生活社会科学科

基礎演習 III テキスト

三木　邦弘

平成 8 年 4 月 18 日

目 次

1　はじめに

平成8年度の私の基礎演習では、ワークステーションを使ってネットワークの利用やCを使用してプログラミングを学びます。

今年で私の基礎演習も5年目になります。これまでの内容を振り返って見ますと、

1. BASICを使用したプログラミング
2. Cを使用したプログラミング
3. Cを使用したプログラミングと電子メール、ニュース、WWW、LaTeX
4. Cを使用したプログラミングと電子メール、ニュース、WWW、HTML

となっています。今年は昨年度と内容的には同じですが、これまでパソコン上でプログラミングを行ってきたのを、ワークステーション上で行うようにします。

1.1　ワークステーションとは

まずワークステーションとは何か？となりますが、ちょっと強力なパソコンと思えばだいたい間違いはありません。以前はかなり処理能力などに差がありましたが、現在では能力的にはワークステーションもパソコンも同等なものが多いようです。ただ個々バラバラに使われることを想定されて作られてきたパソコンと、ネットワークを構成して利用することを想定されたワークステーションでは、特にネットワークへの対応の部分で大きな違いがあります。

ワークステーションでは、ネットワークに接続するにはケーブルを追加するだけです。基本ソフトウェア(UNIX)はネットワークに対応しているので、設定を行うだけで他のワークステーションのディスクを利用したりすることが簡単にできます。パソコンも次第にネットワークを考慮したものに変わりつつありますが、まだ面倒な点が多いのも事実です。近い将来パソコンとワークステーションの目立った違いは無くなり、価格の高いワークステーションは消えていくと思われます。しかし、現在、ネットワークにつながったコンピュータの利用を学ぶには、ワークステーションを利用するのが一番の様です。

1.2　MS–DOSやWindowsとUNIXの違い

ワークステーションでは基本ソフトウェアとしてUNIXが使われています。これはパソコンで使われているMS–DOSやWindowsと異なり、マルチタスク、マルチユーザーに対応しています。マルチタスクとは同時に複数のプログラムが実行できる事です。またマルチユーザーは同時に複数の利用者がシステムを利用できると言うことです。

マルチユーザーのシステムでは必ず、今使用しようとしている利用者は誰なのか明らかにする必要が生じます。またシステムの共有できる部分は勿論共有して構いませんが、個人的なデータを他の人が勝手に見たり変更できては困ります。そのためにシステムの利用開始時に利用者を確認する手続きと、全てのファイル等に誰にどのようなアクセスを許すかなどの設定をしておく必要があります。その点個人による個人利用しか考慮していないMS–DOSは気楽で、そのような手続きや設定は全く必要ありません。そのかわり電源を入れてくれた人に誰でもシッポを振ってしまいます。

逆にMS–DOSとUNIXには同じ所はないのか?と言えばかなりあります。UNIXの方が歴史が古く、MS–DOSは最近まで新しい版に変わる度にUNIXのコマンドや機能を取り込んできました。どちらかと言えばこの共通点ばかり勉強したほうが、大抵の基本ソフトに共通する点を学ぶことになり良いかもしれません。ですがここでは時間の都合もあり、パソコンの事は身近なパソコンで自分で勉強すると言うことにして特に取り上げません。

1.3 プログラミングについて

プログラムを作成することをプログラミングと言います。プログラミングに関しては、3年次の「プログラミング演習」とどうしても内容が重複するなどの問題が出ています。そこでプログラミングの例題をネットワークと関連させてみようかと考えています。

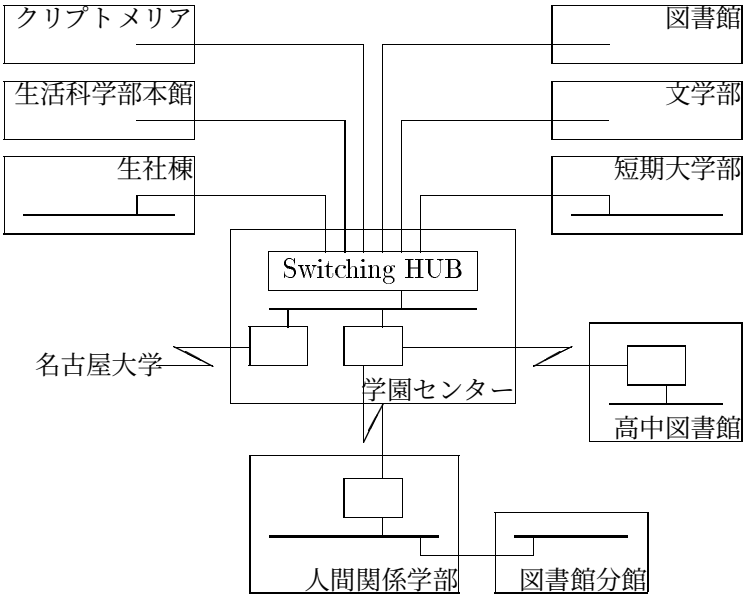
Cを利用してプログラミングを行うのならば、本来ならば何かCの解説の本を買って頂いてそれを元に演習を進めたいのですが、なかなか適当な本がありません。この演習ではCを使用しますが、文法的な事項は最低限に限定します。そのため通常の本では一通りCについて述べなければならないためか、余分な説明が多過ぎます。と言うわけでせっせとテキストを作成したのですが、もっと勉強されたい方は、Cに関する本は非常に多く出版されていますので、本屋で適当な本を購入して勉強して下さい。

1.4 ネットワークの時代

高度情報化社会と言われて随分時が立ちますが、なかなか普通の人には縁がありませんでした。確かにワープロ、ファックス等は普及しました。しかしTV、ラジオ、新聞等のマスメディアは相変わらず送り手から受手のへの一方通行です。最近になってパソコン通信という形のメディアが使われるようになってきました。当初は一部のマニアばかり(野郎ばかり)だったのですが、次第に一般の人の利用が増えて、企業などでも社内のコミュニケーションの手段の1つとして認められて来ました。

最近では「インターネット」が大流行でTVなどでもよく取り上げられていますが、ネットワークの必ずしもWWWだけではないのでそのあたりが理解してもらえるようなものにしようと思います。

このような時代の流れを受けて椙山女学園でもここ数年学内のネットワークの環境整備に努めてきました。ほぼ全ての教員の部屋までLANの配線が行われています。学生が利用できる端末の絶対数やその性能にまだ問題か残っています。次の図は現在の学園のネットワークを図示したものです。



2　使用上の基本

この演習では学園センターの機器を利用しますが、その利用の際に基本となる点をここで述べます。

2.1　学園センターの利用について

学園センターのG階にある学生情報処理実習フロアーは、

- 平日：9:00 A.M. から 17:00 P.M. まで
- 土曜日：9:00 A.M. から 13:00 P.M. まで
- 日曜、祝祭日：お休み

の間利用することができます。ただし夏休み中は、閉める時間が早くなったり、集中講義や演習中は利用できないことがあります。椙山女学園の関係者は誰でも利用することができます。

ここには、ワープロも設置されていますが、パソコンについて見ると、

- 実習室1：40台。ワークステーションの端末として利用可能。一太郎も利用可能。
- 実習室2：24台。端末や一太郎は利用できません。
- カウンターの前：16台。端末としては利用できますが、ちょっと使い方が異なります。
- その他：4台。端末や一太郎は利用できません。

となっています。どのパソコンでもFM–OASYSと言うワープロソフトと表計算ソフトとして有名なLotus 1-2-3は利用することができます。ここにあるのは富士通製のFMRと言う機種で、大変古いものなので近年では普通であるWindowsは利用することができません。来年度になれば新しい機種も導入されるようです。

演習などの時間内に利用されるときには構いませんが、それ以外の時間に利用する際には、入り口横のカウンターに置いてある利用記録簿に記入をお願いします。またこの部屋の中での飲食は禁止されているので守ってください。

2.2　パソコンの起動と終了

パソコンの起動は、本体の正面左の方にあるpowerと書いたスイッチを押します。作動音がして、しばらくすると次のようなメニューが表示されます。

Program 開発	ＯＡ Program	ファイル変換	システムコマンド	Lotus 紹介	利用終了

起動した後は、メニューの一番左端の「Program 開発」が黄色で他の部分は緑色になっているはずです。この黄色になっているのが現在選択されている項目です。選択されている項目を実行するには、キーボードの 実行 のキーを押します。すると通常は説明の画面になりますので、さらに 実行 を押します。ここでメニューにもどるならば、 ESC を押します。

別の項目を選択するときには、カーソルキー(← や →)などを押します。項目によってはサブメニューが縦方向に表示されます。このときも選択にはカーソルキー(↓ や ↑)などを押します。サブメニューから出たい時には ESC 又は ← や → を押します。

パソコンの利用を止めるときには、必ずメニューに戻って「利用終了」の項目を選択して実行します。すると本体の電源も自動的に切れます。フロッピー等を忘れずに取り出すようにします。最近のパソコンは複雑化しているために、形こそ違いますが必ずこのような終了の手続きが必要であることを憶えておいて下さい。

2.3 フロッピーディスクの扱い

昔のパソコンの標準のフロッピーは、5インチと言われる大きさのものでした。学園センターのパソコンではこれを用います。これに対して現在のパソコンでは3.5インチと言われる大きさのフロッピーを用います。5インチのフロッピーは3.5インチのものに比べて構造が簡単で弱く、扱いには十分注意する必要があります。そこで、この演習で使うフロッピーは入り口のカウンターに箱を置いておきますので、必ずその箱に入れておくようにしてください。

最近のフロッピーディスクは既に初期化されたものが売られていますが、配布したフロッピーは次のような手順で初期化しないと利用することができません。

まずメニューの中の「システムコマンド」を選択し、さらに「初期化」、さらに「フロッピーの初期化」を選択します。すると、

```
ディスクフォーマット　第 3.70 版
ドライブ A: に新しいディスクを挿入してください。
準備ができたらどれかキーを押してください。
```

と表示されますので左側のドライブにフロッピーを入れてから、実行キーを押します。するとトラック00から76まで順次初期化を行なって、

```
11 文字までのボリューム・ラベルを入力してください。
ボリューム・ラベルを省略する時は、リターンキーを押してください。
```

と表示されますので、通常はリターンキー(以下このテキストではこのキーのことを△で示します。)を押します。さらに、

```
別のディスクをフォーマットしますか (Y/N) ?
```

と表示されますので、通常はn△と押します。すると、

```
>どれかキーを押してください。
```

と下の方に表示されますので実行でも押します。さらに最初のメニューに戻るには、ESCを2回押す必要があります。

3 タイプの練習ソフトについて

パソコンの標準的な入力装置はキーボードです。プログラムやデータの入力は基本的にこれを用います。そのためにパソコンを使いこなすには、キーボードで早く入力ができることが必須の技能となります。欧米ではコンピュータができる以前よりタイプライターが存在し、タイピストと呼ばれる専門家以外でも多くの人がこれを利用してきました。日本では最近こそワープロが普及していますが、ほとんどの利用者がキーボードのキーを探しながらキーを押しているような状況です。

理想的にはキーボードを全く見ないで打てるのが良いのですが、キーの押しにくい周辺のキーは確認して押すレベルでも、全く練習していない状況よりはましです。キーボードを全く見ない状況とは、キーの位置を全て憶えている状態ですが、いちいちキーの位置を意識の上で考えるのではありません。条件反射でキーを思うだけで反射的に指が動くようにするのです。

3.1 タイプ練習ソフトの起動方法

タイプの練習ソフトの利用の際には各自フロッピーが必要です。これはこのフロッピーに各自の練習記録が保存されるからです。

タイプの練習ソフトの起動は、「ＯＡ Program」と言うメニューの中の「タイプの練習」を選択して実行します。すると、画面に

```
左側のドライブにフロッピィディスクを入れて下さい。
準備ができたらどれかのキーを押してください。
```

と出ますので、左側のドライブにフロッピーディスクを入れてから、実行キーでも押します。練習プログラムが起動されると表題の画面になりますので、もう一回実行キーでも押しますと、メニューが出てきます。これ以降はやりたい項目の先頭に書いてある番号を入力するだけです。

練習が終わり、練習プログラムを終了すると最初のメニューにもどりますので、続けて別の事(端末としての利用など)も可能です。

3.2 タイプの練習の進め方

やはりピアノが上手とか、運動神経に自信のある方は上達が早いようです。しかし普通のひとならば誰でも練習に応じて上達します。まだ20歳にもならない若さを信じて頑張ってください。

進め方は基本的に最初に表示されるメニューの順番に従って下さい。まず1番の「ポジション練習」でどの指でどのキーを押すのか憶えます。「ポジション練習」のメニューでは、やはり1番から順番にやってください。画面上部に押すべきキーが表示されますので、最初のうちはその下のキーボードの図や、その下のそれを押すべき指の表示を確認しながら、その通り押してください。両手にそれぞれ5本の指が無い人以外は必ずそれに従って下さい。

「ポジション練習」は1回につき60個のキーの練習ができます。終わったところで、まだ足りない人は改行キーをおせば続けて練習ができます。終わりたい人はESCキーを押せばメニューに戻ります。練習を続ける際は、1回終わる度に少しパソコンのディスプレイから目を離して休憩を取りましょう。

「ポジション練習」で1つマスターできた様ならば、「ランダム練習」に同じメニューがありますので、そちらに挑戦して見てください。とりあえず目標を70文字／分以上としてください。目標をクリアーできたら次のキーの位置を「ポジション練習」で憶えて下さい。

「ランダム練習」の全てのメニューで目標をクリアーできた方は、「英単語練習」のメニューに進んで下さい。ここでは「基本英単語練習」から「８０８６アセンブラ練習」まであります。大文字ばかり出てくるものもありますが、そのままキーを押せば済みます。これらに関しては100文字／分以上を目標にして、全てのメニューをクリアーしてください。

さらに「ローマ字練習」もありますので、余裕のある方は挑戦してみて下さい。間違って入力した場合、最初の子音の部分なのか母音の部分なのかわかりにくいので苦労するでしょう。また1文字当たり平均2つのキーを押さなければならないので、記録は「ランダム練習」の約半分になるでしょう。

ここでは目標を速度にします。もちろん「ポジション練習」以外でキーボードを見ながらやってはいけません。反射神経育成のために少々間違えてもどんどんキーを押してください。

メニューの「成績」のところでこれまでの記録や練習時間を見ることができます。と言うことは、ちゃんと練習をしたけれども遅い人と、さぼっているので遅い人との区別ができると言うことです。

この練習ソフトでは文字と数字キーのみを練習の対象としているので、プログラムの入力の際に必要な記号の入力の練習ができない欠点があります。英文にしろ和文にしろ入力の際には最低句読点の入力は必要ですので、改善が望まれます。

4　利用開始と終了

現在この椙山女学園大学にはワークステーションが15台しかありません。ワークステーションも正しい?利用形態では、1人の利用者が1台のワークステーションを使用するのですが、現状では無理です。そこで現在学園センターに設置されているワークステーションcc01にはパソコンが48台端末として接続されていますので、この演習ではこれを使ってワークステーションを利用します。なお生社棟の004室の約20台や414室の8台のパソコンも端末として利用可能ですが、少し操作が異なります。

学園センターの端末となるパソコンは、実習室1にあるFMRが40台と、カウンターの前のワープロの隣に2列に16台並んでいるうちのカウンター寄りの8台です。その一番奥にプリンターと並んでいるのが、cc01と言う名前が付いているワークステーションです。見た目は回りのFMRと変わりませんが、メモリーやディスクの容量、処理速度等はかなり違います。

4.1　利用の開始

まずパソコンで端末プログラムを起動します。そえにはメニューの左側から2番目の「ＯＡ Program」を選択し、さらにサブメニューから「ワークステーションの端末」を選択します。すると、

```
cc01 login:
```

と表示されますので、システムの利用を開始する手続きをします。(出ていない時には△を押してみてください。)「login:」と出ているところで、あなたのユーザー名を入力し、△を押します。次に「passwd:」と出てきますので、あなたのパスワードを入力します。このときキーを押しても画面には何も表示されないので慌てないで下さい。最後にはやはり△を押します。ユーザー名やパスワードに間違いがあると再び「login:」が出ますので、また最初から入れ直して下さい。

loginが成功すると次のような感じのメッセージが出ます。

```
Last login: Tue Apr 16 14:03:25 from bach
SunOS Release 4.1.4_JLE1.1.4 (GENERIC) #2: Sat Apr 13 13:42:48 JST 1996
```

これは前回利用した時刻や基本ソフトの構成などを示しています。もしあなた宛の電子メールが来ているとこの後に、

```
You have mail.
```

等のメッセージが表示される事があります。そして、

```
cc01[miwako]%
```

が表示されると、システムはコマンドを入力しても良い状態になったことが判ります。このようなコマンドの入力を促すメッセージの事をプロンプトと呼びます。

4.2　利用終了

まずシステムの利用を終了する手続きをします。プロンプトの出ているときに、logout△と入力すると利用終了の処理がなされて、さらに端末プログラムも終了して、最初のメニューにもどります。(カウンター前の端末では CTRL と PF20 を同時に押さないとメニューに戻りません。)利用終了時には必ずこれを行わないと、次の利用者に不正に利用される可能性がありますのでご注意下さい。

4.3 パスワードの重要性

ワークステーションは利用者の入力するユーザー名とパスワードで利用者の識別をします。あなたの銀行のカードの暗証番号と同様に他の人に教えたりしないで下さい。また安易な語をパスワードにしていると、見破られる原因になります。世の中にはそんなに悪い人は居ません。しかしごく少数の賢い悪人は、ネットワークと自分のマシンを使って自動的に世界中のシステムを荒らそうと狙っています。だから自分の身の回りには善良な人しかいない、そんなにコンピュータを使える人はいないと安心してはいけません。

銀行のカードならば悪用されてもあなたが泣くだけで済みますが、大学や会社のシステムを利用する際のパスワードが漏れると、システムの全利用者に被害が及ぶことがあります。そしてその責任の大半はあなたが取らなければなりません。社会に出る前にきちっとパスワードが扱えるようになって欲しいと思います。

銀行のカードの暗証番号は、カードがなければ実際は使えないために数字４桁と言う簡単なものになっています。ここのシステムの様に誰でもパスワードさえ合えば利用できるような場合は、簡単に想像がつくようなものや辞書に載っている英単語等は駄目です。住所や家族から想像されるようなもの、誕生日や学籍番号から想像されるようなものが駄目な例です。適当な文字や数字を混ぜて適当に長いものがお勧めです。

4.4 パスワードの変更

最初は全員パスワードが設定されていません。このままではお互いに相手になりすまして利用することが出来てしまいますので、必ずパスワードを設定してください。そのやり方は次のようになります。

1. プロンプトが出ている所に、passwd△と入力する。
2. 「Changing password for ユーザー名 on cc01」と表示され、次に「Old password:」と出るので、現在のパスワードと△を入力する。このとき画面には何も表示されないので慌てないように。以下の入力の際も同様です。
3. 「New password:」と表示されたら、新しいパスワードと△を入力する。
4. 「Retype new password:」と表示されたら、もう一度新しいパスワードと△を入力する。
5. 再びプロンプトが表示されたらパスワードの変更は成功。もし、「Mismatch – password unchanged.」と出た場合は、新しいパスワードの入力間違いですので、もう１回最初からやり直して下さい。

本当は毎月のようにパスワードを変更するべきだと言う意見もありますが、そこまでは必要ないでしょう。ただ変更したが、何に変更したか忘れてしまったような事はないようにお願いします。

4.5 演習問題

1. パソコンで端末ソフトを起動し、loginをし、すぐlogoutを試みよ。
2. 再びloginを行い、パスワードの変更を行え。その後logoutし、すぐloginを試みて、パスワードが正しく変更されていることを確認せよ。
3. loginを行い、プロンプトが表示されたら、letters△と入力せよ。すると画面の上から英単語が降ってくるので、キーボードで入力する。正しく入力できれば得点となり、画面の下に達するまでに正しく入力出来なければ過失となり、３回失うと終了してしまう。最終得点の上位20位までは記録に残る。なお途中で終了したい場合には、CTRLキーを押しながらcキーを押すと「are you sure you want to quit?」と聞いてくるのでy△と入力すれば良い。

5　文書 (テキスト) の編集

現在のコンピュータは本来の数値情報の処理だけでなく、文字、音声、画像、動画等を扱うことができます。処理する数値や文字の入力にはキーボードがよく使われ、他の入力にはそれぞれ専用の装置が使われます。

実際のデータ入力の際には必ずごみや誤りがまぎれ込みます。正しく処理を行うためには、これらを訂正しなければなりません。ここでは数値や文字データの入力や訂正に用いられるエディタの使用法について述べます。

5.1　漢字の入力方法

英字や数字はキーボードから直接入力できますが、我々が通常使う漢字などの文字は直接入力することはできません。かつては大きな文字盤に全ての漢字の一覧を載せて、特殊なペンで入力したい文字を指示するようなものもありました。現在は通常のキーボードからローマ字またはカタカナで読みを入力し、それを漢字に変換する方式がよく使われます。必ずしも読みと漢字は1対1に対応していないことと、変換の際の様々なキー操作が変換ソフトによって異なる事が大きな2つの問題です。

ここではATOKによる漢字入力について説明します。これはパソコンのワープロソフトのベストセラーの「一太郎」で用いられているかな漢字変換プログラムです。多くの他社のかな漢字変換プログラムがこれと同様のキー操作で変換を行うようになっていますので、これに慣れておけば応用が効きます。

変換開始　かな漢字 キーを押す。(NECのPC98シリーズなどでは、CTRL キーとXFER キーを押す。) このとき画面の右下に「連R漢」と出る。

読みの入力　ローマ字で入力します。長い文節でもかなり正確に変換しますが、ほどほどの長さにとどめる方がよろしいようです。

変換キー　漢字にしたいときは、スペースバー又は 変換 キーを押します。誤った漢字に変換された場合には再度押します。ひらがなにする場合は、PF6 キーを押します。カタカナにする場合は、PF7 キーを押します。

文節の区切りの変更　長い読みを入れると文節の区切りを間違える事があります。そういう場合は ← や → を押すと区切りの位置を変更する事ができます。

確定キー　正しく変換が行われたときには、次の変換すべき文字を入力すれば自動的に確定されます。とりあえず確定させたいときには△を押します。部分的に変換された部分をそこだけ確定したい場合には、 ↓ を押します。

変換終了　変換開始と同じキーを押します。ただし変換の途中では終了できないので注意します。

5.2　テキスト エディタ

文書の編集を行うプログラムのことをテキストエディタと呼びます。ワープロと同じようなものですが、データやプログラムを入力する為のものなので、清書をするための機能はほとんど持っていません。

ワークステーションでよく使われているエディタは、vi系とemacs系に分けられます。カーソルを動かすためのキー操作等が大きく異なりますので、どちらかの系統で慣れると他の系統のエディタは使いにくくなります。また同じ系統のエディタは基本的な操作はほとんど変わりません。

vi系の代表のviはUnixに標準装備のエディタです。ですからUnixの動くマシンではどこでも利用可能です。ただ操作に関して、コマンド入力状態とテキスト入力状態と分かれているので、初心者にちょっと分かりにくい点があります。

emacs系はUnixの標準装備ではありませんが、多くのマシンにインストールされています。基本的に入力した文字はテキストとして挿入されるので、ワープロに慣れた人には違和感が少ないようです。ここではngと言うemacs系のテキストエディタについて説明します。

5.2.1 使い方の基礎の基礎

まず起動をするときは、プロンプトの出ているところで、ng ファイル名△と入力します。ngの後は必ず空白が必要です。

- 入力する文章は入力すればカーソルのある位置にどんどん挿入されます。入れ間違った場合には、BS（FMRでは⇐）を押します。カーソルは矢印キーを押すことによって動かすことができます。
- 入力を終えた後は結果を保存しなければなりません。CTRLキーを押しながら、x、sの2つのキーを順番に押します。
- テキストエディタを終了するには、CTRLキーを押しながら、x、cと2つのキーを順番に押します。変更したのに保存をせずに終了しようとすると、保存しますかと聞いてくるので、yと押します。

5.2.2 使い方の基礎

以下では次のような形でキー操作を説明します。

C-x CTRLキーを押しながら、x キーを押す。

C-x y C-x を押した後、y キーを押す。

C-x C-y C-x を押した後、C-y を押す。

EC x ESCキーを押した後、x キーを押す。

ESC C-x ESCキーを押した後、C-x を押す。

まずカーソルの移動です。一応キーボードの矢印キーでもカーソルが移動できるように設定してありますが、手がホームポジションから離れてしまいますので、できるだけこちらでやるようにしてください。

C-f	カーソルの一文字前進	C-b	カーソルの一文字後退
C-n	カーソルの次行への移動	C-p	カーソルの前行への移動
C-v	次の画面への移動	ESC v	前の画面への移動

通常の文字を入力するとそのままカーソルの位置に挿入されます。カーソルの位置の文字を消去する時は、C-dを入力します。入力中に今入力したばかりの文字を消したい場合には、BS（FMRでは⇐）を押します。

ちょっと変わったコマンドとして1行の長さを揃えるものがあります。電子メールやニュースのテキストをエディタで入力すると、編集しているうちに行の長さはふぞろいになります。行の長さを揃えたい部分の前後を空行で分離し、その中にカーソルを移動してESC qを入力すると、各行はほぼ70字の所に揃えられます。

5.2.3 知っていると便利な使い方

1. まずはその他のカーソル移動のコマンドです。

C-e	カーソルの行の終わりへの移動	C-a	カーソルの行の先頭への移動
ESC >	カーソルのファイルの最後への移動	ESC <	カーソルのファイルの先頭への移動

   この他に、C-gを入力すると移動したいのは何行目かを聞いてくるので、行数と△を入力するとその行にカーソルが移動します。

2. カーソル以降の一行削除には、C-kを入力します。大量に削除する場合は、その先頭の位置でC-@を入力し印(mark)を付け、最後の次の文字のところでC-wを入力します。このときこの削除した内容はバッファと呼ばれる領域に保存されます。ここには1回の削除分しか入りませんが、C-yを入力するとその内容をカーソルの位置に挿入することができます。これを利用してテキストのファイル内での移動が実現できます。コピーをしたい場合には、C-wの代わりにESC wを使用すれば元の部分が削除されません。

3. 検索をしたい時には、C-sを入力して続けて探したい文字列を入力します。このとき最後に△を入力する必要はありません。入力したものと同じ文字列がC-sを入力した時のカーソルの位置以降にあればそこへカーソルが移動します。さらに一致する文字列の位置へ移動させたいときは、もう一度C-sを入力します。検索を止めるときは、ESCを押します。C-rは探す方向が現在のカーソル位置より以前になる他はC-sと同じ動作になります。

4. テキスト中もある特定文字列を全て、または部分的に置き換えるときには、ESC %と元の文字列△と置き換える文字列△を入力します。すると該当する箇所にカーソルが移動しますので、

!	以下の候補を全て一気に置換する
スペースバー	置換実施後次の候補へ
BS	置換しないで次の候補へ
ESC	置換の終了

   のいずれかのキーを押します。

5. 編集中にちょっとngを中断して他のコマンドを実行したくなることがあります。そのような時には、C-zを入力するとngは一時停止してプロンプトが表示されてコマンドの入力が可能になります。逆に中断しているngに戻る時には、fg△と入力します。中断しているのを忘れてlogoutしようとすると

   There are suspended jobs.

   と言われますので、このコマンドでngに戻ってngを終了して下さい。

6. 他のファイルの内容をカーソルの位置に取り込むには、C-x iとファイル名と△を入力します。

5.3 Unixのコマンド

Unixのコマンドは標準でも1000に近い数のコマンドが使用可能です。その中でも良く使われるファイルに関するものをいくつかあげます。なお説明の最後に( )の中に書かれているのは、ほぼ同じ働きをするMS–DOSのコマンド名です。

ls ファイルの名前を表示するコマンドです。ファイルの名前だけでなく大きさなども知りたいときには、ls -l△のように-lを付けます。(dir)

cp ファイルのコピーをするコマンドです。元のファイルとコピー先の指定が必要です。(copy)

```
[miwako]%cp abc efg△
```

ファイルabcの内容がファイルefgにコピーされます。

mv ファイルの移動をするコマンドです。元ファイルと移動先の指定が必要です。cpと異なり元のファイルは消えます。単に名前を変更するのにも使われます。(ren)

```
[miwako]%mv abc efg△
```

ファイルabcがファイルefgになります。

rm ファイルを消去するコマンドです。消したいファイル名を指定します。(del)

```
[miwako]%rm abc△
```

ファイルabcが消去されます。

lpr ファイルの内容をプリンターに印字するコマンドです。印字したいファイル名を指定します。(print)

```
[miwako]%lpr abc△
```

ファイルabcの内容が印字されます。

man コマンドの説明（マニュアル）を表示するコマンドです。説明を見たいコマンド名を指定します。次の部分を見たいときにはスペースバー、前に戻りたい時にはbを入力します。見終わったらqを入力します。(対応するもの無し)

```
[miwako]%man man△
```

manコマンド自身の説明が表示されます。

5.4 演習問題

1. プロンプトが出ているときにng Ng.txt△と入力せよ。するとngの使い方の説明になるので、良く読んで書いてあるとおりにせよ。
2. プロンプトが出ているときにng aaa△と入力しngを起動せよ。そしてこの章の最初の部分を入力し保存せよ。画面の横幅は狭いので適当な所で改行して行末の長さをできるだけ揃えること。また入力したテキストをコピーし、同じ文章を5回並べよ。正しく入力し保存できたらngを終了し、できたファイルaaaの内容をプリンターに出力せよ。
3. 外国でタイプライターで書かれた手紙などでは最後に確かに本人からのものである証拠として手書きの署名を付けることがある。それと同様に自分の発送するメールに決まった文字列を自動的に追加する機能がある。これを利用するにはプロンプトの出ている状態でchgmsig△と入力するとテキストエディタが起動されるので、適当な署名の内容を入力して、保存、終了する。実際にこれを試みてみよ。
4. プロンプトが出ているときにman ng△と入力するとngの説明が表示される。同様にして他のコマンドの説明を幾つか表示させて見よ。

6　電子メール

電子メールとは手紙の本文が電子的に送られるものです。基本的に1対1の通信に用いられるものですが、郵便、電話、ファックスなどと比べると様々な違いがあります。

	利点	欠点
電子メール	すぐ届く (海外でも数秒) 届いたメールの再利用可能 好きな時間に応答可能	相手が読まないと伝わらない 文書など電子的に送れるものしか送れない 電子メールを利用していない人には送れない
郵便	誰にでも届く 封筒に入れば何でも送れる	届くまで時間がかかる 文書作成が面倒
電話	相談などができる	相手が居ないと駄目 話した内容が後に残らない
ファックス	送った内容が双方に残る 相手が不在でも送れる	あまり綺麗に送れない

電子メールは、空間・時間・身分などに拘束されないコミュニケーションを実現する道具とも言われています。会社などでこれを活用した結果、連絡がとりやすくなる、会議などの減少、直接上層部へ情報などが伝わるために中間管理職層が不要になるなどの効果が出ています。

電子メールの応用でメーリングリストと言うものもあります。小規模なグループ内で用いるもので、特定のサーバーにメールを送るとそれが登録されている全てのメンバーに転送されると言うものです。

ここでは電子メールを送ったり読んだりするためのプログラムとして、比較的初心者向きのelmを取り上げます。なお、メールを扱うパソコンで動くプログラムもあります。これを利用するとUnixの事は何も知らなくてもメールを扱うことができます。会社などで本当にメールの機能だけ必要な場合にはよく使われます。

6.1　メールアドレス

郵便物には宛名が必要なように、電子メールを送るのにもメールアドレスが必要です。かつてネットワークがごく小規模な範囲に留まっていた時代にはメールアドレスは、単にユーザー名のみ、またはメールの転送経路を直接示すものでした。今日ではインターネットを通して世界中の国々とメールの交換が可能になっています。これに対応して論理的な構成を反映したメールアドレスを利用するのが現在では普通です。

この学園センターのワークステーションの利用者のmiwakoさんのメールアドレスは次のようなものになります。

```
miwako@cc01.center.sugiyama-u.ac.jp
```

@の前がユーザー名です。その後はピリオドで区切りながら、マシン名(cc01)、部署名(center)、組織名(sugiyama-u)、組織分類(ac:学術)、国名(jp:日本)を示しています。このメールアドレスでインターネットにつながっているどのような組織からもmiwakoさんに電子メールが届きます。

アメリカのみ例外ですが他の国は国名を示すメールアドレスが決まっています。brがイギリスとかcaがカナダと言うような感じです。組織の分類としては、国内ではacの他に会社組織がco、政府関係がgo等に決まっています。組織名はそれぞれ様々なものがありますが、同名異組織では困りますので、JPNICと言う組織がみな異なるように管理しています。ただこの組織が管理しているのはこのレベルまでで、それより前に来る組織内のアドレスは各組織の自由になっています。

通常はこの長いメールアドレスをいちいち書かなくても良いように設定がされています。同じ cc01 に登録された利用者同志ならば、単にユーザー名だけで十分で、@も不要です。

現在では大規模なパソコン通信の利用者にもメールが送れるようになっています。例えばPC–VAN の XXX12345 さんにメールを出すならば、XXX12345@pcvan.or.jp、Niftyserve の YYY67890 さんにメールを出すならば、YYY67890@niftyserve.or.jp に送れば届きます。

6.2 メールを扱うプログラム

電子メール（以下メールと略す）を実現するためには多くのプログラムが働いています。まずメールの内容を編集するためのエディタ、メールを1つのマシン内で配送するプログラム、ネットワークを通してメールを送ったり受け取ったりするプログラム、受け取ったメールの本文を表示するプログラムなどがあります。通常の利用者はそれらの働きを全て知る必要はありませんが、最低限メールを送る方法と読む方法を知らなければなりません。

エディタにも何種類かあったように、利用者が直接触れるメールを送ったり読んだりするプログラムも何種類もあります。Unix に標準的に付いてくる mail の他、多くの人が様々なものを作っています。ここでは比較的初心者向きの elm を取り上げます。基本的にどのプログラムもできることは同じなので、後に別のプログラムを使うことになってもコマンドの対応さえわかれば問題無いでしょう。

6.3 メールを送る

メールを送るにはまず相手のメールアドレスが判らなければなりません。今それが aaa@bbb.ccc だったとします。メールを送るときには、プロンプトが出ている状態で、elm aaa@bbb.ccc△と入力します。すると

```
                Send only mode [ELM 2.4 (JPFake v0.34) PL24]

To: aaa@bbb.ccc
見出し(Subject):
```

と表示されます。ここでメールの見出しを入力し、△を押します。見出しとしてはメールの内容の要約を簡単に短く入れます。学外にメールを出す場合ここに全角の文字を使用するとその部分が文字化けする事がありますのでご注意下さい。

この後はエディタが自動的に起動されます。そこで相手に送るメールの本文を入力します。このとき1行の長さに注意します。（長すぎると途中で切れてしまうことがあります。）画面の右の方が少し空くぐらいにします。この様にするときには ESC q を使うと便利です。保存してエディタを終了すると、

```
次の中から選んでそのキーを押して下さい：s
    メールを送る：s、破棄：f、再編集：e、ヘッダーの変更：h
```

となります。通常はここで△又は s を押すとメールが送られます。本文の編集を再開する時は e、見出し等を修正したいときには h、メールを送るのを中止するならば f を押します。また英文のメールを送る際には、i を押すとスペルチェックをすることができます。

6.4 メールを受け取る

各利用者宛に送られたメールはワークステーションのディスクに各利用者ごとのファイルとして保存されます。そして利用開始の際にメールが着ていると、

```
You have mail
```

等のメッセージがメニューの上の行に表示されます。

メールを読むには、プロンプトが出ているときにelm△と入力します。すると画面が変わり、中央に到着しているメールの一覧、下の所にコマンドの一覧が出ます。何もメールが来ていないときには，中央部分は空白になります。

例えばメールがいくつか届いている場合には次のような感じになります。

```
Mailbox is '/var/spool/mail/maru' with 2 messages [Elm 2.4 (JPfake v0.34) PL24]
N   1   Apr 24   Larry Fenske     (49)   Hello there
N   2   Apr 24   jad@hpcnoe       (84)   Chico? Why go there?

        次の中から選んでそのキーを押してください：終了：q、説明：?、
     メールを読む：<return>、カーソルを上に：j or ↑、カーソルを下に：k or ↓
        削除：d、返事を返す：r、保存：s、削除取り消し：u、転送：f

Command:
```

Nと言うのはまだ読んでいないメールです。読むと空白になります。また削除のコマンドを入れるとDになります。次がメールの番号です。その後に届いた日付、送り主の名前、行数、Subjectの内容と続きます。

この後は色々なコマンドを入力してやってきたメールを片づけます。

- jやkのキーを押すか，矢印キーを押すと文字が反転している部分が上下します。反転しているのが現在指定しているメールです。
- スペース又は△を押すと、指定されたメールの内容を読む事ができます。長い内容のメールの場合、さらにスペースを押すと続きを見る事ができます。前に戻るには[b]を押します。読み終わったら[q]を押します。
- 返事を出すのならば[r]を押します。すると、

  ```
  Command: Reply to message          To: 相手の名前かアドレス
  見出し(Subject): Re: 相手からのメールの見出し
  ```

  と出ます。Subjectとして、相手からのメールのSubjectにRe:が付いた物がセットされているので通常は単に△を押します。すると相手から着たメールの本文が各行の先頭に>が付いた形でコピーされますので、適当にそれを編集して返事が書けます。後はメールを送る時と全く同じ手順になります。
- 受け取ったメールを別の人に転送するならば[f]を押します。すると、

  ```
  Command: Forward                                    転送する内容を編集しますか? (y/n) y
  ```

  となりますので、何らかの変更を行ってから送りたいときには△を押し、そのまま転送するときには[n]を押します。次に、

  ```
  送り先アドレス：
  ```

と出ますので相手のアドレスを入力して△を押します。するとさらに、

```
Command: Forward                    To: 相手の名前かアドレス
見出し(Subject): 相手からのメールの見出し (fwd)
```

となりますので、必要があれば見出しを変更してから△を押します。最初に編集を希望した場合には、この後テキストエディタが起動されるので編集を行います。その後はメールを送る時と同じ手順になります。

- メールを保存する(ファイルにして取っておく)ならば[s]を押します。すると、

```
Command: Save
Save message to: =相手の名前
```

と出ます。通常は△を押して相手の名前のファイルにします。同じ相手からのメールは1つのファイルにどんどん追加される形で保存されます。何か別のファイルに保存したい場合にはそのファイル名を入力します。保存が終わると、最初の画面に戻りNだったものがDに変わります。

- メールの内容を印刷したいときは[p]を押します。するとワークステーションcc01の横にあるプリンターから出力されます。

- メールを消去するならば[d]を押します。すると現在指定しているメールにDと言うマークが付きます。実際に削除されるのはelmを終了するときです。取り消すためには[u]を押します。Dが付いたメールがあると終了時に、

```
Command: Quit                                   本当に削除しますか? (y/n) y
```

と表示されますので、通常は△を押します。

- [q]を押すとelmが終了します。もし読まなかったメールがあると、

```
読まなかったメールをそのままにしますか? (y/n) y
```

と表示されますので、そのまま△を押します。また読んだが保存も削除もしていないメールがあると、

```
Command: Quit                                   読み終えたメールを他へ移しますか? (y/n) n
```

と表示されますので、通常は△を押します。

6.5　保存していたメールの扱い方

相手からのメールが保存してあれば、それに対する返事を書くと言う操作でメールが送れますので相手のメールアドレスを入力するする手間が省けます。

既に保存してあるメールを見るためには、プロンプトが出ているときに、elm -f +ファイル名△と入力します。このときファイル名は相手のユーザー名になります。後の操作は前節と全く同じです。よって保存していたメールの削除などもできます。

これまでに送ったメールの全てがSmailと言うファイルに保存してあります。よってelm -f +Smail△でそれを呼び出して確認などができます。

ところで相手のユーザー名が思い出せない事もあります。そのような場合には、保存したファイルはすべてMailと言うディレクトリーの下にありますので、プロンプトが出ているときに、ls Mail △と入力すれば保存してあるファイル名を知る事ができます。

6.6　メールのマナー

まだ電子メールが広く利用されるようになってそんなに時がたっていませんので、電子メール道？のような作法は確立しておりません。ここでは次のことを推奨しておきます。

- 本文の先頭には自分からだと言うのが判るようなものと付ける。例えば、「三木@椙山女学園大学です」と言った感じ。メールアドレスを見ても誰からのか判らない場合が多いからです。
- 本文はできるだけ簡潔にする。余分なあいさつ抜きに用件に入る。
- 返事を返す場合には、相手の文章も引用して何について答えているのか明確にする。
- 内容が幾つかの事項に別れる場合には別のメールにする。
- もらったメールにはすぐ返事を出す。質問にすぐには答えられないような場合にも、とりあえず質問はわかったと言うような返事を出す。
- 署名には、漢字による自分の名前とか、住所、電話番号など名刺に書いてあるような内容にする。

6.7　演習問題

1. 2人または数人でグループを作り、お互いにメールを送ってみよ。相手が居ない場合には自分自身に送れ。なお演習の仲間同志でメールを交換する際にはメールアドレスとしてはユーザー名のみを指定すれば良い。
2. 受け取ったメールを読み、保存し、返事を送れ。
3. 保存してあったメールを利用してメールを送れ。

7　電子ニュース

コンピュータネットワークでのコミュニケーションの方法として電子メールと共に電子ニュースは大きな比重を占めています。前者が基本的に1対1のコミュニケーションを対象としているのに対して、こちらは多対多のコミュニケーションを実現します。新聞等のマスメディアに対応する物ですが、従来のマスメディアが少数の送り手だったのに対して誰でも送り手になれると言う特徴があります。

電子ニュースは伝える相手が特定できないためにマシンからマシンへ波紋が広がるように伝わりますのであまり速くは伝わりません。と言っても1日あればほぼ世界中に伝わります。パソコン通信等で用いられるBBS(Bulletin Board System:電子掲示板)もほぼ同じ様なものですが、BBSが集中的な管理がなされているのに対して電子ニュースは特に管理者はなく、多くの利用者の善意と努力で運用されています。

電子ニュースは多くのニュースグループにより構成されています。あるものは研究室内で、あるものは大学内で、あるものは世界中に流れています。現在インターネットで流れているニュースは1日あたり約7万記事、その大きさは約200MBになる日もあり、年々その流量は増大しています。

これだけの量になるととても一人の人が目を通す事はできません。ほとんどの人に取っては、役に立たない情報が大部分を占めます。しかし、自分で必要とする情報を求めている事を記事にして投稿して、多くの人から情報の提供を受ける事が可能です。もちろん一方的に情報を得る事ばかりしては、顰蹙を買う事はまちがいありませんので、日頃から有益な情報を提供して、困ったときには助けてもらうような姿勢が大切だと思います。

ここでは学内で読むことのできる電子ニュースの紹介とmnewsと言うニュースリーダー(ニュースを読むプログラム)の使い方を説明します。

7.1　学内で読めるニュースグループ

現在インターネットで流れているニュースは名古屋大学からまず生活社会科学科のbachに来ます。ただしその全てを受け取ると名古屋大学との間の回線に大きな負担をかけますので、その中からcomp、fj、jp、sci、soc、tnn、jp、tokaiと言うニュースグループのみを名古屋大学から送ってもらっています。学内ではbachから各マシンに必要なグループのみ転送しています。cc01には、fj、tnn、jp、tokai、soc と言うグループを転送しています。以下はその概略です。

comp ここはコンピュータに関するグループで初歩的な内容から非常に高度な内容まで含んでいます。(英語)

fj ここはfrom Japanと言う事で日本発の日本語の記事が流れています。もともと大学などの機関で育ったニュースグループなので営利目的の記事は主旨に反するものとされており、うっかり宣伝などをすると非難の嵐になりますのでご注意下さい。

jp ここは日本のJPNICと言う組織がネットワークに関する情報を流しています。

sci ここは科学技術に関するグループです。(英語)

soc ここは社会に関するグループです。その大部分は世界中の様々な民族の文化に関するグループです。(英語)

sugi これは本学で運用しているニュースグループです。まだ記事を投稿する人も無く、読む人もいないのでほとんどサブグループがありませんが、これから段々発展させていく予定です。

sugi.announce	sugi の読者の全員に関係するような話題
sugi.misc	他のグループには入らないような話題
sugi.lib.announce	図書館からの連絡
sugi.lib.newbook	新着図書の紹介
sugi.lib.wanted	図書館に対する要望など
sugi.stats	ワークステーション等の利用状況の報告など
sugi.test	ニュースの投稿の練習用のグループ

sugi に投稿された記事は本学の外へは出ません。もし外部の人にも訴えたいのならば fj を利用することになります。しかしいきなり日本中を相手にするのはどうかと思われる方は tokai を利用すると良いでしょう。

tnn これは IIJ と言う日本のネットワークの会社が運営しているニュースグループです。流量や規模は fj に比べるとはるかに小ぶりですが、こちらでは商品の宣伝のような営利目的の記事も認められています。

tokai 東海地区だけで運用されているニュースグループもあります。学園での催し物の案内、恋人募集などに適当なグループではないかと思います。

各ニュースグループの中はさらにサブサブグループに分かれるような構造になっています。たとえば fj の中の rec の中の sports の中の soccer と言うグループを fj.rec.sports.soccer と示します。こうして大きなニュースグループでは数百のサブニュースグループを含んでいることもあります。

7.2 fj の内容

fj は日本発のニュースと言うことで大部分が日本語で書かれています。そしてそのサブグループとして、comp、rec、sci、soc 等を持っています。そしてその中がさらにサブサブグループに分かれるような構造になっています。

fj の中にも約 280 のグループがあるので全てを紹介できませんがその一部を紹介します。以下の表のオリジナルは富士通の市川さんによるのものです。

```
+-ai AI、人工知能、認知科学などに関する議論
+-announce fj の読者の全員に関係するような記事
+-archives 文書、FAQ などを (将来はプログラムも) 投稿するグループ
+-binaries 実行形式のプログラムなどを投稿するためのグループ
+-books 各種書籍について
+-comp     計算機に関する議論
+-editor 計算機上で動く各種のテキストエディタに関する話題
+-education 教育一般に関して
+-followup 各種の記事に対する、特に fj.announce についての‘‘その後の議論’’ をここで行う
+-freemarket 個人的な売り、または、交換しようとする物について
+-guide fj の手引きなどについて
+-jokes 冗談に関する情報 (ではなくて冗談そのもの)
+-kanakan 各種かな漢字変換に関する話題
+-kanji ネットワークや計算機における漢字の取り扱いなど
+-lan ローカルエリアネットワークについて
+-lang 様々なプログラミング言語について
+-life
|   +-children 子供、子育て、親の役割について
|   +-in-japan 外国人の日本での生活に関する話題
|   +-money お金とのつき合い方に関する議論
|   +-religion 宗教に関する議論
```

+-living 家事・住環境・健康・家族の世話等の、日常生活に関わる話題と情報交換
+-mail ネットワークの電子メールについて
+-misc 投稿すべきグループが他にないとき使う
| +-handicap handicap解消に関する議論、および行動
+-net ネットワーク関係の話題、情報、議論
+-news ネットワークニュースシステムに関する話題
+-os 各種オペレーティングシステムにかかわる話題
+-questions 各種の質問とその回答
+-rec
| +-aerospace 飛行機、飛行船、スカイダイビングなど、空に関すること、物、話題
| +-animation アニメーションに関する情報、議論
| +-autos 自動車に関する記事
| +-av AV(音響/映像機器)に関すること
| +-bicycles 自転車の話題
| +-bus バスについての議論
| +-comics マンガについて
| +-fine-arts 美術に関する話題、美術とは視覚的芸術のことである
| +-fishing 釣りに関する話題
| +-food 料理、うまい店、健康食品などについての情報、議論
| +-games 各種ゲームについて
| +-ham アマチュア無線(HAM)についての話題
| +-idol いわゆるアイドルについて
| +-marine ダイビング、水泳、ヨットなど海洋に関すること
| +-misc その他の楽しみについて
| +-models 模型全般に関する話題
| +-motorcycles 自動二輪(原付を含む)の話題
| +-movies 映画についての情報、感想、議論など。
| +-music 音楽に関するあらゆることについての情報、感想、議論など
| +-mystery ミステリについての話題
| +-pachinko パチンコに関する議論
| +-pets ペットの話題
| +-photo 写真とカメラに関するグループ
| +-play 演劇、劇場等に関する話題
| +-radio ラジオ放送に関する話題
| +-rail 鉄道に関して
| +-sf SFについて
| +-sports スポーツに関するさまざまな情報と議論
| +-tokusatsu 特撮映像作品に関する話題
| +-travel 旅行に関する情報
| +-tv 各種TV番組(CMを含む)について
+-sci
| +-astro 星と宇宙について
| +-bio 生物学に関する議論を行う
| +-chem 化学に関する議論を行う
| +-geo 地球惑星科学に関する議論
| +-human-factors 人間工学についての議論
| +-informatics 情報学に関する議論
| +-lang 自然言語について
| +-math 数学について
| +-medical 医学に関係する話題
| +-misc その他の各種科学について
| +-physics 物理学に関する議論を行う
| +-psychology 心理学に関する議論と探求
+-soc
| +-culture 社会と文化について
| +-environment 自然環境と社会に関わる話題
| +-history 歴史についての議論
| +-human-rights 人権、生命倫理、差別問題などを議論する
| +-law 著作権、生存権などなど。各種法律に関する議論
| +-media 放送・出版などを含む通信媒体の社会的側面に関する議論
| +-men-women 男女差別の議論など、男と女にかかわる社会的なトピックス

```
|   +-misc 社会現象、または、社会科学に関する議論その他
|   +-smoking 喫煙に関する問題やマナーについて
|   +-tech 科学技術と社会の関係についての議論
|   +-traffic 交通・運輸にかかわる、公害問題、行政やマナーなどの社会問題に関する議論
+-sources ソースプログラムを投稿する
+-sys 各社の計算機システムについて。
+-test テストメッセージを fj 全体に流すために用いる
+-wanted なんらかの情報について知りたいとき
+-windows 各種ウィンドウシステムについて。
```

7.3 ニュースの読み方

ニュースシステムを運用するには、記事を配送するプログラムと利用者が記事を読むためのプログラムが必要です。どちらのプログラムもUnixに標準では付いて来ませんので、入手してインストールする必要があります。

ここではmnewsと言う松下電器の宅間顯さんが開発したニュースリーダー（ニュースを読むプログラム）について説明します。実際はこれ以外にも様々なニュースリーダーがありますが、mnewsは小型化、高速化、そして簡単に使用できることを目標に開発されています。多くの学生が少ないワークステーションを同時に使う事を考えてこれにしました。

まず起動の方法ですが、プロンプトの出ているところで、mnews△と入力します。すると次のような感じの画面に変わります。

```
Mini News Reader Version 1.19      Copyright(C) By A.Takuma [NSPL]cc01
ニュースカテゴリ：                                    位置：All [E:m:S]
  最大     未読     モード  ニュースカテゴリ／ニュースグループ名
        108864    [-]     fj
             2    [-]     jp
       1444164    [-]     soc
            92    [-]     sugi
          1859    [-]     tnn
             3    [-]     tokai
```

最大の所の数値は記事の通し番号の最後の値です。未読の所はまだ読んでいない記事の数です。モードの所の[Y]は投稿可能、[N]は投稿不可、[M]はモデレートグループ(投稿可能ですがモデレータのチェックを受けます)、[–]はこの下にサブグループがあることを示しています。

- ニュースグループを選ぶには、ngと同じようにC-p、C-nか、elmと同じようにk、jか矢印キーでカーソルを動かして選択するグループを指定します。そしてスペース押すとそのグループに入ります。階層構造をなしているニュースグループの場合は同様の選択を記事の一覧が現れるまで繰り返します。
- 間違ったニュースグループを選んだり元に戻りたい場合は[q]を押します。最初の画面でqを押すとmnews は終了します。また記事の内容を読んでいる画面以外で[Q]押しても終了します。
- いつも読まないニュースグループには[U]を押すと、ニュースグループにU印が付いて、次回からそのグループは表示されなくなります。

記事の一覧は例えば次のような感じで表示されます。

```
Mini News Reader Version 1.19      Copyright(C) By A.Takuma [NSPL]cc01
ニュースグループ：sugi.lib.announce                        位置：All[E:m:S]
マーク 番号 日付  行数 差出人        サブジェクト [       ] (15-24)
         15 02/17   18 matsuno@lib. linus/u の検索
         16 02/23   10 tmc@jim.sugi  |  linus/u の検索
         17 02/23   42 matsuno@lib.  |  linus/u の検索
         18 02/23   75 tmc@jim.sugi  |  linus/u の検索
         19 02/24   26 miki@ss.sugi Re: linus/u $B$N8!:w(B
         20 02/24   23 maru@jim.sug  || linus/u $B$N8!:w(B
         21 02/24   27 matsuno@lib.  || linus/u $B$N8!:w(B
         22 02/26   60 tmc@jim.sugi  || linus/u $B$N8!:w(B
         23 02/26   32 maru@jim.sug  || linus/u $B$N8!:w(B
         24 03/09   39 matsuno@lib. 中央図書館の CD-ROM
```

マークの所にRとあるのは既に読んだ記事です。何もないのがまだ読んでいない記事です。記事はできるだけ同じサブジェクトが続くように並び換えられています。サブジェクトが|で始まっている記事は別の記事への応答をしている記事です。ここで、

- グループを選択するのと同じようにして記事を選択します。 さらにPやNを押すと未読記事だけを移動できます。
- スペースを押すと記事の内容を読む事ができます。記事の内容表示中は、スペースで次ページへ、bで前ページへ移動できます。途中でqを押せば記事一覧画面に戻れます。
- DやdでRマーク(既に読んだと言う印)を付ける事ができます。その後dならば次の未読記事へ、Dならば前の未読記事へ移動します。
- UやuでRマークを解除します。その後uならば次の記事へ、Uならば前の記事へ移動します。
- sを押すと記事をファイルに保存する事ができます。次にファイル名を力して下さい。もしファイルが存在する時は、旧ファイルに追加するならばy、中止するならばn、上書きをするならばoを押して下さい。
- cを押すとこのニュースグループの全記事にRマークが付きます。確認を求めてきますので、本当にそうするならばy、そうでなけれfboxnを押します。

例えばこんな感じで記事は表示されます。Pathの部分はこの記事がどのようなマシンを経由してきたものかを示しています。Fromに投稿者のメールアドレス、Organizationに投稿者の所属する組織の名称が示されます。Message-IDは全ての記事に互いに異なるものが付けられるようになっているので、特定の記事を指定するときにはこれを用います。

```
Newsgroups: sugi.lib.announce
From: miki@ss.sugiyama-u.ac.jp (Kunihiro Miki)
Subject: Re: linus/u $B$N8!:w(B
Organization: Dept. of Social Sciences, School of Life Studies, Sugiyama Jogakue
n Univ., Japan
Date: Sat, 24 Feb 1996 10:01:05 JST

三木です


>> 自宅からWWWの方です。
>> 大学のｔｅｌｎｅｔからの画面は、利用意欲を触発しません
>> ＃　一度WWWを見てしまうと欲が出ますね。

平成８年度は学園センターのパソコンの更新が行われない事になりましたので、
多くの学生の利用を考えるとlynx又はtelnetからの利用をすすめるしかありま
せん。なんとか来月中には学園センターの端末を40台増やせると思いますが、
あの古いFMRですからWindowsは無理です。システムにかける負荷から言うと、
イメージデータ無しのWWWが一番軽く、次がtelnet、Xによるものが一番重いと
言う感じを受けています。

来年度早い時期に公衆回線から学内LANに入る口が23回線できます。多分ちゃ
んと設定すればそのうちのいくつかをPPPやISDNにもできるのですが、サポー
/var/tmp/mnews_NV.6403 line 1/33 65%
```

ニュースの楽しみ方

1. まず読まないニュースグループには全部 U を付けます。
2. すると余計なニュースグループは表示されなくなりますので、mnewsを起動したら何も考えずにスペースばかり押せば読みたい記事が表示されます。
3. 読み終われば q を押して、またスペースを押すという繰り返しで一つのニュースグループの記事を全て見る事ができます。
4. このSubjectの記事はもう読まないという時には、K を押せば同じSubjectの記事には全て読んだという印が付きます。
5. ニュースグループの終わりまできたらnを押します。すると次のニュースグループに移るかどうか確認を求めてきますので、スペースを押します。
6. n を押しても新しい記事が出てこなくなったら、 Q を押して一発でmnewsを終了させます。

7.4　ニュースへの応答の仕方

既にある記事に対して何かの応答を行いたい場合には2つの方法があります。一つは記事の投稿者にメールを送ること。もう一つはその記事に対する記事を投稿する（フォローすると言います）ことです。また全く新しい記事を投稿することもできます。

ニュースで答える場合には慎重にそのニュースグループの様子をよく理解してからにしてください。compの様な英文ばかり流れている所に日本語で投稿すれば「おまえの記事は全然読めないぞ」と言う抗議のフォローやメールが世界中から殺到するでしょう。fjでも非常に多くの読者が居て様々な価値観でもって記事を読んでいます。自分勝手な内容や誤解されるような書き方にならないように注意して下さい。fjは世界

中に流れています。アメリカでは全てのニュースグループの記事をCD–ROMの形で保存しているそうです。つまらない記事でも永久保存されますのでご注意下さい。

記事に対してメールを送る

ニュースへの返信は、まず記事の一覧の画面でその記事にカーソルを移動してから、R または r を押します。R の場合には元記事の引用が行われます。エディタに入ったら`--text follows this line--`の次の行以降に本文を書き、保存して終了します。あらかじめ.signature と言うファイルに署名の内容を入れておくと自動的に追加されます。記事全文が表示されるので q で終り、送信の確認に y を押せばメールが送られます。中止するならば n、もう一度編集するならば e を押して下さい。送ったメールは elm で送ったものと同様にSmail と言うファイルに保存されます。

記事へののフォロー

記事へのフォローは、まず記事の一覧画面でその記事にカーソルを移動してから、F または f を押します。F の場合には元記事の引用が行われます。後はメールを送る場合と全く同じです。ただし投稿した記事はNews と言うディレクトリーの中のSnews と言うファイルに保存されます。記事の投稿には少し時間がかかることがあります。投稿したままではmnews は新しい記事を認識しないので、一度mnews を終了して再起動するか、ニュースグループの選択画面で +、g と押してmnews に再認識をさせる必要があります。

記事の投稿

投稿したいニュースグループの記事の一覧の画面で a 押すと記事を投稿する事がてきます。すると配布範囲(Distribution)を聞いてきますので、無指定と言うことで△のみを押します。最後にサブジェクト(見出し)を入力します。メールの場合と同様にサブジェクトに日本語は使用しないで下さいサブジェクトを指定しないとニュースの投稿は中止になります。またニュースグループの選択の画面でaを押すとニュースグループ名も入力する必要があります。後はフォローする場合と全く同じになります。

記事の取り消し

自分の投稿した記事は取り消すことが可能です。しかし一旦投稿してしまった記事はマシンからマシンへと流れており、この取り消しも「あの記事は取り消し」と言う特殊な記事が後から流れて行く形になるので、取り消される前に多くの人の目に触れる可能性はかなりあります。ですから投稿するまえによく確認をし、些細な誤りでの記事の取り消しは混乱を招くので避けるほうが良いようです。

記事の一覧画面で取り消したい記事にカーソルを移動し、C (大文字) を押します。すると確認を求めてきますので y を押します。なお他の人の記事は取り消すことはできません。

7.5 演習問題

1. fj.soc、fj.sci、fj.rec のそれぞれの中のサブグループから各一つずつ、それら以外から一つニュースグループを選んで全ての記事を読み、そこでどのような話題が取り上げられているか、200字程度でまとめよ。

2. .signature に適当な自分用の署名を入力せよ。これにはプロンプトが出ている状態でchgnsig△と入力すればエディタが起動されるので、入力し保存、終了すれば良い。

3. sugi.test に何か適当な記事を投稿してみよ。また、sugi.test の適当な記事に何かフォローをしてみよ。

8 WWW (World Wide Web)

最近の新聞・雑誌上でインターネットを使うと言った話の大部分がこのWWW に関するものです。分散ハイパーメディアシステムと言われるWWWを利用すると文書・画像・音声・動画の情報をWWWのサーバーから受け取ることが可能です。ここ数年の間に急激にサーバーの数が増えて、企業が商品の紹介をしたり、大学などが研究資料を公開したり、個人が個人的に収集したデータを公開したりしています。これを電子メールやニュースと比較すると次のようになります。

名称	形態	内容	特徴
電子メール	1 対 1	個人から個人へ文書を伝達する。拡張したものでは、文書以外にデータやイメージ、音声を送れるものもある。	時間がかからない。外部ネットワークを利用すれば海外にも送れる。
電子ニュース	多 対 多	誰でも見ることができ、誰でも投稿できる新聞のようなもの。基本的にテキストしか送れないが、非常に大量の情報が外部ネットワークには流れている。	情報源、広報などに向く。誰でも投稿できる。
WWW	1 対 多	誰でもアクセスできる情報データベースのようなもの。イメージや音声なども提供できる。	電子ニュースと異なり継続的に情報を提供でき、また得ることができる。

このWWWにアクセスするプログラムとしてはMosaicやNetscapeが有名です。これを利用するとテキスト・画像・音声・動画などで情報を受け取ることが可能です。ところがいくらMosaic等が可能と言ってもハードウェアで画像や音声を再生できなければ無理と言うことで、本学のようにテキストしか表示できない端末では、Mosaicなどを利用することはできません。このような貧乏な環境で利用できるプログラムとしてlynxと言うのがあります。ここではこれについて説明します。

8.1 URL (Uniform Resource Locator)

WWWを利用するには、求める情報のある場所や形式を示すURL(Uniform Resource Locator)と言うものを調べる必要があります。例えば次のような形をしています。

```
http://info.cern.ch:80/default.html
```

:の前が形式を示しています。httpがWWWの主となる形式で様々なメディアからなる情報です。//はその後に続くのがマシン名であることを示しています。/以下はディレクトリーやファイル名を示しています。最近ではURLを、多くの雑誌や日経新聞などの記事で見ることができますし、例えばfj.net.infosystems.wwwのようなニュースグループにも出ています。

8.2 WWWを見る

lynxの使い方は、lynx URL△です。するとlynxはURLに従ってWWWのサーバーと接続しようとします。学外にあるサーバーとの接続には時間がかかります。その間に様々なメッセージが出ます。

```
Looking up www.tut.ac.jp
Could not make connection non-blocking.
Http request sent; waiting for response.
Read 512 bytes of data.
```

その後でたとえば豊橋技術科学大学のホームページの場合次のような画面になります。(ホームページとは、接続したときに最初に表示される画面の事です。)

```
                        Toyohashi University of Technology Home Page (p1 of 3)

                        豊橋技術科学大学ホームページ

    本WWWサービスは実験的運用で無保証です。
    くわしくは、こちら（disclaimer）をご覧ください。

    [IMAGE]
    Welcome to Toyohashi University of Technology home page.
    Here is the home page in English //text version
      ---------------------------------------------------------------

          * 新着情報（4/11）

          * 豊橋技術科学大学について

          * 学内組織（各系紹介）

          * 学生・教官情報

          * サークル（4/21）

-- 続きを見るならばスペース --
'↑'、'↓'で移動、'→'で進む、'←'で戻る、B:前画面に戻る、H:説明、Q:終了。
O:オプション、P:印刷、G:URLの入力、M:主画面、/:検索、<delete>:Historyリスト
```

画面のどこかにカーソルがあり、その辺りの字が反転して表示されます。これが現在選択している項目です。[IMAGE]とあるのは本当はここに綺麗？な画像があったことを示しています。それを見ることができないのは致命的な場合もありますが、大抵の場合そのデータを送らなくて済んだ分速く動きます。

この後は、次のようなキーで色々内容を見ることができます。

- スペースを押すと画面の続きが表示されます。
- b を押すと画面の前の部分が表示されます。
- ↑ か ↓ のキーで別の項目を選ぶ事ができます。どこが項目になるかはWWWのサーバー側の設定ですから、こちらは用意されたものの中から選ぶしかありません。
- → を押すと選択している項目の内容を見ることができます。このとき再びWWWサーバーから情報を転送するので時間がかかります。
- ← を押すと、一つ前の選択の画面に戻ることが出来ます。
- q を押すとプログラムが終了してメニューに戻ります。その前にAre you sure you want to quit?[Y]と聞いてきますので、y を押します。

利用者は単に←や→を押すだけですが、そのために物理的には数百キロも離れたサーバーに途中から切り替わる事が自動的に行われています。

現在国内だけでも数え切れないほどの数のWWWサーバーがあります。例えば、

椙山女学園大学　http://www.sugiyama-u.ac.jp
総理大臣官邸　http://www.kantei.go.jp
朝日新聞　http://www.asahi.com
宇宙開発事業団　http://www.nasda.go.jp
米国議会図書館　http://lcweb.loc.gov

などがあります。本学のホームページは平成7年度の生活社会科学科の卒業研究の一環として作成されました。

8.3 演習問題

1. 豊橋技術科学大学のサーバーに接続してみよ。さらにここから金城学院大学に接続してみよ。

9　Cによるプログラミング

9.1　プログラミング言語とは

コンピュータはプログラムと言う形で与えられた指示を非常に高速に忠実に実行する事ができます。コンピュータが基本的にできることは数値の演算に限られるのですが、それらを高速に多様に組合わすことによって様々な仕事ができます。我々は既に何らかの用途に合わせたプログラムを内蔵したコンピュータをよく利用します。ワープロは文書編集用のプログラムが組込まれたコンピュータで、ファミコンはゲームを実行するためのプログラムが組込まれたコンピュータです。このようにプログラムが組込まれたコンピュータは電源スイッチをオンにするだけで使うことができますが、それ以外の用途には使えません。逆にパソコンを初め通常コンピュータと呼ばれているものは、何か仕事をするためのプログラムは持っていませんが、プログラムを実行するための用意はできているので、仕事をするプログラムを入れてやればその仕事をするようになります。

コンピュータに何か仕事をさせたい時には、それをするためのプログラムが必要です。既に多くのプログラムが市販されていますので、大抵の仕事はそれを購入することにより済ますことができます。しかし仕事の内容によっては、一部だけ市販のプログラムでは合わない部分がある事はよく生じます。また全く対応できるプログラムがないとか、貧乏なのでプログラムを購入できない事もあります。そう言った場合の対処の仕方として自分でプログラムを作成する手があります。

プログラムはコンピュータに何をやらせるか、とか仕事をどのように処理するかを記述したものです。具体的にかつ詳細に論理的に記述するために様々なプログラミング言語が考えられました。各言語はそれぞれが記述しようとするプログラムの対象をある程度想定し、対象が合えば楽に記述ができるようになっています。数値計算ならばFORTRAN、事務用ならばCobol、知識情報処理ならばPrologなどが有名です。もちろんFORTRANで全てのプログラムを記述するのも不可能ではありませんが、向いていない分野のプログラムを書くのは、記述が長くなったり複雑な手法が必要になります。

9.2　Cの生立ち

CはUnixと言うオペレーティングシステム(OS: Operating System)を記述するために生まれました。OSは基本ソフトとも訳されますが、他の実際に仕事をするプログラムと異なり、コンピュータシステムを管理・運用するためのプログラムで、通常のプログラムはこれの助け無しでは働くことができないという重要なものです。OSを記述するためにはコンピュータの非常に基本的な動作まで記述できる必要もあります。そのためにCは他のプログラミング言語では扱うことができないような、コンピュータの生の情報を扱えるようになっています。

Cは1972年にアメリカのベル研究所のDennis Ritchieによって開発されました。ただし単独で全く無の状態からCが作られた訳ではなく、Algol60、CPL、BCPL、Bと言う名前のプログラミング言語の系列の最後にできました。当時のベル研究所ではBを使用してUnixを開発していましたが、幾つかの点で問題があったため、Bを拡張する形でCが誕生しました。それ以降Unixの大部分はCで記述され、開発されたUnixは教育機関には無料で、他の企業にも極めて安価にて供給されたので急速に普及しました。

80年代になると、パソコンがより強力になりかつ普及してきました。当初パソコンにおいてはBASIC言語がよく使われました。これは大抵のパソコンに無料で添付されていたためと、初心者向きの簡単な構造を持っていたためと思われます。一方業務用等のパソコンの性能を限界まで引き出さなければならない分野ではコンピュータ本来の命令に近いアセンブリ言語が使われて来ました。C はまず後者のような分野で使われるようになりました。これはアセンブリ言語では大規模なプログラムの開発が困難である事や、OSのようなプログラムでも記述できるCの良さが認められて来たからだと思います。そしてそれにつられるような形で普通のプログラムもCで書かれる事が多くなりました。

現在コンピュータネットワークでよく用いられるワークステーションのOSはUnixです。Unixには標準でCが使用できるようになっているので、様々なプログラムがCで書かれています。

9.3 コンパイルの仕方

我々がCで記述したプログラムはそのままではコンピュータによる実行はできません。通常はコンパイルと言う変換が必要です。その結果できたファイルを実行することにより初めてプログラムが動くことになります。

我々がプログラミング言語で記述したプログラムの事を**ソースプログラム**と呼び、変換して実行できるようになったものを**オブジェクトプログラム**と呼びます。

ワークステーション上でCによるプログラムを作成するには、次のような手順となります。

1. ソースプログラムの入ったファイルを作成する。拡張子は.cにする。
2. gcc ファイル名△でコンパイルする。何かメッセージが出たら、ソースプログラムの誤りがあるので、修正して再度コンパイルをする。
3. a.out△でコンパイルしたプログラムを実行することができる。コンパイルの際に特に指定しなければオブジェクトプログラムは全てa.outと言う名前になるために注意すること。
4. 次のプログラムをコンパイルした時に今のオブジェクトプログラムが消えないようにするには、mv a.out 適当な名前△で名前を変更しておく。以後ここで指定した適当な名前を入れるとプログラムを実行することができる。

実際の実行例を示すと、

1. ng test.c△と入力して、以下のプログラムを入力してから保存、終了する。

```
#include <stdio.h>

main(){
   printf("This is my first program.\n");
}
```

2. gcc test.c△と入力してコンパイルする。
3. a.out△と入力して、以下の画面に以下のように出力されれば良い。

```
This is my first program.
```

4. mv a.out test△と入力する。時々適当な名前としてtest.c等を使う人が居るが、それをするとソースプログラムが消えてしまう。
5. test△とすると先程と同じ結果が得られる。

9.4 初歩のプログラミング

9.4.1 Cのプログラムの形

今の段階ではCのプログラムは必ず次のような形をしているものと憶えてください。

```
#include <stdio.h>

main(){
  文がいくつか
}
```

最初の#includeはコンパイラが持っているstdio.hと言うファイルをここで読み込む事を指示しています。このファイルの中には通常のプログラムに必要な様々な関数や定数の定義が入っています。

main(){はここからmainと言う名前の関数の定義が始まることを示しています。関数については後で詳しく説明します。Cのプログラムには必ずmainと言う名前の関数の定義が存在し、プログラムを実行するとこの関数が最初に実行されます。

その次の「文がいくつか」の部分には様々なCで言う文が;(セミコロン)で区切られて並びます。原則的に書いた順番に実行されます。どのような文があり、どのような働きをするのかを憶えないとプログラムは書けません。しかしそれほど種類はありませんから、憶えることには問題は生じません。いかに組合わせれば目的とする動作をするのか考えるのが難しいのです。

最後の}を忘れてはいけません。必ず{ }や( )は対になっています。この}はmainの定義の終わりを意味します。

9.4.2 printfの使い方

コンピュータは別名電子計算機とも呼びますので、計算の仕方からやっても良いのですが、計算の結果を画面に出す方法が判らなくては正しく計算できたのかどうかも判りません。Cではprintf()と言う関数を使って文字列や計算結果の出力をします。printf()の全ての機能を一度に憶えるのは大変ですので、少しずつ分けて説明します。

まず一番簡単な形としてprintf(”何でもOK”)があります。( )の中に” ”で囲った文字列を入れるとその入れたものだけが、ほぼそのまま画面に表示されます。ここで「ほぼそのまま」と言ったのは、\や%については特殊な意味があるので、そのままの形ではでないからです。とりあえずここでは\nだけ憶えてください。\nがあると画面上のカーソルが次の行の先頭に動き、以後の文字列は次の行の先頭から表示されるようになります。(通常これを改行と呼びます。) コンパイルの仕方の例では、`printf("This is my first program.\n")`となっていたので、This is my first program.と表示された後で改行がなされています。

[演習問題]

画面に次のような物を表示するプログラムを作れ。(hello.c)

以後の演習問題にも( )の中にプログラムを入れるファイル名を指定しますので、できるだけそれに従って下さい。

```
*********
* HELLO *
*********
```

9.4.3 整数型変数

コンピュータの基本的な機能は数値の計算です。複数の値の計算や複雑な計算式の実行には、途中の計算値を記憶する機能が必要です。そのためにどのようなプログラミング言語でも変数と呼ばれるものが利用できるようなっています。

整数型変数は整数を記憶することができます。パソコンのCでは通常-32768から+32767の範囲の値が、ワークステーションでは±約21億までの値が記憶可能です。このように記憶できる値の範囲がコンピュータによって異なるので注意が必要です。複数の変数を使い分けるために変数には全て名前(変数名)が付いています。変数名は英字、数字、下線(_)、で8文字程度以下にします。変数名の先頭は英字でなければばりません。

一部の言語を除き通常のプログラミング言語では、変数を利用する前に宣言をしなければなりません。Cの場合、int a,b,c; と宣言するとa、b、cと言う名前の整数型変数が利用可能になります。またこの宣言は{のすぐ後になければなりません。

実際の変数の利用は次のような感じになります。a=3; 変数aに3を入れる。a=3+5; 加算、a=5-3; 減算、a=5*3; 乗算、a=15/3; 除算、a=16%3;剰余（余り）、b=3; a=b+3; aに6が入る、a=3; a=a+3; aに6が入ります。最後の例では=の両側にaがあります。同じaですが左側のは結果を入れる場所、右側のは変数の値を意味しています。この他に通常の数式同様に( )を使用して演算の順序を指定する事も可能です。

こうして変数に入れた値を表示するにはprintfの次のような機能を使います。

```
printf("%d",a);  aに入っている値が表示される。
```

%はこの後に出力形式の指示があることを示しています。dは10進数を示します。さらに、printfの出力形式の指定の際に%とdの間に桁数を指定する事ができます。%4dとなっていれば、4桁以下の数値の出力の際には数字の左側に空白が補われます。

aの値	%d	%4d
1	1	1
12	12	12
123	123	123
1234	1234	1234

［演習］

次のプログラムを入力し、実行してみよ。(printf.c)

```
#include <stdio.h>

main(){
   int i,j;

   i=3;j=5;
   printf("i+j=%d  i-j=%d \n",i+j,i-j);
}
```

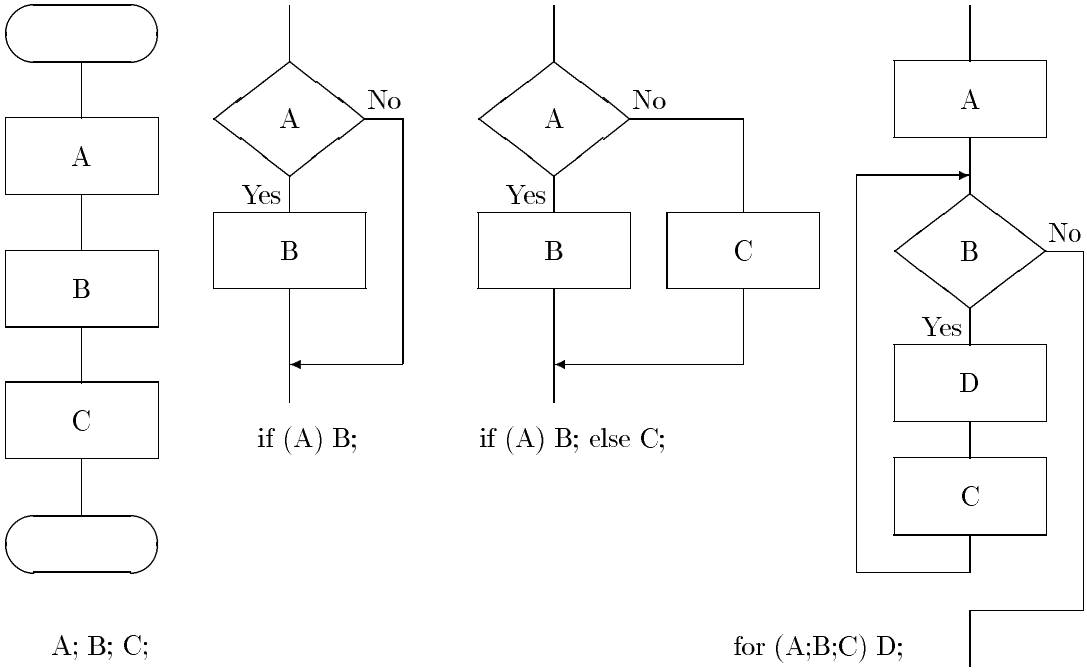
上記のプログラムを実行すると、

```
i+j=8  i-j=-2
```

のように表示されます。このように変数の代わりに式を書いても良いし、複数の値を一度に表示する事もできるし、%d以外の部分はそのまま表示されます。

9.4.4 フローチャート (流れ図)

フローチャートはプログラムの構造を図示するのに用いられます。プログラムの最初と終わりを長丸、処理を長方形、判断をひし形で表します。



必ずしもCの1つの文を1つの箱に対応させる必要はありません。通常はプログラム全体の大きな流れを数個の箱で書き、次にその箱の1つ1つを別のフローチャートでより詳しく記述するような事が行われます。

9.4.5 for文

for文はCで非常によく使われるもので、処理のくり返しを表現する代表的なものです。その形式と動作は次のようになります。

```
for ( A ; B ; C ) D ;        Dを繰り返し実行する
```

まずAを実行する。Bの条件を調べて、だめならば、for文の実行を終えて次の文へ。OKならばD、そしてCを実行して、ふたたび条件判定のところへもどる。判りにくい方はフローチャートの例のところにこのfor文をフローチャートで書いたものがあります。このような動作をします。具体例は次のようになります。

```
for (i=0;i<100;i++) printf("%d\n",i);
```

ここでi++と言う表記が出てきましたが、これは変数iの内容を1増やすと言うC独特の式です。まず変数iの値がゼロになります。条件判定は変数iの値が100未満か？となっているので、変数iの値が100以上になったらおしまいです。100未満ならば、iの値を表示して改行する。そしてiの値を1増やして条件判定にもどる。この繰り返しになります。

注意点として、Dの部分が複数の文からなる場合には{ }で囲う必要があります。次の例で左側はHaHaHaとCyaCyaCyaがそれぞれ交互に10個出ますが、右側ではHaHaHaが10個出た後に1回だけCyaCyaCyaが出ます。

```
for (i=0;i<5;i++) {
   printf("HaHaHa\n");
   printf("CyaCyaCya\n");
}
```

```
for (i=0;i<5;i++)
    printf("HaHaHa\n");
 printf("CyaCyaCya\n");
```

出力：

```
HaHaHa
CyaCyaCya
HaHaHa
CyaCyaCya
HaHaHa
CyaCyaCya
HaHaHa
CyaCyaCya
HaHaHa
CyaCyaCya
```

```
HaHaHa
HaHaHa
HaHaHa
HaHaHa
HaHaHa
CyaCyaCya
```

[演習]

次のプログラムを入力して実行してみよ。ただし実行する前にどのような結果が得られるかよく考えてみること。例えば最初の for 文では何が得られるか？次の for 文では cat を出している様だがどのような形に並ぶか？予想と異なる結果が得られた場合にはよくその原因を考えること。(for.c)

```
#include <stdio.h>

main(){
   int i;

   for (i=1;i<10;i++) printf("%d %d %d\n",i,i*i,i*i*i);
   for (i=0;i<200;i++) printf("cat ");
}
```

[演習問題]

1. 以下の図形を縦横5匹？ずつ計25匹表示するプログラムを作れ。(tanuki.c)

```
 0 0
(o.o)
=( x )=
 U U
```

2. 次のような九九の表を出力するプログラムを作れ。(kuku.c)

```
***  The Multiplication Table  ***

1   2   3   4   5   6   7   8   9
2   4   6   8  10  12  14  16  18
    中略
9  18  27  36  45  54  63  72  81
```

9.4.6 入力用関数 (scanf)

キーボードから入力した数値などを変数に入れるためには、scanf と言う関数が使えます。

```
scanf("%d",&i);      変数 i にキーボードから入力された 10 進数を入れる
```

&は、アドレスを求める演算子です。&iで、変数iが実際にメモリーのどこにあるかを示します。scanfは形式として%dが指定された場合には、入力されたものを10進の数値とみなして解釈し、与えられた変数の位置(アドレス)にそれを書き込みます。

［演習］

以下のプログラムを入力し、実行してみよ。(scanf.c)

```
#include <stdio.h>

main(){
   int x,y;

   printf("x = ");scanf("%d",&x);
   printf("y = ");scanf("%d",&y);
   printf("x+y=%d, x-y=%d, x*y=%d, x/y=%d\n",x+y,x-y,x*y,x/y);
}
```

scanf関数自体は何も画面に出力を行わないので、この例のようにscanfの前にprintfを使用して、何を入力しようとしているのか示すのが普通です。

9.4.7 条件判断(if)

コンピュータは計算だけでなく判断することが可能です。もちろん人間のように玉虫色なファジーな判断は無理ですが、yesまたはnoの論理的な判断をし、その結果によって異なる文を実行することができます。

`if（条件） A;`	条件を満たせばAを実行する

条件が成立したときのみAが実行されます。

`if（条件） A; else B;`	条件を満たせばAをそうでなければBを実行する

条件が成立するとAが実行されて、そうでない時にはBが実行されます。AとBともに複数の文からなる場合には{ }で囲みます。条件としては、

`x<5`	xの値が5より小さい。	`x==5`	xの値が5と等しい。
`x!=5`	xの値が5と等しくない。	`x<=5`	xの値が5と等しいか少ない。
`(x==3) || (x==5)`	xは3に等しいか5に等しい。		
`(x>5) && (y<3)`	xは5より大きく、かつyは3より小さい。		

などがあります。2つの値を比較するのが基本で3つの数が等しいかどうかを判断する際には、2つづつ比較するようにしなければなりません。つまりx、y、zがみな等しい条件は、x==y==zではなく、必ず(x==y) && (y==z)と書かねばなりません。

［演習］

以下のプログラムを入力し、実行してみよ。(hantei.c)

```
#include <stdio.h>

main(){
   int h,w;
```

```
   printf("Height (cm) = ");scanf("%d",&h);
   printf("Weight (Kg) = ");scanf("%d",&w);
   if ((h-100)*0.9>w) printf("You are smart!\n");
    else              printf("You are too heavy!\n");
}
```

[演習問題]

1. 身長と収入を尋ねて、それぞれ170と1000以上ならば、I love you.と答えるプログラムを作れ。条件に合わない場合には適当なメッセージを出す事。(2kou.c)

```
Height=177△      Height=150△                    Height=190△
Income=1500△     Income=2000△                   Income=600△
I love you.      Bye bye, you are too small.    Bye bye, you must work hard.
```

2. 3つの数値を入れたら、その中で最大のものを答えるプログラムを作れ。(max3.c)

```
A=5△             A=7△             A=7△             A=8△
B=6△             B=3△             B=12△            B=5△
C=7△             C=4△             C=8△             C=8△
Max=7            Max=7            Max=12           Max=8
```

[応用問題]

1. 3つの数値を入れたら大きいものの順に表示するプログラムを作れ。(sort3.c)

```
A=5△             A=7△             A=7△             A=8△
B=6△             B=3△             B=12△            B=5△
C=7△             C=4△             C=8△             C=8△
Answer=7, 6, 5   Answer=7, 4, 3   Answer=12, 8, 7   Answer=8, 8, 5
```

2. xの字で直角三角形を表示するプログラムを作れ。その大きさは入力して指定する。(3kaku.c、3kakua.c、3kakub.c、3kakuc.c)

```
Size=8△          Size=7△          Size=6△          Size=5△
x                xxxxxxx              x            xxxxx
xx               xxxxxx              xx             xxxx
xxx              xxxxx              xxx              xxx
xxxx             xxxx              xxxx               xx
xxxxx            xxx              xxxxx                x
xxxxxx           xx              xxxxxx
xxxxxxx          x
xxxxxxxx
```

9.4.8 ループ (1)

コンピュータの処理能力を生かすにはできるだけ大量のデータを処理してもらうことが重要です。数値の計算にしても僅か数個の数値を計算するのならば、プログラムを書くよりも電卓を利用した方が速いでしょう。しかし数百個のデータを加えて平均を求めるような場合に、プログラムに数百回加算する文を書くのでは大変です。そのような場合には、その規則性に注目して少ない文で同様な働きをするプログラムを作成するのが普通です。

くり返しの回数があらかじめ決まっているような場合には、既に学んだfor文を使用するのが普通ですが、さらにちょっとした手法が必要になります。以下のプログラムは1から与えられた数までを全て加えるものです。

```
#include <stdio.h>

main(){
   int i,n,s;

   printf("N = ");scanf("%d",&n);
   s=0;
   for (i=1;i<=n;i++) s=s+i;
   printf("Sum = %d\n",s);
}
```

実行例

```
N = 5△
Sum = 15

N = 10△
Sum = 55
```

［演習問題］

1. nの階乗(n!)を計算するプログラムを作れ。なおn!=1*2*3*...*(n-1)*n である。(kaijo.c)

   ```
   N=5△
   N!=120
   ```

2. 1からnまでの奇数の総和を求めるプログラムを作れ。(kisuwa.c)

   ```
   N=15△
   Kisuwa=64
   ```

［応用問題］

1. ${}_nC_r$(n個の中からr個を取り出す組み合わせの数: ${}_nC_r = \frac{n!}{(n-r)!r!}$)を計算するプログラムを作れ。ただし繰り返しはforを1個だけ使用すること。(ncr.c)

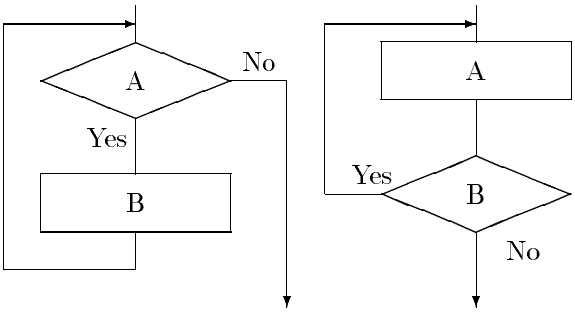
   ```
   N=5△
   R=3△
   nCr=10
   ```

2. 1からnまでの奇数の総和を求めるプログラムを作れ。(kisuwaa.c)

   ```
   N=15△
   1+3+5+7+9+11+13+15=64
   ```

9.4.9 while文

forと同様に繰り返しを行うもので、先に条件判定するものと、後で行うものがあります。



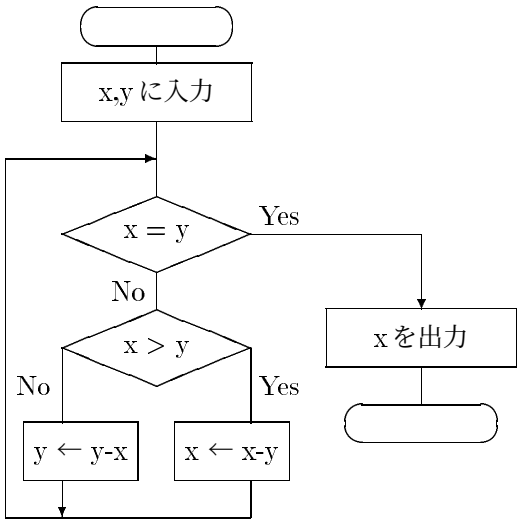
while (A) B;　　do A; while (B);

左側のwhileではまずAの条件を調べます。OKならばBを実行します。そして再びAの条件を調べます。Aの条件がOKの間Bを繰り返し実行します。実はこれはfor(; A;) B; と同じ働きになります。

右側のwhileはまずAを実行します。それからBの条件を調べます。条件がOK ならば再びAを実行します。Bの条件がOKの間Aを繰り返し実行します。先程のwhileと異なるのは、条件に係わらずこちらのwhileは最低１回の実行がなされることです。左側のwhileでは条件が最初から駄目な場合には１回も実行されません。

［演習問題］

1. フローチャートに基づいて最大公約数を求めるプログラムを作れ。(gcd.c)




```
X=12△
Y=15△
G.C.D.=3
```

2. 任意の自然数に対して、偶数ならば2で割る、奇数ならば3倍して1を足すと言う操作を繰り返すと、必ず１になることが知られている。実際に入力した数に対して同様な操作を行うプログラムを作れ。(test231.c)

```
x = 5△
x = 16
x = 8
x = 4
x = 2
x = 1
```

［応用問題］

1. 最小公倍数を求めるプログラムを作れ。(lcm.c)

   ```
   X=12△
   Y=15△
   L.C.M.=60
   ```

2. 任意の自然数に対して、偶数ならば2で割る、奇数ならば3倍して1を足すと言う操作を繰り返すと、必ず1になることが知られている。実際に入力した数に対して同様な操作を行うプログラムを作れ。演習問題と計算は同じだが出力の形式が異なる事に注意せよ。(test231n.c)

   ```
   x = 5△
   1 : 16
   2 : 8
   3 : 4
   4 : 2
   5 : 1
   ```

9.4.10 ループ (2)

以下は任意の数のデータの最大値を求めるプログラムです。データの個数が予め判っている場合にはforを使って繰り返すほうが簡単です。ここでは正のデータを入力して行って、最後に数値のゼロを入力してプログラムにデータの終わりを知らせるようにしています。

```
#include <stdio.h>

main(){
   int i,m;

   m=0;
   do {
      printf("Data = ");scanf("%d",&i);
      if (i>m) m=i;
   } while (i>0);
   printf("Max  = %d\n",m);
}
```

実行例

```
Data = 5△
Data = 3△
Data = 10△
Data = 0△
Max  = 10
```

［演習問題］

1. 上記のプログラムをフローチャートに直せ。

2. 平均値を求めるプログラムを作れ。ただしデータは全て正の値でデータの終わりを示すのに0を用いるものとする。(heikin.c)

   ```
   Data=20△
   Data=30△
   Data=10△
   Data=40△
   Data=0△      （データの終わりを示す）
   Mean=25      ((20+30+10+40)/4)
   ```

［応用問題］

1. 任意の数のデータの最小値を求めるプログラムを作れ。ただしデータは全て正の値としデータの終わりを示すのに0を用いるものとする。(mmin1.c)

```
Data=20△
Data=30△
Data=10△
Data=40△
Data=0△
Min =10
```

2. 任意の数のデータの最小値を求めるプログラムを作れ。ただしデータは全て正の値としデータの終わりを示すのに0を用いるものとする。なお以下の例のように、最初からデータの終わりの0が入力されても対応するようにする。(mmin2.c)

```
Data=20△
Data=30△
Data=-10△
Data=40△
Data=0△
Min =-10
```

```
Data=0△
No Minimum
```

9.4.11 コメント

プログラムの説明等をプログラム中に埋め込むことができます。/* と */で囲みます。例は次のbreak文の例を見てください。

9.4.12 break文

breakを実行すると一番内側のループ(forでもwhileでも)から抜け出る事ができます。これを利用した際には、ループの次に実行が移るのが、正常にループを終了した場合と、途中でbreakで抜けてきた場合の２通りになるので注意が必要です。

```
/*      素数の判定プログラム      */

#include <stdio.h>

main(){
   int i,s;                    /*  iには割ってみる数、sには素数かどうか調べる数が入る。 */

   printf("?=");scanf("%d",&s);
   for (i=2;i<s;i++)                              /*  2からs-1までの数で割ってみる。          */
      if ((s%i)==0)                               /*  もしsがiで割れたとすると。。。          */
         break;                                   /*  forを中断する。                          */
   if (i==s) printf("%dは素数である。\n",s);  /* forを完了して出てきたときはi==sである。*/
   else      printf("%dは素数ではない。%dで割れる。\n",s,i);
}
```

［演習問題］

1000以下の素数を表示するプログラムを作れ。(s1000.c)

```
  2   3   5   7  11  13  17  ...
 73  79  83  89  97 101 103  ...
179 181 191 193 197 199 211  ...
...
```

［応用問題］

1. 与えられた数を素因数分解するプログラムを作れ。(bunkai1.c)

```
N=120△
120=2*2*2*3*5
```

2. 与えられた数を素因数分解するプログラムを作れ。(bunkai2.c)

```
N=120△
120=2^3*3^1*5^1
```

9.4.13 配列

同じ型の変数を複数個、配列と言う形で扱う事ができます。その場合宣言の際に必要な個数を要求しなければなりません。例えば、int a[10]; とすると、a[0]、a[1]、a[2]、... a[9] と言う変数が計10個使えるようになります。

このとき[ ]の中は変数を含む数式でも構いません。変数の値によって同じものが別の変数を意味するようになります。例えば、scanf("%d",&x); はいつも変数xに入力する意味ですが、scanf("%d",&a[i]); は変数iの値が0ならばa[0]に入力しますし、変数iの値が9ならばa[9]に入力します。だからと言って、宣言した範囲を越えた値を変数iに入れて使用してはいけません。通常[ ]の中の値は宣言した範囲内かどうか調べないので、プログラムや他の変数の内容を破壊する事になります。

次の例は配列を使った入力されたデータを入力した逆順に出すプログラムです。データの入力の部分を見ると1つのscanf()で全てのデータの入力が可能になっています。

```
#include <stdio.h>

main(){
   int i,n,data[100];

   printf("How many data? = ");scanf("%d",&n); /*  データの個数を入力する */
   for (i=0;i<n;i++) {                         /*  データを配列に読み込む */
      printf("data=");scanf("%d",&data[i]);
   }
   for (i=n-1;i>=0;i--)                        /*  データを逆順に出力する */
      printf(" %d",data[i]);
   printf("\n");
}

How many data? = 4△
data=1△                1はdata[0]に入る
data=2△                2はdata[1]に入る
data=3△                3はdata[2]に入る
data=4△                4はdata[3]に入る
 4 3 2 1
```

i--はi++の反対の働きで、変数iの内容を1減らします。

［演習問題］

上記のプログラムを改良し、データの終わりを0で示すことにし、最初にデータの個数を入れなくても良いようにせよ。(reverse.c)

```
data=5△
data=8△
data=9△
data=0△
 9 8 5
```

［応用問題］

分散を定義通り計算するプログラムを作れ。データは1つ以上あり、全て正の値とし、データの終わりは0で示すものとする。分散は与えられた各データから平均を引き、自乗したものの総和を個数で割って求められる。(variance.c)

```
data=10△
data=30△
data=20△
data=40△
data=0△
Mean=25  Variance=125
```

9.4.14　文字型変数

Cのプログラムでは文字と文字列を区別して扱います。文字は俗に半角と呼ばれる大きさの文字1つを示します。プログラム中では、'a'の様に文字を'(Single quote)で囲います。一方文字列は文字が0個以上連続したものと定義されます。プログラム中では、"abc"の様に文字を"(Double quote)で囲います。0個でも文字列ですので、""も文字列です。また文字には空白も含みますので、"I have a pen."も1つの文字列です。

これらの文字データを扱うために文字型の変数があります。宣言はintの代わりにcharを用います。普通の文字型変数1つには1文字しか入りません。文字列を入れるためには文字型の配列を用います。配列には文字列を構成する文字が1つずつ入りますが、それに加えて最後には必ず'\0'(ヌル文字)が入ります。例えばwordという文字型の配列に"bat"と言う文字列を入れると、word[0] には'b'、word[1]には'a'、word[2]には't'、word[3]には'\0'が入ります。

word	[0]	[1]	[2]	[3]	[4]	...	[29]
内容	'b'	'a'	't'	'\0'	?	...	?

文字型の配列に入った文字列は、1文字ごと扱うことも可能です。上記のような状態で word[0]='h'; を実行するとwordの中の文字列は"hat"になります。ただ一文字しか含まない様な"a"と言う文字列でも'a'と言う文字とは異なるので注意して下さい。

```
#include <stdio.h>

main(){
   int i,l;
   char word[30];                          /*  最大30文字入れる事ができる。              */

   printf("Word = ");scanf("%s",word);     /*  文字列の入力の時は%sにする。&や[ ]は不要 */
   for (l=0;word[l]!='\0';l++) ;           /*  変数lの値を文字列の終わりまで増やす       */
   printf("Word Length=%d\n",l);
   for (i=l-1;i>=0;i--)                    /*  単語を逆順に出力する                      */
      printf("%c",word[i]);                /*  文字を出すときには%cを用いる              */
   printf("\n");
}
```

```
Word = computer△
Word Length=8
retupmoc
```

これまでscanf()やprintf()の中で%dは数値を意味していました。同様に%sは文字列を、%cは文字を意味します。文字列の入力の時だけちょっと違うので注意して下さい。

	整数型	文字型	文字列型
宣言	int n;	char c;	char s[80];
入力	`scanf("%d",&n);`	`scanf("%c",&c);`	`scanf("%s",s);`
出力	`printf("%d",n);`	`printf("%c",c);`	`printf("%s",s);`
代入	n=10;	c='x';	不可

［演習問題］

1. 入力した単語の中に母音(a、e、i、o、u)が幾つ含まれるか計算するプログラムを作れ。(vowels.c)

```
Word=computer△
There are 3 vowels in 'computer'.

word=big△
There is a vowel in 'big'.
```

2. 入力された文字列の先頭3文字と残りの文字列を別々に表示するプログラムを作れ。なお、入力される文字列の長さは3文字以上あるものとする。(cut3.c)

```
String=personal△
head=per, body=sonal
```

［応用問題］

1. 入力した単語の中でもっとも短いものの長さを表示するプログラムを作れ。(shortest.c)

```
Word=book△
Word=are△
Word=personal△
Word=today△
Word=Z△          ← 一番最後はZを入れるが、これは考えない。
The shortest word length is 3.
```

9.4.15 文字列を扱う関数

Cにはいくつかの文字列を扱うための関数が標準で用意されています。今sとt を文字列変数（char s[100],t[100];のように宣言した物）とします。

```
gets(s)      sに１行丸ごと取り込みます。
puts(s)      sの内容を表示して改行します。
strcat(s,t)  sに入っている文字列の後にtに入っている文字列を追加します。
strcmp(s,t)  sに入っている文字列とtに入っている文字列を比較します。結果は数値です。
    strcmp("abc","def") → 負の値     strcmp("abc","abc") → ゼロ
    strcmp("def","abc") → 正の値
strcpy(s,t)  tに入っている文字列をsにコピーします。
strlen(s)    sに入っている文字列の長さを求めます。
strstr(s,t)  sの中にtが含まれるかどうか調べます。なければNULLを返します。
```

Cでは文字列を代入する事はできません。つまりs=”abc”;はできないので、かわりにstrcpy(s,”abc”);を使います。gets()とscanf()の違いは、入力されたものに空白が含まれるときに、gets()ならば空白も含めて全て変数に入りますが、scanf()ならば空白の手前までしか変数に入りません。puts(s)は、`printf("%s\n",s)`と全く同じです。

注意:これらの関数のうちgets()、puts()以外を使用する際には、プログラムの先頭に`#include <string.h>`が必要です。

以下は上記の関数を使用したプログラムの例です。

```
#include <stdio.h>
#include <string.h>

main(){
   char word[20],sentence[80];

   printf("Sentence = ");gets(sentence);
   printf("Sentence Length = %d\n",strlen(sentence));
   strcpy(sentence,"");strcpy(word,"");
   do {
      strcat(sentence," ");
      strcat(sentence,word);
      printf("Word = ");gets(word);
   } while (strcmp(word,"end")!=0);
   puts(sentence);
}
```

```
Sentence = I love you△
Sentence Length = 10
Word = I
Word = love
Word = you
Word = end
 I love you
```

［演習問題］

1. 入力した姓と名前の順番を入れ替えて表示するプログラムを作れ。ただし姓と名前の間には1つだけ空白があるものとする。(exchg.c)

   ```
   Full Name=Miki Kunihiro△
   In English=Kunihiro Miki
   ```

2. 文中に指定した文字列が含まれているときにyesと答えるプログラムを作れ。(search.c)

   ```
   Sentence = I have a book.△
   String = oo△
   yes
   ```

3. 入力した文から単語を取り出すプログラムを作れ。(単語間にはスペースが必ず1つだけあり、行末には何もつかないものとする。) (getword.c)

   ```
   Sentence = I have a book△
   Word = I
   Word = have
   Word = a
   Word = book
   ```

9.4.16 switch文

Xの値がAだったらこうして、Bだったらああして。。。と言った事をしたい時には、ifを何個も書く必用がありましたが、このような場合にはswitch文と言うものが使えます。

```
switch (X) {
   case A : こうして; break;
   case B : ああして; break;
   default : その他する;
}
```

Xの部分には任意の式が書けます。AやBの部分には、数値とか文字を書きます。ここには変数を含む式は書けません。「こうして」の部分には複数の文を{ }で囲わなくても書けます。その後のbreakは通常必要ですが、無くともエラーにはならず、「こうして」をやった後で「ああして」もするような動作になります。Xの値がAでもBでもなかった場合にはdefault以降の「その他する」の部分が実行されます。そのような物が必要でない場合にはdefaultの行は省略可能です。

9.4.17 関数の定義のやり方(1)

Cでは自分で好きな関数を定義して利用することができます。と言うか数十行を越えるプログラムは、1画面に収まる程度の長さの関数に分割して作成するのが普通です。

これまでどのプログラムもmain(){...}と言う形をしていました。実はこれはmain()と言う関数を定義していた事になります。ですからtekitou()と言う関数を定義したい時にはmain()同様に行えば良いのです。

```
tekitou(){
   tekitou()の内容
}

main(){
   main()の内容
}
```

main()の内容でtekitou()を利用したいときには、tekitou();と言う行を書けばできます。何度利用してもかまいませんからプログラム中に繰り返し用いられる部分を関数として定義すると、プログラムの長さを短く、見易くすることができます。さらにtekitou()の中で別の関数を利用することもできます。この場合利用される関数の定義を、利用する関数の前に書くことを薦めます。

tekitou()の中で変数の宣言をしてそれを使うことができます。ただしそこでどのような事を行っても他の関数で宣言された変数の値は変化しません。例えば、

```
tekitou(){
   int i;
   i=100;
}

main(){
   int i;
   i=3;
   tekitou();
   printf("i=%d\n",i);
}
```

のような場合、main()で最初に3が変数iに入ります。そしてtekitou()の中では変数iに100を入れてますが、main()にもどるとそれとは関係無く、画面にはi=3と表示されます。このような関数の中でしか通用しない変数の事を局所変数(ローカル変数)と呼びます。逆に大域変数(グローバル変数)と呼ばれるものも

あります。関数の定義の前に宣言した変数はそれ以降の関数の中で利用することができますが、どこかの関数でその値を変更するとその変更は他の関数に対しても有効になります。

```
int i;      /*  関数の定義の外にあるので、大域変数  */

tekitou(){
   i=100;  /* 変数 i は既に宣言されているので、すぐ使える  */
}

main(){
   i=3;
   tekitou();
   printf("i=%d\n",i);
}
```

この場合にはi=100と表示されます。各関数の定義の際にint i;が無いことに注意して下さい。もしint i;をそのまま残していると、ローカル変数の宣言の方が優先されます。つまり次のような場合には、i=3と表示されます。よく間違える点なので、注意してください。

```
int i;      /* 関数の定義の外にあるので、大域変数  */

tekitou(){
   int i;  /* tekitou() の中では、変数 i は局所変数  */
   i=100;  /* これは局所変数 i に入れるだけ          */
}

main(){
   i=3;
   tekitou();
   printf("i=%d\n",i);
}
```

関数を利用する側から関数に値を渡すことができます。例えば、

```
tekitou(int height){
   if (height>170) printf("You are tall!\n");
}

main(){
   tekitou(155);
   tekitou(165);
   tekitou(175);
   tekitou(185);
}
```

のような形になります。最初の行の( )の中のintが変数の型を示し、その次が変数名です。tekitou()の中ではここで指定した変数がローカル変数と同様に使えます。複数の値を渡したいときには、型と変数名の後にカンマを入れて型と変数名を繰り返します。上のプログラムでは最初の呼び出しで155がheightに入り、次の呼び出しで165がheightに入ります。最終的には4つの値がheightに渡されるために、画面には2回You are tall!が表示される事になります。

［演習問題］

数回にわけて三目並べのプログラムを作成する。今回は関数に分割しないプログラムを元にして、これを関数を使った形に直せ。(ox1.c)

```
#include <stdio.h>

main(){
   int i,j,k,ban[10];

   printf("\n  ***  Tick-Tack-Toe Game  ***\n");
   printf("       ver.1 Man - Man\n\n");
   for (i=1;i<=9;i++) ban[i]=0; /* ban[i]がゼロならばiマスは空         */

   j=1;                         /* jが1ならばoの番、−1ならばxの番 */
   for (i=0;i<9;i++) {
      do {
         printf("==> ");scanf("%d",&k);
      } while ((k<1) || (k>9) || (ban[k]!=0));
      ban[k]=j;
      j=-j;
      printf("\n      |   |\n");
      switch(ban[1]) {
         case -1 : printf("    X |"); break;
         case  0 : printf("    1 |"); break;
         case  1 : printf("    O |");
      }
      switch(ban[2]) {
         case -1 : printf(" X |"); break;
         case  0 : printf(" 2 |"); break;
         case  1 : printf(" O |");
      }
      switch(ban[3]) {
         case -1 : printf(" X\n"); break;
         case  0 : printf(" 3\n"); break;
         case  1 : printf(" O\n");
      }
      printf("      |   |\n");
      printf("   ---+---+---\n");
      printf("      |   |\n");
      switch(ban[4]) {
         case -1 : printf("    X |"); break;
         case  0 : printf("    4 |"); break;
         case  1 : printf("    O |");
      }
/*   途中省略   */
      printf("      |   |\n\n");
   }
}
```

関数を使わないと大変長くて何をしているか判らない物になるが、ここでintialize()、input()、display()と言う関数の定義を適当に作れば、main() の部分は次のようになる。

```
main(){
   int i,j;

   initialize();      /*  メッセージの表示とban[ ]の初期化  */
   j=1;

   for (i=0;i<9;i++) {
      input(j);       /*  空いているところに入力  */
      j=-j;
      display();      /*  ban[ ]の表示   */
   }
}
```

関数に分割したために、main() の内容を見るだけである程度のプログラムの動作が理解できるようになっている。なお、単純にここで示した関数を別に定義するだけならば、全体の行数はむしろ増加する。ここで display() の内容をさらに別に定義した関数を複数回呼ぶようなものにすると、全体の行数を減らすことができるので、是非試みること。

```
  ***  Tick-Tack-Toe Game  ***
       ver.1 Man - Man

==> 1△
       |   |
    0  | 2 | 3
       |   |
   ----+---+----
       |   |
    4  | 5 | 6
       |   |
   ----+---+----
       |   |
    7  | 8 | 9
       |   |

==> 5△
       |   |
    0  | 2 | 3
       |   |
   ----+---+----
       |   |
    4  | X | 6
       |   |
   ----+---+----
       |   |
    7  | 8 | 9
       |   |
```

以下省略。

9.4.18 関数の定義のやり方 (2)

関数本来の働きと言えば、与えられた値に対して何か値を返すことです。C の関数は数値だけでなく、文字なども返すことができます。そのやり方を以下に説明します。まず関数定義の先頭に返す値の型を書きます。何も返さない場合には void と書きます。(前節のように何も型を書かない場合には、整数が返されるものと解釈されます。)

関数から値を返す場合には return(...); を使います。この ( ) の中に入っている値が関数から返す値になります。なおこの return が実行されると、関数の実行は終わってこれを呼び出した方へ実行が戻ります。

```
int abs(int i){              /*   abs() は整数の値を返す関数     */
   if (i>=0) return (i);     /*   i が正ならばそのまま返す       */
   else      return (-i);    /*   i が負ならば符号を変えて返す   */
}

main(){
   printf("abs(3)=%d\n",abs(3));
   printf("abs(-3)=%d\n",abs(-3));
}
```

関数は他の関数を呼び出せるだけでなく、自分自身を呼び出すことも可能です。例えば n の階乗（$n! =$

$n*(n-1)*(n-2)*...*3*2*1$）と呼ばれる値は以前 for を利用して計算しましたが、以下のように関数を使っても計算可能です。

```
int fact(int n) {
   if (n>1) return (n*fact(n-1));
      else  return (1);
}

main(){
   printf("6!=%d\n",fact(6));
}
```

このように関数が自分自身を呼び出すことを再帰呼び出しと言います。

9.4.19 プログラムの挿入

次の演習問題は前の演習問題で作成した関数をそのまま利用します。その場合2つのやり方があります。includeを使用する方法と別々にコンパイルしたものを結合する方法です。残念ながら後者の説明は今年度のテキストからは外しました。前回の課題の関数の部分がox1.cに入っているとすると、今日の課題のプログラムの中に、

```
#include "ox1.c"
```

と言う指示を入れておくと、入れた所にox1.cの内容が挿入されます。これまで通常のプログラムの先頭には`#include <stdio.h>`と言う行がありましたが、基本的には全く同じ動作になります。違いは利用者が独自に作ったファイルを挿入する場合にはファイル名を" "で囲み、Cのシステムが持っているファイルを挿入する場合には< >で囲む所だけです。

［演習問題］

1. 前回の入ったox1.cから、先頭の`#include <stdio.h>`、大域変数の宣言、main()の定義の部分を取り除いて保存せよ。

2. 念のためにox2.cと言うファイルに以下の内容を入力し、前回同様に動作することを確認せよ。

   ```
   #include <stdio.h>
   大域変数の宣言
   #include "ox1.c"
   前回のmain()の定義
   ```

3. 次のような改良を行うこと。

   (a) initialize()の後で盤面を表示させるように直す。

   (b) owari()と言う関数を定義する。これはOかXがどこかに3つ並んでいないかどうか調べて、もしOが3つ並んでいれば、「あなたの勝ち」と表示して1を返す。もしXが3つ並んでいれば「あなたの負け」と表示して1を返す。どちらも3つ並んでいない場合には、0を返すような関数である。

   (c) main()に手を加えて、毎回盤面を表示してからowari()を呼び、勝負がついた場合はそこでプログラムが終了するようにする。

   ox2.cには主として、owari()の定義部分とmain()の定義が入ることになる。

9.4.20　式の値

Cにおいては、関数だけでなくプログラムを構成するほとんどの要素が値を持ちます。例えばj=1は1と言う値を持ちます。それをさらに利用することも可能でj=k=1とすると、変数kに1を代入した結果の値1を変数jにも入れることになります。

forの中によく使われるi++に似た物として++iと言うものがあります。どちらも変数iの内容を1増やす働きがありますが、前者の値は1増やす前の変数iの値であり、後者の値は1増やした後の変数iの値になります。

owari()==1の結果は0又は0以外の数になります。条件が成立した場合に0以外になります。ifはこの値によって判断するので実は if (owari()==1) ... はif (owari()) ... と同じ事になります。

逆にこのあたりが災いしてif (ban[1]=1) ... なんて間違えると、ban[1]=1はban[1]の値にかかわらず1になりますので、... の部分が必ず実行されることになります。さらにban[1]の値まで変化するので大きな被害が出ることがありますが、Cでは文法に従った正しい記述とみなされます。

9.4.21　乱数

コンピュータのプログラムは同じデータを与えると同じ結果が得られるのが普通です。しかしゲーム等では、コンピュータ側がいつも同じ手順で仕掛けてくるのでは面白くありません。大抵のプログラミング言語では、呼び出すたびに毎回異なる値を返す関数を用意しています。

Cではrand()と言う関数がそれにあたります。rand()は呼び出す度に0から約21 億迄の適当な値を返します。通常はもう少し狭い範囲の数が必要となりますので工夫が必要になります。例えばさいころの目の代わりをさせるには、1から6 までで十分です。以下はさいころを10回振るプログラムです。

```
#include <stdio.h>

main(){
   int  i;

   for (i=0;i<10;i++)
      printf("%d ",(rand()%6)+1);  /* a % b でaをbで割った余りが求まる  */
   printf("\n");
}
```

このようにrand()の値を6で割って余りを求めると0から5までの数になるので、1を加えると1から6までの値になります。本当にでたらめな数列の事を乱数と呼びますが、このrand()の返す値は内部でそれらしくなるように計算した値なので疑似乱数と呼ばれます。実際上記のプログラムを実行すると、それらしき1から6までの数が10個出ますが、もう一回プログラムを実行するとまた同じ物が出てきます。

これでは困ることが多いので、通常はrand()の計算の元になる数を設定する関数srand()を利用します。ただこの関数を利用してもそこで設定する値はプログラム起動する度に異なる値を設定しないといけないので面倒です。ここではシステムが実行プログラムごとに設定するプロセス番号を求めるgetpid()と言う関数の結果を利用しています。

```
#include <stdio.h>

main(){
   int  i;

   srand(getpid());  /*  実行時のプロセス番号をsrand()に与える。  */
   for (i=0;i<10;i++) printf("%d ",(rand()%6)+1);
   printf("\n");
}
```

３２５２１６５６１２

３２５４３２３４５４

［演習問題］

1. 前回作成したox2.cをox3.cにコピーせよ。それからox2.cの関数の定義の部分だけ残して保存し、逆にox3.cから関数の定義部分を消去しincludeでox2.cの内容を取り込むようにする。念のためにここでプログラムのの動作確認をする。

2. Xの順番の時に、適当に空いた所にXを入れる関数umeru()を作れ。これはrand()を利用して1から9の値を求めて、対応する位置が空いていればそこにXを入れるものである。srand(getpid());の行はmainの先頭に追加しておく。またmain()を少し直して、Oの番ならばinput(j);を呼び、Xの番ならばumeru()を呼ぶようする。

3. OやXが２つ並んでいる時には、空いた所にXを入れるtomeru()を作れ。tomeru()はXを入れた場合には0を返し、何もしなかった場合には1を返すようにすること。main()もまた少し直して、tomeru()をやってみて、何もなければ先程のumeru()を実行するようにする。

4. umeru()やtomeru()の定義部分はox3.cに入れてox3.cの内容を提出せよ。

［応用問題］

1. 関数umeru()を改良し、中心が空いていればかならず中心を埋め、そうでなく四隅のどれかが空いていれば、その中からランダムに選んで埋め、そうでもなければ残りの中からランダムに選んで埋めるようにせよ。

2. rand()を使わずに人に負けないプログラムにせよ。

9.4.22　ファイルの読み書き (1)

ゲームプログラムなどを除き、有用なプログラムはファイルの読み書きができることが最低必要です。例えば文書が保存できないワープロソフトでは困ります。通常ファイルからプログラムにデータを取り出すことをファイルの読みだし、プログラムからファイルにデータを入れる事をファイルへの書き込みと言います。ファイルにデータを入れることにより、コンピュータの電源が切れてもデータを残すこともできますし、フロッピー等を使用してデータの交換も可能になります。ここではファイルの読み書きの基本を説明します。

ファイルからデータを読み込むときには、

1. fopen()を実行します。この時にファイルを読むのか書くのかの指定もします。通常1つのファイルに同時に読み書きを行うことはしません。正常にファイルを開くことができるとこの関数はファイルディスクリプタへのポインターを返します。(ファイルディスクリプターはファイルに関する様々なパラメタを記憶している所の事です。そのポインターとはそれを記憶している場所を示すものの事です。)ファイルを操作する関数にはこのポインターの値が必要です。

2. ファイルからデータを読む場合にはfscanf()等を使います。これはscanf()とほぼ同じ動作ですが、キーボードから読み込む代わりにファイルから読み込んでくれます。

3. 全てのデータを読み込んだ後は、fclose()を必ず実行します。

次は数値を１つdataと言う名前のファイルから読み込むプログラムの例です。

```
#include <stdio.h>

main(){
   FILE *fp;     /*   ファイルディスクリプターへのポインターを入れる変数の宣言      */
   int i;

   fp=fopen("data","r");           /*   ファイル名(data)と読み出し(r)           */
   fscanf(fp,"%d",&i);             /*   最初にfpが入るだけで後はscanf()と同じ  */
   printf("Read Data is %d.\n",i); /*   ファイルから読みだした値を画面に表示   */
   fclose(fp);                     /*   使い終わったら必ずこれを呼ぶこと         */
}
```

逆にファイルへデータを書き込むときには、fopen()でファイル名と書き込みを指定して、fprintf()等でデータを書き込み、最後にfclose()を行います。以下はキーボードから入力した数値をdataと言うファイルに書き込むプログラムです。

```
#include <stdio.h>

main(){
   FILE *fp;
   int i;

   fp=fopen("data","w");    /*   ファイル名(data)と書き込み(w)          */
   printf("Number = ");scanf("%d",&i);
   fprintf(fp,"%d\n",i);    /*   最初にfpが入るだけで後はprintf()と同じ  */
   fclose(fp);
}
```

ファイルを読み込もうとしたが指定したファイルが存在しない場合には、fopenはNULLと言う特殊な値を返します。またファイルに書き込もうとしたが何等かの理由でできない場合も同様です。

［演習問題］

1. 実行する度に1つずつ大きな値を表示するプログラムを作れ。(count.c)
2. 指定したファイルが存在すればyes、存在しなければnoと答えるプログラムを作れ。(aru.c)

```
File Name = count.c
Yes
```

9.4.23　ファイルの読み書き (2)

ファイルへの出力に関しては、プログラムが出力したいだけデータを出力するで済みますが、入力の場合で特にデータの量が既知でない場合は、まだファイルにデータが残っているかどうか調べながら読むことになります。例えばfscanf()は正しく読めなかった場合-1を返すので、そうでなければファイルからデータが読めたことになります。

```
#include <stdio.h>

main(){
  FILE *fp;
  char line[80];

  fp=fopen("test.c","r");
  while (fscanf(fp,"%s",line)!=-1) puts(line);
  fclose(fp);
}
```

またファイルから1行丸ごと読み込みたい時にはfgetsを用いますが、この関数はファイルの終わりまで読んでしまってもうデータが無い場合にはNULLと言う値を返します。また読み込んだ文字列の終わりには'\n'が付くので注意が必要です。

```
#include <stdio.h>

main(){
  FILE *fp;
  char line[80];

  fp=fopen("test.c","r");
  while (fgets(line,80,fp)!=NULL) printf("%s",line); /* 80は配列の大きさ */
  fclose(fp);
}
```

ファイルにデータをどんどん追加して行きたい場合には、fopen()の際に"w"のかわりに"a"を指定すると、以前出力した結果の後にこれから出力する結果を追加することができます。逆に言えば、"w"を指定した場合には以前ファイルに出力したデータは全て消去されます。

［演習問題］

1. 指定したファイルの行数を数えるプログラムを作れ。(line.c)

   ```
   File Name = test.c△
   There are 5 lines.
   ```

2. 指定したファイルの先頭の5行を表示するプログラムを作れ。ただし元々内容が5行よりも少ないファイルに対しても適切に対応できるようにせよ。(head.c)

3. 指定したファイルの最後の5行を表示するプログラムを作れ。ただし元々内容が5行よりも少ないファイルに対しても、行数が非常に多いファイルにも適切に対応できるようにせよ。(tail.c)

10　ホームページの作成

ここでは、ワークステーション上に自分のホームページを作るためのやり方を説明します。現在学内のいくつかのワークステーションでは各利用者が自分のホームページを作成する事が可能になっています。そしてそれは学内だけでなく、学外(もちろん海外も含む)からも見ることができるようになっています。どのような内容の物を作るのも自由ですし、通常は関係者以外はURLを知らないので大丈夫ですが、他の人にも宣伝するのならば、それなりに恥ずかしくないホームページを作ってからにしてください。

簡単なページを作成するだけならば1時間もかかりません。高度な事を行おうとすると、それなりに学ばなければならないことが多くなりますが、むしろ内容を工夫することに勤める方が良いと感じています。

10.1　準備作業

自分で作って自分で見るだけならばどこのディレクトリーに作成しても良いのですが、後に出てくるcgi機能を使ったり、他の人からも見えるようにするためには、自分のホームディレクトリー(loginした時に居るディレクトリー)の下のwwwと言うディレクトリーの中に、関連するファイルを入れる必要があります。このディレクトリーは通常存在しないので作らなければなりません。また、このディレクトリーが他の人から見えるようにしなければなりません。以上の作業をするためには、次のようなコマンドを実行してください。

```
[miwako]% w3setup△
```

このw3setupの実行は毎回何か作業をする前に必要です。w3setupは必要ディレクトリーの作成だけでなく、ディレクトリーの移動など様々な設定の変更をするからです。[miwako]%はmiwakoさんの場合で、通常は自分のユーザ名が[ ]の中に表示されます。

せっかく色々作成したが、もう公開はやめようと言う時には、

```
[miwako]% w3close△
```

を実行して下さい。作成したファイルは消えませんが、他の人からは見ることができなくなります。

10.2　ファイル名とURL

wwwの下にaaa.htmlと言うファイルを作成した場合のURLは、

```
http://cc01.center.sugiyama-u.ac.jp/~miwako/aaa.html
```

になります。miwakoの部分は各自のユーザー名になります。このURLならば海外からでもアクセスできますが、知らなければちょっと思い付かないでしょう。

index.htmlと言う名前のファイルを作ると、このファイルのURLは少し短くなって次のようになります。

```
http://cc01.center.sugiyama-u.ac.jp/~miwako
```

短いほうが伝えるのにも楽ですので、公開する場合にはindex.html と言うファイルを使うと良いでしょう。

aaa.htmlをテストをする際に毎度このURLを入力するのは大変です。自分のファイルならば単に、

```
aaa.html
```

でも可能ですし、同じワークステーションのmasamiの物ならば、

```
http:/~masami/aaa.html
```

でアクセスできます。

10.3 HTML 入門

WWW のサーバーはクライアント (利用者) からの要求に従ってデータを送ります。クライアントはもらったデータを表示するのですが、そのデータは HTML (HyperText Markup Language) と言う言語で記述されています。WWW は予め用意してあったデータを送るだけなので、自分のデータを公開したい場合には、自分のデータを HTML で記述する必要があります。ここではその概要を説明します。

10.3.1 簡単な例

例えば次のような内容を、aaa.html と言うファイルに入力してみましょう。

```
<TITLE>簡単な例</TITLE>
<H1>これはレベル１の見出し</H1>
HTML の世界へようこそ。
これは１番目の段落です。<P>
そしてこれは２番目の段落です。<P>
```

そして次のようにしてこれを見ることができます。

```
[miwako]%lynx aaa.html△
```

lynx で見る場合には入力したものとあまり変わらないのではないかと言う印象を受けるでしょうが、Netscape 等で見ると、見出しの所の字の大きさが大きくなっています。

HTML は文章中に様々なマークアップタグ (markup tags) を挿入して様々な指示を行います。この例では、< >で囲まれた部分がそうです。< の次にはタグ名が続きます。これは大文字と小文字の区別は無いので、<TITLE>の代わりに<title>と書いても構いません。タグ名が／で始まっているのは有効範囲の終わりを示します。</XXXX>は<XXXX>の終わりを示しています。通常のタグは全て終わりのタグと対になって使われますが、例外も幾つかあります。段落の切れ目を示す<P>等がその例です。

10.3.2 基本タグ

ここでは、先程の例にも出てきた基本的なタグについて説明します。

- 表題：文章の表題を示すものです。通常本文とは別の場所に表示されます。また索引などの見出しに使われることもありますので、文章の内容を的確に示すものが望まれます。タグの形式は次のようなものです。

  <TITLE>表題の文</TITLE>

- 見出し：HTML は、1 から 6 までの 6 つのレベルの見出しが可能で、レベル 1 が一番大きな見出しになります。見出しとして指定された文は独立した左詰めの行として表示されます。タグの形式は次のようなものです。ただし、y の所は実際は 1〜6 の数字になります。

  <Hy>見出しの文</Hy>

- 段落：何もタグの付いていない文章は、クライアント側の都合 (通常画面の幅) に合わせて詰め込まれます。段落として独立させたい場合には、段落の切れ目に次のようなタグを付ける必要があります。

  文文文... 文<P>

- 番号なしリスト：この説明文のような●が先頭に付いた箇条書きをするためには次のようなタグを使います。

```
<UL>
<LI>文章 1
<LI>文章 2
</UL>
```

・ 文章 1
・ 文章 2

<LI>が●になる感じです。<LI>の部分は幾つでも構いません。また3重までの入れ子にすることも可能です。

- 番号付きリスト：先頭に1、2、3と数字が順番に付いた箇条書きをするためには次のようなタグを使います。

```
<OL>
<LI>文章 1
<LI>文章 2
</OL>
```

1. 文章 1
2. 文章 2

今度は<LI>が数字になる感じです。<LI>の部分は幾つでも構いません。また3重までの入れ子にすることも可能です。

- 定義型リスト：言葉ではちょっと説明しがたいものですが、次のような形にしたいときにこれを用います。

電子計算機
　　コンピュータのこと。
コンピュータ
　　かつて電子計算機と呼ばれたもの。パソコンの項を参照のこと。

これは、次のような3種類のタグを使って記述します。

```
<DL>
<DT>電子計算機
<DD>コンピュータのこと。
<DT>コンピュータ
<DD>かつて電子計算機と呼ばれたもの。パソコンの項を参照のこと。
</DL>
```

- 引用文：引用などで通常の文章よりも行頭が右に凹んだ文章を記述したいときには、次のタグを使います。

```
<BLOCKQUOTE>
文文文...文
</BLOCKQUOTE>
```

- 整形済み文章：既に整形が終わっていると言う事で、次のタグで囲まれた文章は入力したままの形で表示されます。

```
<PRE>
                +- 食品栄養学科
生活科学部----+- 生活環境学科
                +- 生活社会科学科
</PRE>
```

この場合画面に表示されるのは、<PRE>のタグが無いだけで後は全く同じものです。

- 強制改行：段落を示すタグ<P>を使用すると段落の間に空行が入ります。それを避けたい場合には、次のようなタグを使います。

  文文文...文

- 水平線：画面一杯の水平線を引くタグは次のようなものです。

  <HR>

- コメント：文章の説明的なもので、表示されては困るものは次のようなタグを付けておきます。

  <!-- 文文...文 -->

10.3.3 文字の形式について

通常の文字はそのままですが、< > & "の4文字は特殊な意味を持つためにそのまま使えません。それぞれ次のような形で記述します。

< → &lt
> → &gt
& → &amp
" → &quot

次のようなタグを付けることにより論理的な意味付けを文字に与えることが可能です。異なる意味付けのものはクライアントで色や書体の違いとして表示されます。

<DFN>定義された語</DFN>	通常イタリックで表示される。
<EM>強調された語</EM>	通常イタリックで表示される。
<CITE>本等の表題</CITE>	通常イタリックで表示される。
<CODE>プログラムなど</CODE>	通常等幅文字で表示される。
<KBD>キーボード のキー</KBD>	通常等幅の太字で表示される。
<SAMP>コンピュータの状態</SAMP>	通常等幅文字で表示される。
<STRONG>強調された語</STRONG>	通常太字で表示される。
<VAR>変数など</VAR>	通常イタリックで表示される。

また直接次のように直接字体を指定することも可能です。

<B>太字</B>
<I>イタリック</I>
<TT>等幅文字</TT>

10.3.4 リンクの付け方

他の文章へのリンクや同じ文章内にリンクを張ることができます。利用者はそこを指定するだけでそのリンクを張った先を見ることができます。このリンクが可能である点がハイパーテキストの由来だと言われています。

- 他文章へのリンク：これは次のような形式のタグを用います。

  ```
  <A HREF="URL">クリックされる文</A>
  ```

  URLの部分には実際にリンクする先の文章のURLが入ります。「クリックされる文」の所はlynxならば選択されたときに反転表示されたり、Netscape等では色が異なる表示がなされます。ここはリンク先が判るような文にします。実際は例えば次の様な形になります。

  ```
  <A HREF="http://www.sugiyama-u.ac.jp">椙山女学園大学のホームページ</A>
  <A HREF="betu.html">同じディレクトリーにあるbetu.htmlと言うファイル</A>
  ```

- 文書内へのリンク：予め次の様なタグ(アンカー)を入れておくと、そこへ行くリンクを張ることが可能です。

  文...文<A NAME="namae">文</A>文...文

  namaeは適当な語を使います。同じファイル中で同じ語は使えません。そしてリンクを張るときには次の様にします。

  ```
  <A HREF="#namae">クリックされる文</A>
  ```

  要するに先程指定した語の前に#を付けます。長めの文章で先頭の所に目次や索引を設けて、そこから後に続く文章の該当するところへリンクを張ると言う形でよく使われます。

この両者を同時に使う事も可能です。つまり他文書の中で予めアンカーを指定しておけば、リンクを張る側は、URLの後に#とアンカーで指定した語を書けば良いようになっています。

```
<A HREF="http://cc01.center.sugiyama-u.ac.jp/~mailbase/student/index.html#ss">生社一覧</A>
```

演習問題

以下のような超簡易CAIを作れ。

1. cai1.htmlと言うファイルを作成し、この中には問題文と答えの選択子(4つぐらい)を入れる。正解を選択したらcai2.htmlへ、間違った選択ならばcai3.htmlへ行くようにリンクを張る。
2. cai2.htmlと言うファイルを作成し、正解者に対するメッセージを入れる。また「次の問題」と言う所を作成して、他の人のcai1.htmlへリンクを張る。
3. cai3.htmlと言うファイルを作成し、誤答者に対するメッセージ(ヒントなど)を入れる。また「問題にもどる」と言う所を作成して、cai1.htmlへリンクを張る。
4. lynx cai1.html△と入力してちゃんと文章が表示されるか、リンクが正しく張れたかを確認せよ。

各ファイルでは、<TITLE>なども忘れずに入れること。

10.4　画像の指定の仕方

WWW が他のサービス (メールやニュースなど) と比べて有利な点は文字情報だけでなく、画像や音声情報が扱える点です。しかしこれらの情報は文字情報に比べると扱いが難しいのでこのテキストでは簡単なイメージの扱い方についてのみふれます。

画像の入っているファイルをここでは画像ファイルと呼んでいますが、画像と言っても様々な種類があります。大きく分けるとモノクロとカラーに分かれます。モノクロも単に白と黒の点からなるものと、灰色の点も許すものがあります。カラーにしても同様に何色まで使えるかによっていろいろで、最も少ないもので 8 色から多いものでは 1,600 万色まであります。

色数の多い画像は非常に大きなファイルになります。よってそのままでは多くの記憶容量を必要とするほかに、通信の際に時間がかかるなどの問題が生じます。そこでこれを圧縮する方法がいろいろ考えられてきました。現在では画像を保存する方式が数十種類ありますが、その中で WWW で利用できるのは数種類です。カラー画像に限定すれば、通常 GIF (Graphics Interchange Format) 形式が標準で JPEG (Joint Photographics Experts Group Bitmap) 方式のものもまず問題なく使えるという状況です。数 cm 角以上の画像では JPEG 方式の方が小さなファイルになります。それ以下の画像では GIF 形式の方が小さくなります。学内で見ているのに関してはあまり問題はありませんが、より多くの人に見てもらう事を考えると、できるだけ容量は少ないのに越した事はなく、画像の大きさにあった方式を選ぶ他に、できるだけ一つのページに大量の画像を利用しないなどの工夫が必要でしょう。

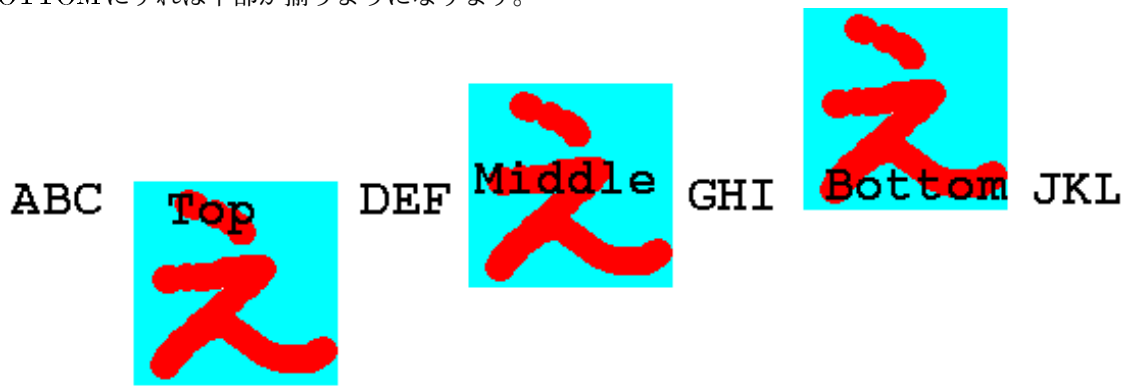
JPEG 方式の画像ファイルの拡張子は jpg に、GIF 形式の画像ファイルの拡張子は gif にします。そして画像を取り込みたい所に、

```
<IMG SRC="画像ファイル名">
```

を挿入します。一つの画像が一つの文字と同様に扱われます。ところが通常画像は文字よりも大きいので前後の文字と画像のどこを合わせるかでかなり違ったものになります。そこで、

```
<IMG SRC="画像ファイル名" ALIGN=TOP>
```

とすると前後の文字に画像の上部が並ぶようになります。TOP の所を MIDDLE にすれば画像の中央が、BOTTOM にすれば下部が揃うようになります。



もちろん通常の端末で画像を見る事ができないので、lynx で画像の指定があるページを見るとその部分が [IMAGE] と置き換えられて表示されます。しかしこれではいったい何の画像があったのか全く解りません。そこで、

```
<IMG SRC="画像ファイル名" ALIGN=TOP ALT="画像の表題">
```

のように指定すると lynx などで見たときに [IMAGE] の代わりにここで指定した表題が出るようになるので、できるだけこの指定を付けるようにして下さい。

同じページの中に表示しなくても良い場合には、

```
<A HREF="画像ファイル名">文</A>
```

も可能です。こうすると「文」を選択すると画像が表示されるようになります。ページ内には小さく縮小したものを入れて、元の大きさを見たい人だけ選択すれば見えるようにすると良いと思います。
また通常のリンクとの組み合わせも可能です。

```
<A HREF="URL"><IMG SRC="画像ファイル名"></A>
```

とすると表示された画像をクリックするとURLで指定したところへ行きます。

10.5 画像の入力と編集

一般的には画像の入力方法には次のようなものがあります。

- 自分でマウス等を利用して入力する。
- 既にある画像をイメージスキャナーで読みとる。
- デジタルカメラで撮影する。
- ビデオ信号からビデオキャプチャー装置で取り込む。

などです。文章の入力でも後での訂正が必ず必要となるように画像の入力だけでは通常済みません。余分な部分を削ったり、全体の色の補正をしたり、拡大・縮小をしたり、ちょっとした画像を作成するのにもかなりの手間がかかります。また文章でも本などからそのまま取ってくると著作権の問題が生じますが、画像の場合も同様な問題があるので注意して下さい。
ここでは機器的な制約より、Windows95のマシンでマウス等を使って画像を作成する方法について説明します。この場合、できた画像をワークステーションに転送する必要があります。ワークステーションで作成すれば転送する必要はありませんが、本体でしか作業ができませんので基礎演習で大勢でやるには不適当のようです。

10.5.1 画像ファイルの作成

Windows95のマシンを起動した後、画面の右下にある「スタート」をクリックします。(左ボタンを押すこと)出てきたメニューの中の「プログラム(P)」をクリックします。さらに「アクセサリ」をクリックして、「ペイント」をクリックすると、Windows95に標準的についてくるお絵描きソフトを起動する事ができます。
この後いろいろ操作をして画像を作成するのですが、説明を書き出すときりが無いので省略します。次の点だけ注意しておきます。

1. まず画像の大きさを設定します。メニューの「変形」の中の「キャンバスの色とサイズ」の所で設定します。後ではめ込むページの事を考えてあまり大きなものにしないようにしましょう。画像を描いた後でも変更は可能ですが初めからよく考えましょう。
2. 塗りつぶすツールで失敗すると悲惨な事になりますが、そういう場合もあわてずに、すぐメニューの「編集」の中の「元に戻す」を利用すればなんとかなります。
3. 画像が完成したらメニューの中の「ファイル」の中の「ファイル名を付けて保存」を選択して、ファイル名を指定した上で保存します。このときどこに保存したか忘れないようにします。

4. パソコンに画像ファイルを置いたままにしないで下さい。断り無く消去する事があります。必ず次項のファイルの転送を行って、無事転送できたら「ごみ箱」アイコンに重ねて消して下さい。

5. 保存する場所としてデスクトップを選択すると、次項の転送の際に困るので、ハードディスク (C:) または自分のフロッピー (A:) 以下に入れて下さい。

10.5.2 画像ファイルの転送

ほぼ同じ手順でパソコンからワークステーションに送るだけでなくパソコンに送る事も可能です。画像を修正するような場合はパソコンに送り返して直す事になります。

1. 画面右下の「スタート」から順番に、「プログラム」、「Ws_ftp」、「WS_FTP95 LE」を選択します。

2. するとSession Profileという画面が出ますのでその中央付近にあるUser IDという所に自分の登録名を、Passwordと言う所に自分のパスワードを入力して「OK」をクリックします。

3. すると画面の左側にパソコンのディレクトリー、右側にワークステーション側のディレクトリーが表示されます。左側を操作して先ほど入力した画像ファイルを選択します。保存した時に指定した名前に.BMPという拡張子が付いていますので注意して下さい。

4. 右側のワークステーション側のディレクトリーをwwwに切り替えてから、[→]をクリックします。複数のファイルも同様に転送できます。

5. 転送が終わったら右下にある「Exit」か右上にある[X]をクリックします。

ワークステーションからパソコンに送るときは、右側のディレクトリーから選択してから逆向きの矢印をクリックします。

10.5.3 ファイル形式の変換

パソコンで作成した画像ファイルはBMP形式と呼ばれるものです。これをGIFまたはJPEG形式に変換する必要があります。画像が小さければ、

```
[miwako]% convert 画像ファイル名.bmp 画像ファイル名.gif△
```

また画像が大きい場合には、

```
[miwako]% convert 画像ファイル名.bmp 画像ファイル名.jpg△
```

とします。

演習問題

前章で作成した超簡易CAIの各ページに適切な画像を追加せよ。

11　会話的なホームページ

前章で説明した方法で作成するページは、全てあらかじめ作られたものを見せるのに過ぎません。利用者側からの入力を受けて、これに応えるようなものの作成法について取り上げます。

11.1　プログラムの作成方法

利用者の入力に応じて異なる応答をするためには、それを実行するプログラムを作成する必要があります。このためにWWWのサーバーには設定したプログラムをサーバー上で起動し、利用者からの入力をそのプログラムに渡し、プログラムからの出力をまた利用者に送り返すと言うような機能があります。これをCGI (Common Gateway Interface)と呼んでいます。

このCGI機能を利用するにはプログラムに様々な条件を満たすことが必要です。

- プログラムの拡張子は.cgiにする。通常のプログラムはコンパイルするとa.outと言う名前になりますが、そのままでは使えません。
- 利用者側からの入力は、パラメタの形か標準入力の形でプログラムに渡されます。このテキストでは後者の方について説明をします。標準入力とは通常はキーボードからの入力ですので、テストをするときはキーボードで入力することで可能です。
- プログラムの標準出力はそのまま利用者側に送られます。利用者側はそれが単なるテキストなのか、HTMLの記号が付加されたものなのか、画像データなのかわからないので、最初にその目印を送ります。通常標準出力は画面ですのでテストの際は画面に正しいものが表示されれば大丈夫です。
- プログラムの実行はサーバーが行います。そのためにファイル等の読み書きを行うプログラムはファイルの許可に注意する必要があります。

実際の簡単な実行例は次のようになります。

1. ng testcgi.c△と入力して、以下のプログラムを入力してから保存、終了する。

```
#include <stdio.h>

main(){
   puts("Content-type: text/html");
   puts("");
   puts("<H1>This is a test for cgi.</H1>");
}
```

2. gcc testcgi.c△と入力してコンパイルする。
3. a.out△と入力して、以下の画面に以下のように出力されれば良い。

```
Content-type: text/html

<H1>This is a test for cgi.</H1>
```

4. mv a.out test.cgi△を入力する。

5. lynx http://cc01.center.sugiyama-u.ac.jp/~miwako/test.cgi △と入力して画面に、

```
This is a test for cgi.
```

と表示されます。cgiプログラムを作成する際には以下のような点に注意します。

- プログラムを修正した場合には再度コンパイルをし、名前を変え、実行できるように設定を変更しなければならないが、名前を変えるところで2回目からは、remove ファイル名?と確認を求められるので、y△と答えます。
- プログラムが別のファイルを読み書きするならば、そのプログラムの設定を変更する必要があります。今そのプログラム名がtest.cgiならば、a.outをこの名前に変更した後で、

  chmod u+s test.cgi△

  を必ず行わないとテストでは動いても正しく動作しません。
- 出力の先頭には必ず“Content-type: text/html”の行と次の空行がなければなりません。WWWのサーバーはこれを読んで以下に続く内容を扱うからです。
- プログラムからページの内容を出力するのではなく、既存のページの内容を参照する様にするには、“Location: URL”と言う行と空行のみを出力します。URLの部分はもちろん参照させたいページのものです。

11.2 入力用ページの作り方

実際の処理を行うのはプログラムですが、通常はプログラムは利用者から何かデータをもらってからそれを処理します。そのもらうためのページの作るために用いるHTMLのコマンド について説明します。

11.2.1 FORM: 形式とプログラムの指定

FORMを使って、入力されたデータを送る先のプログラムの指定とそのときのデータ形式を指定します。

```
<FORM method=POST action="URL">
 .....
</FORM>
```

methodとしてはPOST以外にGETと言うのも利用できますが、GETはPOSTと比べてやや扱いが簡単なものの、入力できるデータ量などに制限があるので、ここでは説明を省略します。URLの部分には実際に作成したプログラムのURLが入ります。

</FORM>までの部分に以下で説明するボタンや入力欄の定義が入ります。

11.2.2 INPUT: ボタン入力

入力が全て終わって、プログラムを起動してくれと言うボタンの定義は次のようにします。

```
<INPUT type="submit" value="Send">
```

Sendの部分はボタンの上に書かれる文字列なので、任意のものが可能です。また似たような形ですが、利用者の入力したものを全て消して最初の状態に戻すボタンは次のように定義します。

```
<INPUT type="reset" value="Erase">
```

Eraseの部分はボタンの上に書かれる文字列なので、任意のものが可能です。
利用者が印を付ける形のボタン(チェックボックス)は次のようにします。

```
<INPUT type="checkbox" name="info">
```

この場合はボタンしか出てこないのでこの前後に文章を補って、何のボタンかを示す必要があります。infoはプログラムに結果を返すときの名前になりますので、同じFORMの中で重複しなければinfo以外の適当なものでも構いません。
幾つかの選択の中で一つだけしか選べないボタン(ラジオボタン)には、次のようにします。

```
<INPUT type="radio" name="service" value="ip">
<INPUT type="radio" name="service" value="uucp">
<INPUT type="radio" name="service" value="ppp">
```

name=で指定しているものが同じもの同士が一つのグループとなり、この中では一つだけしか選択できなくなります。そしてこれはプログラムに返すときの名前になりますので、service以外の適当なものでも構いません。ip、uucp、pppはそれぞれのボタンが選択されたときにプログラムに返される値です。これも識別さえできれば適当な値で構いません。

11.2.3 INPUT: 短い文字列の入力

短い文字列の入力もINPUTで行えます。長い文章などの入力には次のTEXTAREA を用います。

```
<INPUT name="realname">
```

realnameはプログラムに結果を返すときの名前になりますので、適当なものでも構いません。
また、次のようにすると利用者が入力した文字が全て*印で置き換えられて表示されるので(もちろんプログラムには入力した文字がそのまま送られる。)パスワード の入力などに使えます。

```
<INPUT type="passwd" name="pass">
```

passの部分は適当なものでも構いません。

11.2.4 INPUT: 定形データを送る

いくつかのページで同じプログラムを利用する場合、プログラム側からどのページから来たものか判断するための情報が必要となります。またWWWでは、サーバーとクライアントは一つのページをやりとりする度に接続を切ってしまうので、プログラムが複数のページを生成するときに何番目までを送ったかを知るためにも使われます。

```
<INPUT type="hidden" name="address" value="miwako">
```

この場合、画面には入力のためのものは何も表示されません。そして後述のように、address=miwakoと言うデータがプログラムに渡されます。

11.2.5 TEXTAREA: 長い文字列の入力

長い文章などを入力するにはTEXTAREAを使います。

```
<TEXTAREA name="bunsyo" rows=10 cols=50>内容</TEXTAREA>
```

この例では縦10行、横50文字の入力領域が確保されます。「内容」の部分はこの入力用領域に予め入るものです。不要ならば省略しても構いません。

11.2.6 SELECT: 選択メニュー

項目を示してその中から一つを選んでもらう方式です[1]。選択をしようとすると項目の一覧が表示されるので、多くの項目でも場所をとりません。

```
<SELECT name="selone">
   <OPTION>寝る
   <OPTION selected>食べる
   <OPTION value="run">走る
</SELECT>
```

この例では、「寝る」、「食べる」、「走る」の3つの中から一つ選ぶ事ができます。<OPTION>の所はいくつでも可能です。selectedは1つだけ指定可能で、この項目には最初から印が付きます。またvalueの指定をすると、プログラムに渡される値はこちらになります。(この例では「走る」の代わりに「run」が渡される。)

なお、複数選択可能な項目とするには、最初の行を次のようにします。

```
<SELECT name="selmul" multiple>
```

この場合にはselectedを複数指定しても構いません。複数選択したときには名前=値と言うものが複数繰り返されますので注意が必要です。

また、画面上で幾つかの項目が最初から表示されるようにしたい場合には、最初の行を次のようにします。

```
<SELECT name="selone" size="3">
```

とすれば、最初から項目が3つ表示されるようになります。項目が3 つ以上ある場合には、残りの項目を表示させるためのスクロールバーが自動的に表示されます。

11.2.7 入力されたデータの表現方法

以上の入力欄に入力されたデータは、FORMの所で指定されたプログラムに次のような形で渡されます。

名前=値&名前=値&...&名前=値\r

名前は今まで説明したタグの中でname=で指定した内容です。そして値が実際に入力された内容になります。名前と値の間には=が、次の名前との間には& が入ります。このデータはプログラムへの標準入力として渡されますが、いくら多くのデータであっても1行です。そして一番最後に\rと言う通常カーソルを行の先頭に戻すと言う意味のコードが付きます。

また値の中に含まれる記号類は全て%の後に16進数のコードとして表現されます。また空白は+になっています。これらの例を以下に示しますが、漢字などはクライアント側で使用している漢字コードの種類によって変わってきますので対応はかなり難しくなります。

今次のようなファイル(例えばinput.html)を作成して実行すると、

```
<FORM method=POST action="show.cgi">
これはTextAreaです。
<TEXTAREA NAME="name-1" ROWS=3 COLS=60>Sample Data</TEXTAREA>
<P>これはInputです。
<input NAME="name-2">
<p><input type="submit" VALUE="Send">
   <input type="reset" VALUE="Erase">
</FORM>
```

[1]現在本学にインストールされているlynxにはFORMの直後にこのSELECTを使用するとエラーで終了すると言うバグがありますのでご注意下さい。

次の様な画面になります。

```
これはTextAreaです。
Sample Data_____________________________________________
--------------------------------------------------------
--------------------------------------------------------


これはInputです。 ____________________

Send Erase
```

これを実行する前にshow.cgiを作成しておく必要があります。例えば、cgiプログラムに渡されたものをそのまま返すプログラムならば次の様になります。

```
#include <stdio.h>

main(){
   char line[300];  /*  十分な大きさの変数にすること */

   gets(line);      /*  FORMからの入力               */
   puts("Content-type : text/html");
   puts("");
   puts(line);      /*  FORMからの入力をそのまま出力する */
}
```

これをコンパイルして前述のinput.htmlを実行して適当なデータを入力してSendボタンを実行すると、

```
name-1=abc%0def%0ghi&name-2=jkl
```

のような表示がされます。

11.3 定型的なプログラムの起動法

FORMを利用すると様々な入力データをプログラムに渡すことが可能ですが、場合によっては、

- 単に指定のプログラムを起動すれば良い程度である。
- プログラムの起動がINPUTで指定したボタンでは嫌だ、普通のリンクと同じような形式にしたい。
- 一つのページで様々なプログラムを起動したい。

のような要求が出ることもあります。そのような場合には、既に述べた`<A href="URL"> ... </A>`でURLで起動したいプログラムを指定する事により解決できます。(ただしプログラム名は前節同様に.cgiで終わっている必要がある。)

さらにプログラム名の後に次のような形でプログラムのパラメタを渡すことが可能です。

```
<A href="プログラムのURL?パラ1+パラ2+パラ3"> ... </A>
```

この場合プログラムに渡されるパラ1からパラ3の引き数は、プログラムのmain の先頭を次のようにしていると、argv[1]からargv[3]までにそれぞれ入ります。

```
main(int argc, char *argv[])
```

引き数の中に空白等が含まれる場合には前節同様にコード化したものを指定します。

11.4 演習問題

1. 呼び出した回数を表示するcount.cgiを作成せよ。

   (例：http://cc01.center.sugiyama-u.ac.jp/~miki/count.cgi)

2. 呼び出す回数を元に毎回異なる運勢を表示するおみくじプログラムを作成せよ。なお運勢の内容は/usr/users/miki/omikujiと言う名前のファイルの中にあるメッセージを利用せよ。

   (例：http://cc01.center.sugiyama-u.ac.jp/~miki/omikuji.cgi)

3. 和文英訳の問題のページを作成せよ。入力された英文が正解と一致するかどうかで判定を下す。

   次の和文を英訳せよ。入力を終えたならO.K.を選択すること。

   「私は少年です。」

   解答 ----------
   O.K.

4. Tic-Tac-Toeゲームの相手をするページを作成せよ。