダイレクション

UNIX の標準的なコマンドには、標準入力と標準出力と呼ばれる入出力を処理対象としているものが数多くあります。普通にシェルを使っているときは、標準入力はキーボード入力、標準出力はディスプレイ表示に割り当てられています。

この例として bc コマンドを挙げておきます。bc コマンドを実行して四則演算をキーボードからタイプすると、画面上に計算結果を表示します。これはつまり bc コマンドは標準入力から四則演算を受け付けて、計算結果を標準出力に流していると言うことの結果です。

[7] all の積りでしょうか？

さて、標準入力、標準出力はそれぞれ他のファイルに割り当てることが出来ます。この標準入出力のファイルへの割り当て直しを「リダイレクション[8]」と呼んでいます。

標準入力のリダイレクション

標準入力のリダイレクションは「<」記号で表現します。

コマンド [オプション] [引数...] < ファイル名 と書いて、コマンドの標準入力を指定のファイルに割り当て直します。例えば bc コマンドの標準入力をキーボードではなくファイルに割り当てて見ましょう。bc < **ファイル名** <Return>です。

まず bc.in というファイル名で、四則演算を並べたファイルを用意します。Emacs で新規に作成、編集して下さい。以下に出来上りを cat した例を示します。簡単な四則演算を 3 行用意しました。

```
csosf01(85)% cat bc.in
1 + 2
2 * 3
4 / 2
csosf01(86)%
```

bc コマンドの標準入力に対するリダイレクションによってファイルの内容を四則演算の式の列として実行され、画面すなわち標準出力に結果が表示されます。

```
csosf01(86)% bc < bc.in
3
6
2
csosf01(87)%
```

標準出力のリダイレクション

標準出力のリダイレクションは「>」記号で表現します。

コマンド [オプション] [引数...] > ファイル名 と書いて、コマンドの標準出力を指定のファイルに割り当て直します。例えば bc コマンドの標準出力をディスプレイではなくファイルに割り当てて見ましょう。bc > **ファイル名** <Return>です。

```
csosf01(88)% bc > bc.out
1 + 2
2 * 3
quit
csosf01(89)%
```

cat コマンドでリダイレクションによってファイルに結果が残っていることを確認しましょう。

```
csosf01(89)% cat bc.out
3
6
csosf01(90)%
```

[8] redirection 宛名を変える

標準出力のリダイレクションで追加書き

標準出力のリダイレクションを既存のファイルに対して指定すると、単純に上書きしてしまい、そのファイルの元の内容は失われてしまいます。しかし追加書きするようなリダイレクションもあります。標準出力の追加リダイレクションは「>>」記号で表現します。

コマンド [オプション] [引数...] >> ファイル名 と書いて、コマンドの標準出力を指定のファイルに追加するように割り当て直します。例えば bc コマンドの標準出力をファイル追加するように割り当てて見ましょう。**bc >> ファイル名** <Return>です。

```
csosf01(90)% bc >> bc.out
5 * 2
quit
csosf01(91)%
```

cat コマンドでリダイレクションによってファイルにもとの結果に加えて新しい結果が追加されているのを確認しましょう。

```
csosf01(91)% cat bc.out
3
6
10
csosf01(92)%
```

標準入力、出力の両方のリダイレクション

先ほどの bc.in ファイルを利用して標準入力をそこから、標準出力をこれまた先ほどの bc.out ファイルに割り当てて見ましょう。以下にその例を挙げます。例では cat コマンドでファイルの中身を確認しています。

```
csosf01(92)% bc < bc.in > bc.out
csosf01(93)% cat bc.out
3
6
2
csosf01(94)%
```

5.4.4 パイプ

bc などのように、標準入力から何かを受取り、標準出力にその結果を返すようなコマンドをフィルタコマンド[9]と呼んでいます。

先ほどの例で、bc.out というファイルが結果として出来上がっていると思いますが、例えばこのファイルの中身を数字の小さいもの順に並べ替える[10]為に、sort コマンドの利用を考えます。sort もフィルタコマンドですから、実行は以下のようにする事になるでしょう。-n オプションはソートの順番を数値の小さいもの順にするためのオプションです。

[9] あたかもフィルターの様に働くという意味ですね。
[10] 並べ替える作業を一般的に sort（ソート）と呼んでいます。

```
csosf01(94)% bc < bc.in > bc.out
csosf01(95)% sort -n < bc.out
2
3
6
csosf01(96)%
```

単純なコマンドでも、それを組み合わせて実行すると、なかなか便利な[11]ものです。ところで上記のコマンド 2 つの組み合わせを、もっと簡単に表現できるような仕掛けが UNIX には用意されています。以下の例を見てください。

```
csosf01(97)% bc < bc.in | sort -n
2
3
6
csosf01(98)%
```

2 つのコマンドを一行で書いています。リダイレクションを使わないために中間的に発生していた bc.out ファイルも必要ありません。このような、コマンドとコマンドをつなぐ「|」記号を「パイプ」と呼んでいます。パイプがあれば、パイプの左のフィルタコマンドは標準出力をパイプの右のコマンドの標準入力につなぎ合わせて実行くれます。

例えば cat コマンドもフィルタコマンドですから、上の例は以下のようにも書き換えられます。

```
csosf01(98)% cat bc.in | bc | sort -n
2
3
6
csosf01(99)%
```

パイプは幾つでも重ねて使うことが出来ます。

あまり実用的な例ではありありませんが、echo コマンドを使えば簡単な計算を標準入力からキーボード入力しないで処理するような組み合わせも出来ます。

```
csosf01(99)% echo '3 * 4' | bc
12
csosf01(100)%
```

UNIX ではパイプとリダイレクションを応用することによって、各種のコマンドを柔軟に組み合わせて実行することが出来ます。

5.4.5 シェルの鬼へのヒント

以下に少しだけシェルの鬼[12]へのヒントをあげておきます。

[11] ちょっと意味のない例しか挙げられませんでしたが

[12] ここでキックの鬼と言ってもわかる人は、、、

キーボードからの標準入力の終了

今回用意した bc.in ファイルなどは Emacs を使わなくても以下のようなやり方で作成できます。

```
csosf01(100)% cat > bc.in
```

cat コマンドは引数としてファイル名が与えられていない時は、入力を標準入力から行ないますから、それをキーボードから入力して出力先を画面ではなくファイルにすると言うわけです。

上のコマンドを実行して、タイプする内容は 1 + 2<Return>2 * 3<Return>4 / 2<Return>C-d です。最後の C-d はファイルの終了を意味します。キーボードからファイルの終了[13]を入力するのは、大抵の場合この C-d です。

今まで黙っていましたが、bc コマンドは、サブコマンド quit ばかりでなく、このファイルの終了を受けとっても終了します。だからこそ cat bc.in | bc で文句も言わずにちゃんと処理を終了したのです。

複数のコマンドを一行に書く

「;」を利用して、複数のコマンドを一行に書くことが出来ます。例えば以下のような感じです。

```
csosf01(101)% date ; hostname ; whoami ; id
1994 年 03 月 12 日（土）16 時 13 分 33 秒
csosf01
yasuda
uid=2126(yasuda) gid=700(admin) groups=500(clerk)
csosf01(102)%
```

ところで上の 4 つのコマンドの結果をぜんぶ一つのファイルにリダイレクションしたい場合は、以下のようにします。

```
csosf01(102)% ( date ; hostname ; whoami ; id ) > hostinfo.txt
```

このような書き方で、別々に処理することももちろん出来ます。

```
csosf01(103)% ( date ; hostname ) > hostinfo1.txt ; ( whoami ; id ) > hostinfo2.txt
```

ファイルにコマンドを書いておく

シェルもフィルタコマンドです。ですからファイルにコマンド列を書いておいて、それをシェルの標準入力に与えて処理させることが出来ます。cc 環境標準のシェルは tcsh ですから、シェルのコマンド名は tcsh です。以下に例を挙げておきます。

```
csosf01(104)% cat batch
echo 'Here is my session information.'
date
hostname
whoami
id
csosf41(82)% tcsh < batch
```

対話形式ではないので警告のメッセージが表示される時がありますが、とりあえず結果は表示されると思います。

[13] 一般的に end of file と表現します

5.4.6　シェルよ永遠に

シェルの説明をすると言うのにあの話（もしくはあの話）をしないとは何事か！と、お怒りの方もおられるでしょう。シェルには非常に多くの機能があります。さすがに全部書くのは無理ですので、これ以降は付録の参考文献を参照して下さい。

第 6 章

Mathematica

6.1 Mathematicaってなあに？

Mathematica は Wolfram Research,Inc から発売されている数学とその応用のための汎用プログラムです。と、いっても何のことか解らない人も多いと思われるので、ともかく動かして試してみましょう。

6.2 ともかく起動、そしてやってみる！

さて、産大では何処に行けば Mathematica に出会えるか、です。以下のような場所に Mathematica は在ります。

1. ２号館４階２１情報処理教室。
2. 計算機科学研究所３階の C ２情報処理教室。
3. 計算機科学研究所３階の C ３情報処理教室。
4. あちこちにあるマッキントッシュの中。

Mathematica の起動のやり方の説明

１）２号館４階２１情報処理教室

まず csosf* に login し、アプリケーションで cc2000 のコマンドツールを呼び出します。そして math と打ち込むだけです。そうするとプロンプト In[1]:= が出てきます。これで Mathematica は入力待ちの状態です。[1]

２）計算機科学研究所３階の C ２情報処理教室

classic01 から classic07 までは /usr/local/bin/math と打ち込むだけです。すると、プロンプト In[1]:= が出てきます。

[1] 一つ注意があります。ここでは現在 figure3.2 のようなグラフィックが出ません。どうしてもここでグラフィックを見たい場合は Mathematica のプログラムをファイルにおとして、それをさらに psfix などで EPS ファイルに変換して、それをさらに TeX に取り込んで表示する、という方法もあります。

３と４）ノートブックについて

Next や Mac ではアイコンをクリックするとファイルが直接ウインドウとして開かれます。そこでは In[1]:= はでてきていません。しかし In[1]:= に対して入力するのと同じようにそのファイル上で

4+5*5　⇐ 入力

と、いうふうに **Return ではなくて Shift-Return または Enter で**打ち込みます。すると、

In[1]:= 4+5*5　⇐ 出力
Out[1]:= 29　⇐ 出力

と、出力され正しくノートと同じように全部をファイル上に書き込んでいけます。これがノートブック機能です。

起動ができたら例として、まず $\sqrt{10}$ の計算をやってみましょう。

In[n]:=　N[Sqrt[10], 40]
Out[n]:=　3.1622776016837933199889354443271853371955

図 6.1: 数値計算

上の例で、*In[n]:=* [2] は入力を受け付けるプロンプト。*Out[n]:=* はその入力に対する出力が右にあることを示していて、 $\sqrt{10}$ を 40 桁の精度で計算しています。

また、以下のようなグラフィックを描くことも可能です。

In[2]:=　Plot[Sin[x], {x, 0, 2Pi}]
Out[2]:=　-Graphics-

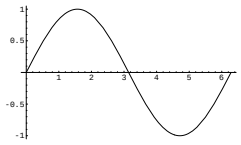

図 6.2: $y = \sin(x)$ のグラフ

★）終了

終了する時にはいずれの場合も Quit[] または Quit と打ち込むか Ctrol-d とします。

[2] カッコの中の n は自然数です。入力を受け付ける度に数字が増えていきます。

6.3 Mathematica の簡単な命令

1) : 数値計算

- 足し算: x+y
- 引き算: x-y
- : x*y または x y

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

- 割算: x/y
- 乗巾: xˆy
- 厳密な計算:
  - 通常の電卓と違い、厳密な計算も行ないます。

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

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

- 近似値計算:
  - //N を使えば近似値で出ます。

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

  - 実数値で入力しても近似値となります。

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

2) : 数学関数

数学関数は、普通の英語の名前がついていて大文字から始り引数は [ ] の中に入れるようになっています。また、複数の引数をとる場合は引数と引数の間を「 , 」で区切ります[3]。例えば以下のような関数があります。

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

厳密な計算、近似値計算 //N の使い方は 1) : 数値計算と同じです。

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

[3] [x y] では [x*y] というふうにかけ算になってしまいます。

- In[1]:= ??Sin

とすると、Sin[z] gives the sine of z と出てきます。[4] つまり、 Sin[ ] も数学関数として組み込まれていて、

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

というふうになります。

3) : 代数解析

- Expand[ ]: 式を展開

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

- Factor[ ]: 因数分解

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

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

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

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

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

4) : グラフィック

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

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

[4] これは次章でも説明します。

6.4　入出力一般

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

直接入力する場合のコマンド色々

1 ）前の結果の取り込み

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

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

2 ）変数

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

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

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

3 ）情報を引き出す

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

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

例えば N の使い方を忘れてしまったとしましょう。そのときは、

- In[1]:= ??N ⇐ 入力

と、すると、

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

などと出てきて、使い方、内容などを表示します。

ファイル

Mathematica はエディターを内蔵していません。よって、直接、Plot3D[sin[x y], x, 0, 2Pi, y, 0, 2Pi] と打ち込んでしまったら気が滅入って来ます[5]。そこで、普通にファイルをつくってそれを Mathematica に読み込ませましょう。

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

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

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

6.5 さらに進みたい人には

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

[5] sin ではなくて Sin と書かなければなりませんよね。
[6] C2 情報処理教室に何冊かあります。

第 7 章

NQS

大勢の人たちで一つのコンピュータを共用していて、科学技術計算などの大規模な処理を実行したい人がたくさんその中に含まれていたとします。例えば 10 人の利用者がそれぞれ 3 時間かかるような大規模な計算を同時に同じコンピュータに掛けたとすると、全員の処理が終了するのが合計 30 時間よりも遥かに長くなったりする場合が往々にしてあります。これは一時に負荷のかかる計算をやらせた為に、計算機が過負荷状態になってしまうことから来ています。これを回避する為には、一つずつプログラムを実行し、先のプログラムの実行が終ったら次のプログラムの実行に掛かるようにするのが最善です。

このようなシステムは一般にバッチシステムと言われ、科学技術計算を良く行なうコンピュータ[1]には大抵実装されているもので、そのような要求のある人たちにとっては大変便利なもので、VMS[2]や MSP[3] などの OS には実装されています。

しかし、その生まれの経緯から Unix にはもともとバッチキューという概念がありません。NQS はこのバッチキューを Unix 上で実現するソフトウェアです。cc 環境では cc2000 マシン（SPARCcenter2000）に実装されています。ここでは cc2000 上の NQS を例にとって、NQS の使い方を説明します。

7.1 今どんなバッチキューがあるか？

バッチキューの確認には `qstat -b` とします。

```
% qstat -b
```

です。以下に例を示しておきます。

```
cc2000(411)% qstat -b
==========================================
NQS Version: 2.3  BATCH QUEUES on cc2000
==========================================
QUEUE NAME      STATUS   TOTAL   RUNNING  QUEUED   HELD TRANSISTION
------------------------------------------------------------------------------
BATCH           AVAILBL    0      0/4/4      0       0       0
I               AVAILBL    0      0/2/2      0       0       0
cc2000(412)%
```

[1] もしくは古典的なコンピュータ
[2] DEC-3500 などの OS です。
[3] FACOM の OS です。

STATUS のところが available になっていますから、利用可能なキューはこの時点で BATCH と I の二つですね。また、RUNNING 表示に出てくる A/B/C のそれぞれの数字は、各キューで

A : 実行されているジョブの数

B : 一人のユーザが同時に走らせることが出来るジョブの数

C : 同時に実行可能なジョブの数

をそれぞれ示しています。SPARCcenter2000 に限って言えば、このマシンは 4 つの CPU を同時に並行動作させられますから、同時に 4 つ以下の数のジョブが流れるぶんには全く負荷は掛かりません。(もちろんメモリを食いすぎると負荷になりますが。)

このうち BATCH は、デフォルトキューとなっていますから、これ以降の全てのコマンドでキュー指定をしなかった場合は、この BATCH という名前のキューが対象となります。(どのキューがデフォルトキューかという情報は表示されません。)

7.2 バッチジョブの投入

バッチジョブの投入には qsub コマンドを利用します。

```
% qsub [ -q QUEUE ] [ SHELLSCRIPT ]
```

です。QUEUE にはキュー名を、SHELLSCRIPT には、実行したいコマンド列を自分の login shell の文法で書かれたファイルの名前を書きます。

デフォルトキューで実行する場合は-q QUEUE を外してください。スクリプトが簡単に済むなら省略が可能です。その場合は標準入力にコマンド列を書くように要求してきますから、

```
% qsub [ -q QUEUE ]
```

だけで実行し、そこあとにずらずらとコマンド列を書くという形になります。

例として bb というファイルをデフォルトキューで実行します。以下の例を見て判るように、今回のスクリプトは非常に単純で、単に自分で作った resolv というプログラムをホームディレクトリのすぐ下の projectX という名前のサブディレクトリに置いていて、それにパラメタを与えて実行するだけです。date コマンドを前後に入れて時間を表示させるようにしています。

```
cc2000(360)% cat bb
cd projectX
date
resolv 100 x10 y20 z30
date
cc2000(361)% qsub bb
Request 16.cc2000 submitted to queue: BATCH.
cc2000(362)%
```

このように、16 番と言うエントリ番号を貰いました。

7.2.1 ジョブの始まりと終りにお知らせを貰う

自分が投入したジョブの前に、誰かが非常に時間の掛かるジョブを同じキューに投入していたりすると、自分のジョブはいつ実行されるか判りません。また、いつ終わるのかも判りません。

ジョブを投入するときに -mb , -me オプションを付けておけば、それぞれジョブが始まったときと終わったときに mail をくれます。両方のオプションを付けた例を示しておきます。

```
cc2000(402)% qsub -mb -me bb
Request 24.cc2000 submitted to queue: BATCH.
cc2000(403)%
```

こうしておくと、Subject が 24.cc2000 beginning. などというメイルをお知らせに貰う事が出来ます。

7.3 ジョブの標準出力

ジョブの結果がどのように残ってプログラマに与えられるのかはプログラム次第ですが、普通に shell script を実行したときに発生する標準出力と標準エラー出力は自動的にファイルとして生成され、残されます。投入したジョブが終わると、ジョブを投入したときのスクリプトファイルの名前の後に ".o" とエントリ番号を付けたファイルが標準出力の内容を残します。また、".e" とエントリ番号を付けたファイルが標準エラー出力の内容を残しています。もしもスクリプトファイルを利用せずに標準入力からジョブの投入を行った場合は、STDIN に続いて".o"と".e"が付いたファイルが生成されます。

例を示しておきます。まずはエラーファイル。

```
cc2000(425)% more bb.e16
cc2000(426)%
```

空っぽでした。続いて標準出力。

```
cc2000(426)% more bb.o16
Sun Microsystems Inc.   SunOS 5.2       Generic March 1993
1994 年 01 月 26 日 (水) 19 時 43 分 01 秒 JST
Start
End.
1994 年 01 月 26 日 (水) 19 時 43 分 19 秒 JST
cc2000(427)%
```

ああ、うまく行ったようです。安心安心。

これらのファイルは自動的に、qsub でジョブを実行した時のディレクトリに生成されます。

7.3.1 標準出力、エラー出力のファイルを変更したい

もしもファイル名や生成される場所を変えたい場合は、qsub に-o , -e オプションを加えます。

```
% qsub [ -o STDOUT ] [ -e STDERR ] ...
```

です。STDOUT には標準出力の宛先ファイルを、STDERR には標準エラー出力の宛先ファイルを指定します。例として両方のオプションを付けたものを示します。

```
cc2000(431)% qsub -o cc -e ee bb
Request 29.cc2000 submitted to queue: BATCH.
cc2000(432)%
```

標準出力ファイルが cc に、エラーファイルが ee に作成されました。それぞれ cc.o29 や ee.e29 にはならない事に注意してください。

また、エラー出力と標準出力を同じファイルに出したいときは-eo オプションを使います。

```
% qsub -eo [ -o STDOUT ] ...
```

-eo オプションを付けて、なおかつ-o オプションでファイルを指定すると、エラー出力も、標準出力も共に STDOUT に指定したファイルに出力されます。

7.4 ジョブの状態表示

自分がエントリーしたジョブの状態を見るには、qstat コマンドを利用します。

```
% qstat { ENTRY | QUEUE }
```

です。ENTRY には上記のエントリ番号がはいります。QUEUE にはキュー名が入ります。例として先ほどの 16 番の状態を見てみます。

```
cc2000(362)% qstat 16
============================================================
NQS Version: 2.3  BATCH DEVICE PIPE REQUESTS on cc2000
============================================================
REQUEST       NAME            OWNER       QUEUE       PRI NICE CPU     STATE
   16.cc2000  bb              yasuda      BATCH       31   20  36000   RUNNING
cc2000(363)%
```

最後の STATE が RUNNING ですから、無事に今処理されている最中という事です。

ところで他の全てのジョブの状態を知るには qstat -a とします。

```
% qstat -a [ QUEUE ]
```

です。QUEUE にはキュー名が入ります。省略すると全てのキューの情報が表示されます。例を示しておきます。

```
cc2000(365)% qstat -a
============================================================
NQS Version: 2.3  BATCH DEVICE PIPE REQUESTS on cc2000
============================================================
REQUEST       NAME            OWNER       QUEUE       PRI NICE CPU     STATE
   17.cc2000  bb              yasuda      BATCH       31   20  36000   RUNNING
   18.cc2000  bb              yasuda      BATCH       31   20  36000   QUEUED
   28.cc2000  bb              yasuda      I           31   20  36000   RUNNING
cc2000(366)%
```

現在 17 番が BATCH キューで実行中で、18 番は待ち状態にあるということが判ります。また、28 番が I キューで実行中です。

7.5 流れているジョブの中身を確認する

qcat コマンドで、キューに入っているジョブの内容などの確認が出来ます。

7.5.1 ジョブの記述を確認する

既に投入したジョブの記述を確認するには qcat コマンドを利用します。

```
% qcat ENTRY
```

です。ENTRY には上記のエントリ番号がはいります。例として先ほどの 16 番の中身を見てみます。

```
cc2000(363)% qcat 16
>>> Input file  is /usr/spool/nqs/private/root/data/+++++F/++++F+++++0+++.
#
date
cd projectX
resolv 100 x10 y20 z30
date
cc2000(364)%
```

という表示が出てきました。勿論 qcat コマンドは現在実行中、もしくは実行待ちのジョブに対してしか効きません。既に実行が終ってしまったジョブについてはどうにもなりません。

7.5.2 流れているジョブの途中経過を確認する

現在実行中のジョブの出力を実行の最中に確認するには qcat -o コマンドを利用します。

```
% qcat -o ENTRY
```

です。ENTRY には上記のエントリ番号がはいります。例として先ほどの 16 番の中身を見てみます。

```
cc2000(364)% qcat -o 16
Sun Microsystems Inc.   SunOS 5.2       Generic March 1993
1994 年 01 月 26 日 (水) 19 時 43 分 01 秒 JST
Start
cc2000(365)%
```

という表示が出てきました。勿論 qcat -o コマンドは現在実行中のジョブに対してしか効きません。既に実行が終ってしまったジョブについては出力ファイルを見ることになります。

非常に長いジョブを実行する場合は処理の切れ目などで時々標準出力にメッセージを出すようにプログラムを書いておくと qcat -o 機能をうまく利用できて便利でしょう。

なお、-o オプションの代わりに -e オプションを与えることによって (もしあれば) エラー出力の内容を実行途中に確認できます。

7.6 ジョブの実行を保留する

待ち状態にあるジョブの実行を一時保留するには qhold コマンドを利用します。

```
% qhold ENTRY
```

です。ENTRY には上記のエントリ番号が入ります。以下に例を示します。

```
cc2000(459)% qhold 33
Request 33 has been held.
cc2000(460)%
```

hold されたジョブを再び解放するには、qrls コマンドを利用します。

```
% qrls ENTRY
```

です。ENTRY には上記のエントリ番号が入ります。以下に例を示します。

```
cc2000(464)% qrls 33
Request 33 has been released.
cc2000(465)%
```

qhold は、現在実行中のジョブには適用出来ません。qstat -a でチェックしたときに、QUEUED になっているものだけです。以下に qstat -a の例を示します。

一番右の STATE の列を見てください。RUNNING が現在実行中で、QUEUED が現在実行待ちのものです。31 番は HOLDING になっていますね。これは先ほど実行した qhold のせいで、実行が保留されているという事です。

```
cc2000(462)% qstat -a
============================================================
NQS Version: 2.3  BATCH DEVICE PIPE REQUESTS on cc2000
============================================================
REQUEST       NAME            OWNER       QUEUE       PRI NICE CPU     STATE
   32.cc2000  bb              yasuda      BATCH       31   20  36000   RUNNING
   33.cc2000  bb              yasuda      BATCH       31   20  36000   QUEUED
   31.cc2000  bb              yasuda      BATCH       31   20  36000   HOLDING
cc2000(463)%
```

7.7 ジョブの実行を停止、削除する

ジョブの実行を停止するには qdel コマンドを利用します。

```
% qdel [ -SIGNAL ] ENTRY
```

です。現在実行中のジョブの実行を停止させるには、SIGNAL に 9 を入れると良いでしょう。待ち状態のジョブの実行を取り消すだけなら、-SIGNAL は省略可能です。ENTRY には上記のエントリ番号が入ります。以下に例を示します。

```
cc2000(561)% qdel -9 54
Request 54 is running, and has been signalled.
cc2000(562)%
```

7.8 もっと詳しいキューの情報を調べる

最初に qstat -b コマンドによってキューの情報を調べる例を挙げましたが、もっと詳しく、それぞれのキューに割り当てられている最大メモリ量、最大 CPU タイムなどを調べるには qstat コマンドに-bl オプションを与えます。

```
% qstat -bl [QUEUE]
```

です。QUEUE にはキュー名が入ります。省略すると全てのキューの情報が表示されます。例を示しておきます。

```
cc2000(562)% qstat -bl BATCH
=====================================================
NQS version: 2.3        BATCH QUEUE: BATCH.cc2000       status: AVAILBL
=====================================================
                                        Priority: 0
ENTRIES:
        Total: 0        Running: 0
        Queued: 0       Held: 0         Transiting: 0

COMPLEX MEMBERSHIP:

RESOURCES:
  Per-proc core file size limit= 1 Mbytes <DEFAULT>
  Per-process data size limit  = 10 Mbytes
  Per-proc perm file size limit= 10 Mbytes
  Per-proc execution nice value= 0 <DEFAULT>
  Per-process stack size limit = 10 Mbytes
  Per-process CPU time limit   = 300.0
  Per-process working set limit= 10 Mbytes

ACCESS
  Unrestricted access
```

上記の Per-proc data size limit などが一つのジョブ当たりの最大メモリ量です。Per-process CPU time limit などが一つのジョブ当たりの最大 CPU タイムとなります。これらの制限を越えると、その時点でジョブは強制的に終了させられます。

7.9 エラーメッセージ

いつもの通りにジョブが動くはずのところが、以下のようなエラーが出てくる時があります。

7.9.1 あなたの所在地はどこですか？

これは tty 情報が取れない環境で biff を起動したときに表示されるエラーです。biff コマンドが $HOME/.login ファイルなどに含まれているのではありませんか？対処方法はありません。無視するか biff コマンドを使わないようにするしかありません。

7.9.2 警告: tty にアクセスできないため、このシェルでジョブ制御はできません ...

```
Warning: no access to tty; thus no job control in this shell...
```
もしくは
```
Warning: no access to tty (Bad file number).
Thus no job control in this shell.
```

これは tty 情報が取れない環境で `csh` もしくは `tcsh` を起動したときに表示されるエラーです。対処方法はありません。無視してください。

7.10 マニュアルなど

cc2000 にインストールされている NQS に関するドキュメントとしては cc2000:/NF/local/Solaris2J/lib/nqs 以下に troff の mm マクロ形式と、それを PostScript 形式に落したものが用意してあります。

また、一般的なオンラインマニュアルがインストールされているはずですので、`man -k nqs` などとして探せば大抵のコマンドの詳細情報が得られます。

付録 A

情報処理教室利用要項

１．利用目的

次の目的に利用できます。

1) 授業

2) 研究および自習

２．利用資格

次の者が利用できます。

1) 学部学生および大学院学生

2) 専任教員および職員

3) その他、特に所管長が認めた者

３．利用形態

授業利用を優先とし、次の形態で運用します。

1) 授業利用

2) 自由利用

４．休止日

1) 日曜・祝日

2) 本学創立記念日（５月４日）

3) 夏期一斉休業期間（８月１０日～１９日）

4) 年末年始（１２月２６日～１月６日）

5) その他、所管長が必要と認めた日

５．利用時間

次の時間を利用時間とします。

1) 月曜日から金曜日　　８：４５～２０：００

2) 土曜日　　８：４５～１７：００

ただし、大学休業期間中の利用については、必要に応じその都度定めます。

なお、学生が利用時間を超えて利用する場合は、監督・指導にあたる教職員が事前に教室所在の所管長に申し出て、所定の手続きを行い、許可を受けてから利用してください。

６．入室方法

入室は各自の学生証を扉横のカード読取機に通し開錠の上、入室してください。扉を開放状態で放置すると警報が鳴りますので、入室後は必ず扉を閉めてください。

※ C3 情報処理教室および 52 情報処理教室については５、６の条件が少し異なります。C3 情報処理教室利用の際は計算機科学研究所事務室が配布している利用要項を参考にしてください。52 情報処理教室利用の際は経営学部事務室が配布している利用要項を参考にしてください。これらの利用要項は各情報処理教室前の掲示版にも掲示されています。

７．注意事項および利用心得

1) 最終利用者は退出時に次のことを厳守してください。

・窓等の戸締り

・室内の消灯

2) 教室内のマニュアルおよび備品の持ち出しを禁止します。

3) 教室内の飲食および喫煙を禁止します。

31 情報処理教室、51 情報処理教室、52 情報処理教室、C1 情報処理教室についてはパーソナルコンピュータ利用のため、以下の事項に注意してください。

８．ソフトウェアの利用および運用

1) 教材として使用するソフトウェア（ＰＤＳを含む）は、ハードディスクに格納し利用提供します。

2) ハードディスクに格納するソフトウェアは、登録制とします。

3) ソフトウェアの登録を希望する場合は、所定の申請書にて計算機センターに申請してください。

4) ハードディスクへの格納は教室設置機器の全台に行います。

5) ハードディスク内の整理を年数回計算機センターが行います。

９．個人データ等の管理

1) データや作業結果は、個人がフロッピーディスクに格納し所持保管するものとします

10．ソフトウェアの取り扱いおよび遵守事項

各種ソフトウェアの著作権等法的保護およびコンピュータ・ウイルス等の犯罪被害防止のため、次の事項を遵守してください。

1) 学生および大学院生による個人所有の市販ソフトウェア, PDS, FreeWare 等の持ち込み使用を禁止します。

2) 教員が未登録のソフトウェアを研究または授業の教材として利用する時は、原則としてフロッピーディスクで利用することとします。

ただし、不正にコピーされたもの等、正規以外のソフトウェアの持ち込み使用を禁止します。

3) ハードディスク内のソフトウェアの複製および持ち出しを禁止します。

4) ハードディスク内にデータ、作業結果および教員の持ち込みソフトウェアの放置を禁止します。

5) その他ソフトウェアの改造等、著作権法等に抵触する行為を禁止します。

〔Ｈ４．４制定〕

付録 B

著作権法（抜粋）

公布　昭和四五・五・六（法四八）
施行　昭和四六・一・一（附則）
改正　昭和五三法四九、昭和五六法四五、昭和五八法七八、昭和五九法二三・四六、昭和六〇法六二、昭和六一法六四
出典　六法全書　昭和六十三年版 2　 3932〜3939 ページ
(株) 有斐閣　昭和六十三年二月二十五日発行　　ISBN 4-641-00488-9

※これ以降の記述の最新性は保証致しません。※
※タイプミスなどの誤りが含まれている可能性もある事を予め御了承下さい※

第一章　総則

第一節　通則

第一条（目的）
この法律は、著作物並びに実演、レコード、放送及び有線放送に関し著作者の権利及びこれに隣接する権利を定め、これらの文化的所産の公正な利用に留意しつつ、著作者等の権利の保護を図り、もつて文化の発展に寄与することを目的とする。

第二条（定義）
（１）　この法律において、次の各号に掲げる用語の意義は、当該各号に定めるところによる。
一　著作物　思想又は感情を創作的に表現したものであつて、文芸、学術、美術又は音楽の範囲に属するものをいう。
二　著作者　著作物を創作する者をいう。
三　実演　著作物を、演劇的に演じ、舞い、演奏し、歌い、口演し、朗詠し、又はその他の方法により演ずること（これらに類する行為で、著作物を演じないが芸能的な性質を有するものを含む。）をいう。
四　実演家　俳優、舞踊家、演奏家、歌手その他実演を行う者及び実演を指揮し、又は演出する者をいう。
五　レコード　蓄音機用音盤、録音テープその他の物に音を固定したもの（音をもつぱら影像とともに再生することを目的とするものを除く。）をいう。
六　レコード製作者　レコードに固定されている音を最初に固定した者をいう。
七　商業用レコード　市販の目的をもつて製作されるレコードの複製物をいう。
八　放送　公衆によつて直接受信されることを目的として無線通信の送信を行なうことをいう。
九　放送事業者　放送を業として行なう者をいう。
九の二　有線放送　有線送信のうち、公衆によつて同一の内容の送信が同時に受信されることを目的として行うものをいう。
九の三　有線放送事業者　有線放送を業として行う者をいう。
十　映画製作者　映画の著作物の製作に発意と責任を有する者をいう。
十の二　プログラム　電子計算機を機能させて一の結果を得ることができるようにこれに対する指令を組み合わせたものとして表現したものをいう。

十の三　データベース　論文、数値、図形その他の情報の集合物であつて、それらの情報を電子計算機を用いて検索することができるように体系的に構成したものをいう。
十一　二次的著作物　著作物を翻訳し、編曲し、若しくは変形し、又は脚色し、映画化し、その他翻案することにより創作した著作物をいう。
十二　共同著作物　二人以上の者が共同して創作した著作物であつて、その各人の寄与を分離して個別的に利用することができないものをいう。
十三　録音　音を物に固定し、又はその固定物を増製することをいう。
十四　録画　影像を連続して物に固定し、又はその固定物を増製することをいう。
十五　複製　印刷、写真、複写、録音、録画その他の方法により有形的に再製することをいい、次に掲げるものについては、それぞれ次に掲げる行為を含むものとする。
　イ　脚本その他これに類する演劇用の著作物　当該著作物の上演、放送又は有線放送を録音し、又は録画すること。
　ロ　建築の著作物　建築に関する図面に従つて建築物を完成すること。十六　上演　演奏（歌唱を含む。以下同じ。）以外の方法により著作物を演ずることをいう。
十七　有線送信　公衆によつて直接受信されることを目的として有線電気通信の送信（有線電気通信設備で、その一の部分の設置の場所が他の部分の設置の場所と同一の構内（その構内が二以上の者の占有に属している場合には、同一の者の占有に属する区域内）にあるものによる送信を除く。）を行うことをいう。
十八　口述　朗読その他の方法により著作物を口頭で伝達すること（実演に該当するものを除く。）をいう。
十九　上映　著作物を映写幕その他の物に映写することをいい、これに伴って映画の著作物において固定されている音を再生することを含むものとする。
二十　頒布　有償であるか又は無償であるかを問わず、複製物を公衆に譲渡し、又は貸与することをいい、映画の著作物又は映画の著作物において複製されている著作物にあつては、これらの著作物を公衆に提示することを目的として当該映画の著作物の複製物を譲渡し、又は貸与することを含むものとする。
二十一　国内　この法律の施行地をいう。
（２）　この法律にいう「美術の著作物」には、美術工芸品を含むものとする。
（３）　この法律にいう「映画の著作物」には、映画の効果に類似する視覚的又は視聴覚的効果を生じさせる方法で表現され、かつ、物に固定されている著作物を含むものとする。
（４）　この法律にいう「写真の著作物」には、写真の製作方法に類似する方法を用いて表現される著作物を含むものとする。
（５）　この法律にいう「公衆」には、特定かつ多数の者を含むものとする。
（６）　この法律にいう「法人」には、法人格を有しない社団又は財団で代表者又は管理人の定めがあるものを含むものとする。
（７）　この法律において、「上演」、「演奏」又は「口述」には、著作物の上演、演奏又は口述で録音され、又は録画されたものを再生すること（放送、有線送信又は上映に該当するものを除く。）を含み、「上演」、「演奏」、「口述」又は「上映」には、著作物の上演、演奏、口述又は上映を電気通信設備を用いて伝達すること（放送又は有線送信に該当するものを除く。）を含むものとする。
（８）　この法律にいう「貸与」には、いずれの名義又は方法をもつてするかを問わず、これと同様の使用の権原を取得させる行為を含むものとする。
（９）　この法律において、第一項第八号、第九号の二若しくは第十三号から第二十号まで又は前二項に掲げる用語については、それぞれこれらを動詞の語幹として用いる場合を含むものとする。

※ 略 ※

第二章　著作者の権利

第一節　著作物

第十条（著作物の例示）
（１）　この法律にいう著作物を例示すると、おおむね次のとうりである。
一　小説、脚本、論文、講演その他の言語の著作物
二　音楽の著作物
三　舞踊又は無言劇の著作物
四　絵画、版画、彫刻その他の美術の著作物
五　建築の著作物
六　地図又は学術的な性質を有する図面、図表、模型その他の図形の著作物
七　映画の著作物
八　写真の著作物
九　プログラムの著作物
（２）　事実の伝達にすぎない雑報及び時事の報道は、前項第一号に掲げる著作物に該当しない。
（３）　第一項第九号に掲げる著作物に対するこの法律による保護は、その著作物を作成するために用いるプログラム

言語、規約及び解法に及ばない。この場合において、これらの用語の意義は、次の各号に定めるところによる。
一　プログラム言語　プログラムを表現する手段としての文字その他の記号及びその体系をいう。
二　規約　特定のプログラムにおける前号のプログラム言語の用法についての特別の約束をいう。
三　解法　プログラムにおける電子計算機に対する指令の組合せの方法をいう。

第十一条（二次的著作物）
二次的著作物に対するこの法律による保護は、その原著作物の著作者の権利に影響を及ぼさない。

第十二条（編集著作物）
（１）　編集物（データベースに該当するものを除く。以下同じ。）でその素材の選択又は配列によって創作性を有するものは、著作物として保護する。
（２）　前項の規定は、同項の編集物の部分を構成する著作物の著作者の権利に影響を及ぼさない。
第十二条の二（データベースの著作物）
（１）　データベースでその情報の選択又は体系的な構成によつて創作性を有するものは、著作物として保護する。
（２）　前項の規定は、同項のデータベースの部分を構成する著作物の著作者の権利に影響を及ぼさない。

第十三条（権利の目的とならない著作物）
次の各号のいずれかに該当する著作物は、この章の規定による権利の目的となることができない。
一　憲法その他の法令
二　国又は地方公共団体の機関が発する告示、訓令、通達その他これらに類するもの
三　裁判所の判決、決定、命令及び審判並びに行政庁の裁決及び決定で裁判に準ずる手続きにより行なわれるもの
四　前三号に掲げるものの翻訳物及び編集物で、国又は地方公共団体の機関が作成するもの

第二節　著作者

第十四条（著作者の推定）
著作物の原作品に、又は著作物の公衆への提供若しくは提示の際に、その氏名若しくは名称（以下「実名」という。）又はその雅号、筆名、略称その他実名に代えて用いられるもの（以下「変名」という。）として周知のものが著作者名として通常の方法により表示されている者は、その著作物の著作者と推定する。

第十五条（職務上作成する著作物の著作者）
（１）　法人その他使用者（以下この条において「法人等」という。）の発意に基づきその法人等の業務に従事する者が職務上作成する著作物（プログラムの著作物を除く。）で、その法人等が自己の著作の名義の下に公表するものの著作者は、その作成の時における契約、勤務規則その他に別段の定めがない限り、その法人等とする。
（２）　法人等の発意に基づきその法人等の業務に従事する者が職務上作成するプログラムの著作物の著作者は、その作成の時における契約、勤務規則その他に別段の定めがない限り、その法人等とする。

第十六条（映画の著作物の著作者）
映画の著作物の著作者は、その映画の著作物において翻案され、又は複製された小説、脚本、音楽その他の著作物の著作者を除き、制作、監督、演出、撮影、美術等を担当してその映画の著作物の全体的形成に創作的に寄与した者とする。ただし、前条の規定の適用がある場合は、この限りでない。

第三節　権利の内容

第一款　総則

第十七条（著作者の権利）
（１）　著作者は、次条第一項、第十九条第一項及び第二十条第一項に規定する権利（以下「著作者人格権」という。）並びに第二十一条から第二十八条までに規定する権利（以下「著作権」という。）を享有する。
（２）　著作者人格権及び著作権の享有には、いかなる方式の履行をも要しない。

第二款　著作者人格権

第十八条（公表権）
（１）　著作者は、その著作物でまだ公表されていないもの（その同意を得ないで公表された著作物を含む。次項において同じ。）を公衆に提供し、又は提示する権利を有する。当該著作物を原著作物とする二次的著作物についても、同様とする。
（２）　著作者は、次の各号に掲げる場合には、当該各号に掲げる行為について同意したものと推定する。
一　その著作物でまだ公表されていないものの著作権を譲渡した場合　当該著作物をその著作権の行使により公衆に

提供し、又は提示すること。
二　その美術の著作物又は写真の著作物でまだ公表されていないものの原作品を譲渡した場合これらの著作物をその原作品による展示の方法で公衆に提示すること。
三　第二十九条の規定によりその映画の著作物の著作権が映画製作者に帰属した場合　当該著作物をその著作権の行使により公衆に提供し、又は提示すること。

第十九条（氏名表示権）
（１）　著作者は、その著作物の原作品に、又はその著作物の公衆への提供若しくは提示に際し、その実名若しくは変名を著作者名として表示し、又は著作者名を表示しないこととする権利を有する。その著作物を原著作物とする二次的著作物の公衆への提供又は提示に際しての原著作物の著作者名の表示についても、同様とする。
（２）　著作物を利用する者は、その著作者の別段の意思表示がない限り、その著作物につきすでに著作者が表示しているところに従って著作者名を表示することができる。
（３）　著作者名の表示は、著作物の利用の目的及び態様に照らし著作者が創作者であることを主張する利益を害するおそれがないと認められるときは、公正な慣行に反しない限り、省略することができる。

第二十条（同一性保持権）
（１）　著作者は、その著作物及びその題号の同一性を保持する権利を有し、その意に反してこれらの変更、切除その他の改変を受けないものとする。
（２）　前項の規定は、次の各号のいずれかに該当する改変については、適用しない。
一　第三十三条第一項（同条第四項において準用する場合を含む。）又は第三十四条第一項の規定により著作物を利用する場合における用字又は用語の変更その他の改変で、学校教育の目的上やむを得ないと認められるもの
二　建築物の増築、改築、修繕又は模様替えによる改変
三　特定の電子計算機においては利用し得ないプログラムの著作物を当該電子計算機において利用し得るようにするため、又はプログラムの著作物を電子計算機においてより効果的に利用し得るようにするために必要な改変
四　前三号に掲げるもののほか、著作物の性質並びにその利用の目的及び態様に照らしやむを得ないと認められる改変

※ 略 ※

第二一条（複製権）
著作者は、その著作物を複製する権利を専有する。

※ 略 ※

第五款　著作権の制限

第三〇条（私的使用のための複製）
著作権の目的となつている著作物（以下この款において単に「著作物」という。）は、個人的に又は家庭内その他これに準ずる限られた範囲内において使用するこたを目的とする場合には、公衆の使用に供することを目的として設置されている自動複製機器（複製の機能を有し、これに関する装置の全部又は主要な部分が自動化されている機器をいう。）を用いて複製するときを除き、その使用する者が複製することができる。

第三一条（図書館等における複製）
図書、記録その他の資料を公衆の利用に供することを目的とする図書館その他の施設で政令で定めるもの（以下この条において「図書館等」という。）においては、次に掲げる場合には、その営利を目的としない事業として、図書館等の図書、記録その他の資料（以下この条において「図書館資料」という。）を用いて著作物を複製することができる。
一　図書館等の利用者の求めに応じ、その調査研究の用に供するために、公表された著作物の一部分（発行後相当期間を経過した定期刊行物に掲載された個個の著作物にあつては、その全部）の複製物を一人につき一部提供する場合
二　図書館資料保存のため必要がある場合
三　他の図書館等の求めに応じ、絶版その他これに準ずる理由により一般に入手することが困難な図書館資料の複製物を提供する場合

第三二条（引用）
（１）　公表された著作物は、引用して利用することができる。この場合において、その引用は、公正な慣行に合致するものであり、かつ、報道、批評、研究その他の引用の目的上正当な範囲内で行なわれるものでなければならない。
（２）　国又は地方公共団体の機関が一般に周知させることを目的として作成し、その著作の名義の下に公表する広報資料、調査統計資料、報告書その他これらに類する著作物は、説明の材料として新聞紙、雑誌その他の刊行物に転載することができる。ただし、これを禁止する旨の表示がある場合は、この限りでない。

※ 略 ※

第四七条の二（プログラムの著作物の複製物の所有者による複製等）
（１）　プログラムの著作物の複製物の所有者は、自ら当該著作物を電子計算機において利用するために必要と認められる限度において、当該著作物の複製又は翻案（これにより創作した二次的著作物の複製を含む。）をすることができる。ただし、当該利用に係る複製物の使用につき、第百十三条第二項の規定が適用される場合は、この限りでない。
（２）　前項の複製物の所有者が当該複製物（同項の規定により作成された複製物を含む。）のいずれかについて滅失以外の事由により所有権を有しなくなつた後には、その者は、当該著作権者の別段の意思表示がない限り、その他の複製物を保存してはならない。

第四八条（出所の明示）
（１）　次の各号に掲げる場合には、当該各号に規定する著作物の出所を、その複製又は利用の態様に応じ合理的と認められる方法及び程度により、明示しなければならない。
一　第三十二条、第三十三条第一項（同条第四項において準用する場合を含む。）、第三十七条、第四十二条又は第四十七条の規定により著作物を複製する場合
二　第三十四条第一項、第三十九条第一項又は第四十条第一項若しくは第二項の規定により著作物を利用する場合
三　第三十二条の規定により著作物を複製以外の方法により利用する場合又は第三十五条、第三十六条第一項、第三十八条第一項、第四十一条若しくは第四十六条の規定により著作物を利用する場合において、その出所を明示する慣行があるとき。
（２）　前項の出所の明示に当たつては、これに伴い著作者名が明らかになる場合及び当該著作物が無名のものである場合を除き、当該著作物につき表示されている著作者名を示さなければならない。
（３）　第四十三条の規定により著作物を翻訳し、編曲し、変形し、又は翻案して利用する場合には、第二項の規定の例により、その著作物の出所を明示しなければならない。

※ 略 ※

第四節　保護期間

第五一条（保護期間の原則）
（１）　著作権の存続期間は、著作物の創作の時に始まる。
（２）　著作権は、この節に別段の定めがある場合を除き、著作者の死後（共同著作物にあつては、最終に死亡した著作者の死後。次条第一項において同じ。）五十年を経過するまでの間、存続する。

※ 略 ※

（２）　プログラムの著作物の著作権を侵害する行為によつて作成された複製物（当該複製物の所有者によつて第四十七条の二第一項の規定により作成された複製物並びに前項第一号の輸入に係るプログラムの著作物の複製物及び当該複製物の所有者によつて同条第一項の規定により作成された複製物を含む。）を業務上電子計算機において使用する行為は、これらの複製物を使用する権原を取得した時に情を知つていた場合に限り、当該著作権を侵害する行為とみなす。
（３）　著作者の名誉又は声望を害する方法によりその著作物を利用する行為は、その著作者人格権を侵害する行為とみなす。

※ 以降略 ※

付録 C

参考文献

ここにあげる文献にある記述が全て cc 環境で適用出来るとは限らないことに注意してください。

—— **Unix 全般、シェル、コマンドなどについて** ——

たのしい UNIX -UNIX への招待- 坂本　文著：アスキー出版局刊
月刊雑誌 UNIX Magazine の連載を集成した UNIX 初心者向けの入門書。

続 たのしい UNIX -シェルへの招待- 坂本　文著：アスキー出版局刊
上の一冊の続編。今度はシェルについて解説してくれる。

実用 UNIX ハンドブック 舟本　奨著：ナツメ社
UNIX コマンドの簡単なリファレンス。

UNIX C SHELL フィールドガイド G・アンダーソン、P・アンダーソン著：落水　浩一郎、大木　敦雄訳：パーソナルメディア刊
csh のほぼ完全なガイド。

UNIX step++ シェルプログラミングのコツ 西沼　行博著：マグロウヒル刊
残念ながら csh ではなく、sh についての説明が主体。記述も少々古いがシェルを使いこなしたい人には便利。

—— **Emacs について** ——

GNU Emacs Debra Cameron and Bill Rosenblatt 著：ハイパーウェア監訳：ソフトバンク株式会社刊
Emacs のほぼ完全なガイド。NutsShell（ナッツ（どんぐり）のカラ？）シリーズと呼ばれる非常に詳細な Unix 関係のドキュメントのシリーズの一冊。

入門 NEmacs 大木　敦雄著：アスキー出版刊
Emacs だけでなく、EGG, MHE, GNUS などについても説明してくれている。

—— **TeX について** ——

楽々LaTeX 野寺　隆志著：共立出版刊
LaTeX を用いた TeX の入門書。初心者には最適。

LaTeX 美文書作成入門 奥村　晴彦著：技術評論社刊

TeX についていろいろ丁寧に教えてくれる。

日本語 LaTeX 定番スタイル集 No.1, No.2 鷲谷　好輝著：インプレス刊

京都産業大学でキャンパスライセンスを取得している LaTeX のスタイルファイルの使い方解説書。きれいなスタイルファイルの見本としてもよい。