「プログラミング入門」のための入門
— シェルとエディタでプログラムを書く —

September 2016

大事な心がけ

以下では，たくさんの新しい言葉が出てきます．大事な言葉は**太字**で印刷されています．そのときには決して読み飛ばさず，また解説を聞き流さないでください．たとえば「ディレクトリ」という言葉が出てきたら，そこに赤線を引き，声に出して「ディレクトリ！」とつぶやき，そしてノートに「用語集」のページを作って，書き留めましょう．
私たちは，言葉を知らないで何かを修得することはできません．言葉がわからないと説明は理解できないのです．そして知的な世界では沢山の言葉が使われます．何かを理解するということは，言葉がつながってイメージされるということなのです．プログラミングは高度に知的な作業ですから，言葉があいまいなままで進むことはできないのだと，胸にしっかり刻んで先に進んでください．それでは始めましょう．

■これからやること　Windows にインストールした Cygwin，または Mac の**ターミナル**[*1]の上で Emacs というエディタを使って Ruby のプログラムを作成し，走らせる手順を学びます．まずはターミナル上のシェルの使い方を学びます．

1　シェル操作入門

1.1　シェル (ターミナル) を起動する

図 1　Cygwin の Bash シェル

Cygwin のアイコン

Cygwin をインストールすると，Windows のデスクトップとタスクバーに Cygwin の**ターミナル**のアイコンができます．そのアイコンをクリックしてください．

Mac の場合には，「アプリケーション」の中の「ユーティリティ」にあるターミナルのアイコンを起動します．いつも使うものなので，ドックに登録しておいてください．

すると図 1 のような画面が現れます．Mac では少し違ったデザインになりますが，操作は同じです．この画面を**シェル画面**あるいは**コンソール**などといいます．

図の**プロンプト**の部分は「ここに文字を打ってください！」というコンピュータからの呼びかけです．タイプするとその右側で点滅している**カーソル**の位置に入力がエコーされます．この画面でキーボードから**コマンド**，つまり命令を入力してコンピュータと対話するのです．また，このようにして文字が打たれる行を**コマンドライン**といいます．

1.2　コマンドを打ってみる

シェル画面での操作はキーボードからの入力が基本です．マウスは使いません．コマンドは沢山ありますが，最初にディレクトリ（フォルダ）[*2]関連の操作を行うことを学びます．

まずはプロンプトの後に `ls` と打って，最後に**リターンキー**[*3]を押してください．

[*1] 他の OS たとえば Linux でも同様です．

[*2] フォルダもディレクトリも名前の違いだけで，ファイルやフォルダを入れるための「容器」のことです．

[*3] **リターンキー**というのは，キーボードの右側にある **Enter** と書かれた大きなキー (↵) です．「エンター

~/16:00:20> ls ←ここで↵

すると，図 2 のようにシェル画面に何か表示されるかもしれません．これは今いるディレクトリの中にあるファイルやディレクトリの名前が表示されたのです．

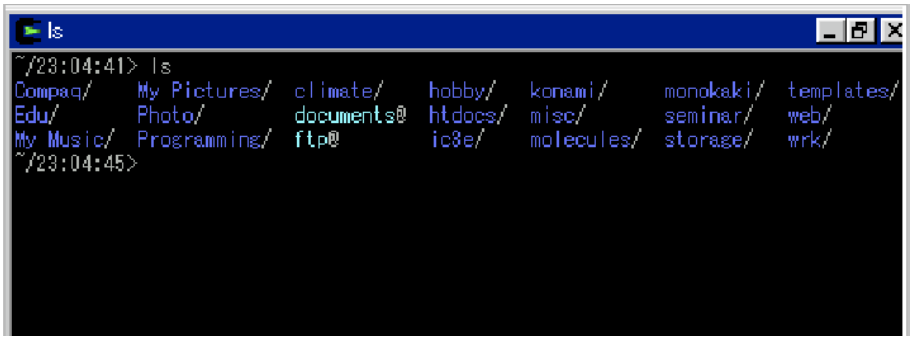


図 2　ls コマンドで表示される画面の例

もしも何も表示されなかったら（最初はたいていそうですが），次のように入力してみましょう．ls **と** -a **の間にスペースがあるのを忘れないように！**

~/16:00:20> ls -a ←ここで↵

今度は，いくつかの項目が表示されるはずです．ここで ls の後ろに付けた -a というのは，ls コマンドのオプションといい，隠しファイルやディレクトリを含めて，すべての (all) ファイルやディレクトリを表示するように指示するためのものです．

■お約束　なお，これ以降，シェル入力におけるプロンプトとリターンキーは省略して，単に次のように表すことにします．**先頭の > はプロンプトなので入力しないこと，最後にリターンキー↵ を押すことを忘れないようにしてください．**

```
> ls -a
```

1.2.1　引数やオプションを伴うコマンド

次のようにコマンドを打ち込んでみましょう．

```
> mkdir abc
> ls
> ls -l
> mv abc xyz
```

キー」ともいいますが，しばしば「改行」の意味で「リターンキー」と呼び習わされています．そもそもこのマークのデザイン自体は「リターン」を意味しているのです．

```
> ls
> rmdir xyz
> ls
```

これらの「かたち」を見てください．mkdir, ls, mv, rmdir はコマンドで，その後ろの abc や xyz は，目的語にあたる**引数 (arguement)** です．

mkdir はディレクトリを作れというコマンドです．この場合，何という名前のディレクトリなのかという目的語がなければ，コンピュータは何をすべきか分からないので，abc という引数が与えられています．

コマンドの中には mv のように引数を 2 つ持つものもあります．実行してみればわかるように，mv は名前を変える目的で使われています．

また，ls -l の "-l" は**オプション**（option）で，コマンドの働きをいろいろと修飾する働きを持っています．

これらのコマンドのまとめは，別紙プリント「簡単 UNIX/Emacs リファレンス」に掲載されているので，それを見ながら操作の意味を理解してください．

CUI と GUI

シェル画面の上でキーボードからコマンドを打つと，コンピュータの返事が返ってくるというやりとりが，UNIX の基本的な**ユーザインタフェース**の流儀です。このようなインタフェースのことを **CUI (Character User Interface)** とよびます。character というのは文字のことです。また，よく見ているようなグラフィックス画面とマウスを使うユーザインタフェースのことを **GUI(Graphical User Interface)** といいます。

2　ディレクトリで作業場所を区切って使う

2.0.1　フォルダとディレクトリ

コンピュータのディスクは，情報の巨大な倉庫で，パソコンでさえも 100 GB(約 1 千億バイト！) を超える容量を持っています．仮に 100 KB (約 10 万バイト) のワードのファイルを考えると， 100 万個のファイルが収納できるわけです．

そのため，ファイルの役割に応じて，ハードディスクの中に「区切り」を設けて仕分けます．この「区切り」を**フォルダ**，あるいは**ディレクトリ**と呼びます．一般に，GUI 環境ではフォルダ，シェルを使った CUI 環境ではディレクトリという使い分けがなされています．

1 台のコンピュータの中には重要なシステムに関わるファイルを収納するもの，アプリケーションプログラムのためのファイルを収納するものなど，他種類のフォルダがあります．それはちょうどお店の売り場や棚のように入れ子[*4]の箱のような関係で配置されています (図 3(a))．

このような入れ子の構造は，ツリーの形を使っても表現することができます．図 3 の (a) と (b) を見比べてください．

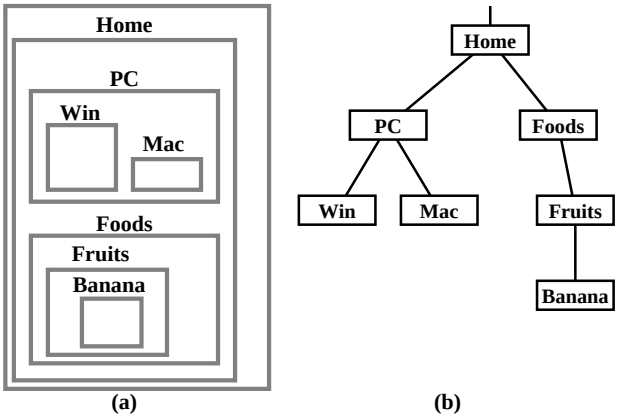


図 3　(a) 入れ子になった箱で表したフォルダの関係 (b) ツリー構造で表したディレクトリの構成

[*4] 「入れ子」というのは箱の中の箱とか，かっこの中のかっこのような包含関係をさす言葉です．ロシアの伝統人形のマトリョーシカも入れ子の構造ですね．

2.1 ホームディレクトリ—ユーザの作業場所の入り口

Cygwin ターミナルなどのシェルを起動した時，そのときに開いている作業場所は**ホームディレクトリ**です．コンピュータの中には無数のディレクトリがあるわけですが，あなたのために開かれているディレクトリはホームディレクトリなのです．

ホームディレクトリには，UNIX ではユーザ名が付けられますが，京都女子大学の授業で学生の Windows ノート PC に設定する環境では，学習環境の統一のために `home` という名前を付けています．以後ホームディレクトリをホームと省略します．

2.1.1 ホームのサブディレクトリを作ってファイルを仕分ける

ホームはいわば玄関口につながった大広間です．そこに荷物を次々に置いたら，数が多くなった時に何がなんだかわからなくなってしまいます．そこで，家の場合には壁を設けて小部屋を作るわけです．

コンピュータの場合には，その目的でホームの下に**サブディレクトリ**を作ります[*5]．

それでは，試しにサブディレクトリを作ってみましょう．

1. Windows の「コンピュータ」，または Mac の「ファインダー」から，ホームディレクトリを開く．ホームディレクトリは，京女設定の Cygwin なら，`C:¥home` であり，Mac なら家の形のアイコンで示されている．
   以後，適宜マウスで操作して，ホームディレクトリ以下にどのような変化が起きるかを観察する．
2. Cygwin のターミナルを起動する．
3. 次のように入力する．入力のあと，どのような表示が返ってくるかをよく見て，その意味を理解する．同時にフォルダの画面の方も注意する．

```
> pwd
> mkdir PC
> ls
> cd PC
> pwd
> ls
> mkdir Mac
> ls
```

[*5] 「サブ (sub)」というのは，英語で「下の」を意味します．地下鉄は Subway ですね．

```
> cd Mac
> pwd
> ls
> cd ..
> mkdir Win
> ls
> cd Win
> pwd
> cd ../..
> pwd
```

4. 別紙プリントにある UNIX のコマンドのまとめを見ながら，上でやっていることを理解しなさい.
5. 最後に次のようにタイプして，上で作ったディレクトリとファイルをすべて消去する.

```
> cd
> rm -rf PC
```

3　エディタでテキストファイルを作成する

エディタ (editor) というのは，テキストファイルを作成したり修正したりするためのツールで，プログラムを書くためになくてはならない道具です．ここでは，標準的なエディタ GNU Emacs(ぐにゅー　いーまっくす) について説明します．

3.1　エディタを起動する

シェル画面から次のように入力してください．最後の `&` の後にはもちろん↵ です．

> **emacs &**

すると，下のような画面が現れます．これが Emacs の開始画面です．場合によっては他の表示になっているかも知れませんが，それでも構いません．どの表示からでも次に進むことはできます．

3.2　ファイルを開く

まず，練習のためにちょっとしたプログラムのソースファイルを作ってみることにしましょう．ファイルの名前は `sample1.rb` とします．このファイルを作成して編集する操作は次の通りです．

`C-x C-f`　**→プロンプトに従ってファイル名を入力**

これだけでは何も分かりませんね．次の説明を見てください．

scratch 画面が出ているところで，次のように操作します．**指定された指使いを守ってキーをタイプしてください．**

1. 左手の小指か薬指で **Ctrl** キーを押しながら **x** のキーを左手中指で打つ．
2. 左手の小指か薬指で **Ctrl** キーを押しながら **f** のキーを左手人差し指で打つ．
3. ミニバッファ (3.4 節参照) に `Find file: ~/` という文字が現れ，カーソルが点滅するので，そのまま `sample1.rb` と打って，リターンキーを打つ．

上の操作では，**Crtl キー** (コントロールキー) という，あまりさわったことのないキーを使っています．

もうひとつ，これとは別に，**メタキー** というのも Emacs でしばしば使います．メタキーは，Windows だと左下にある **Alt キー** (オルタナティブキー) のことで．Mac では **command キー**になります[*6]．

これらのキー操作をそのつど細かく記述するのはめんどうですから，次のように省略した表し方を採用します．

`C-x`　`Ctrl` キーを 押しながら `x` キーを押す
`M-x`　`Alt` キーを 押しながら `x` キーを押す

3.3 プログラムを書いてみる

開いた画面の下側には，`sample1.rb` という表示が出ているはずです．これは「今，`sample1.rb` というファイルを編集中」と言っているわけです．

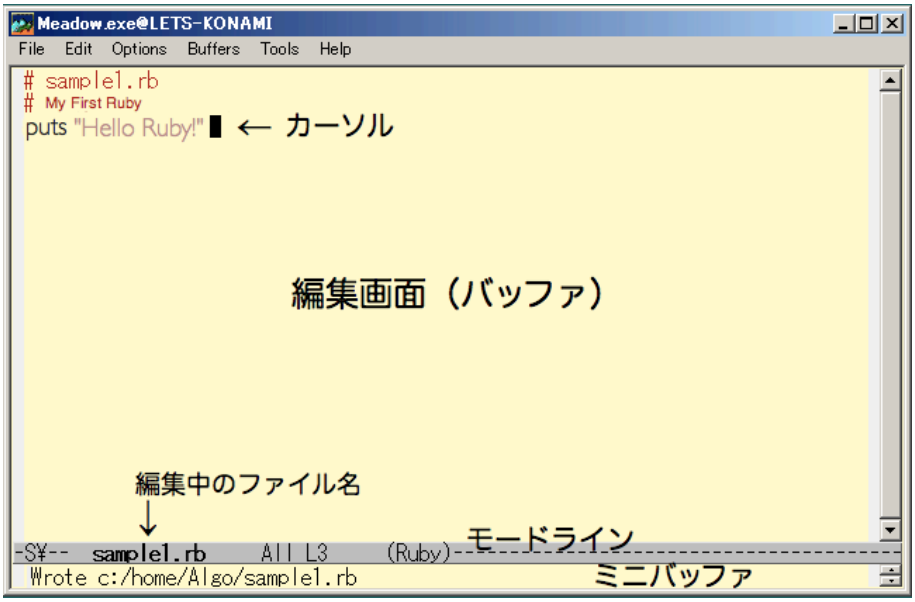
そしたら，以下のようにタイプして書き込んでみて下さい．

```
# sample1.rb
# My first Ruby.
puts "Hello Ruby!"
```

3.4 画面の意味

ここまでの操作の結果，画面がどうなっているかをちょっとくわしく見てみましょう．下の図を見てください．

[*6] 状況によっては `Esc` キーがメタキーとして使われることもあり，その場合の操作は微妙に異なります．



各部の役割は次の通りです．

バッファ　　ファイルの内容を編集するためのメインの画面
ミニバッファ　エディタと人のやり取りを表示
モードライン　編集中のファイル名や行などの情報を表示

3.5　ファイルをセーブ，エディタの終了

編集が終わったら，次のように操作してファイルを保存します．

`C-x C-s`

すると，ミニバッファに，

`Wrote: ~/sample1.rb`

と表示されます．これで無事保存されました．ここで，シェル（Cygwin ターミナル）の中をクリックして，シェルでの作業に移りましょう．

今は `sample1.rb` がセーブされたところのはずなので，まず `ls` コマンドで存在を確かめます．

その後，次のようにタイプすれば，プログラムが走ります．

```
> ruby sample1.rb
```

これで，それらしい文字列が出力されたら，この小さなプログラムは正しく動いています．

この一連の作業が終わったら，次のように Emacs を終了させます．

C-x C-c

ただし，プログラムは何度も書き換えながら書いていくものですから，その時間のすべての作業が終わってから終了させましょう．

3.6 最も基本的な編集操作

本格的なエディタの機能の解説には，本 1 冊分が必要ですが，ここではプログラムを編集するための最も基本的な操作だけを紹介しておきます．

パニックからの脱出！： C-g

画面が変な状態になったり，何かの操作を途中でやめたくなったら，C-g を連打！ミニバッファに Quit と出たら正常に復帰しています．

切り取り： C-k

カーソルを行頭に位置させてから，C-k．これで行末までの文字列が切り取られて，バッファにセーブされます．もう一度 C-k で，改行コードもセーブされます．

これらを繰り返すと，何行にもわたって切り取りながら，バッファにセーブされることになります．

貼り付け： C-y

バッファにセーブされた内容は，C-y でまた画面に吐き出されます．

コピー＆ペースト操作： C-k と C-y を組み合わせます．

コピーしたい領域の先頭にカーソルを位置させて，C-k を必要な回数打ちます．その後 C-y をその場で打って元の状態を復元．→貼り付けたい場所までカーソルを移動させて C-y．

インデントを調整する： [Tab]

ソースのインデント (行頭の空白) を正しく調整するには，その行の任意の位置にカーソルを動かしてから，[**Tab**] キーを押します．これを必要な範囲の上から下まで実行します．

以前の状態に戻す： C-_

いろいろと打ち込んだけど，以前の状態に戻したいときには，C-_を必要な回数だけ打っていくと，次々に昔の状態に戻っていきます．

文字列を検索する C-s

文字列を検索できます．ただしデフォルトの設定では，日本語の検索はできません．

3.7　マウスを使わないで編集作業！

エディタを使いこなすためには，**すべての操作をマウスを使わずにやる**心がけが大切です．ついついマウスを使って画面を移動しないこと！キーボードから手を離さずにカーソルや画面を動かしましょう．

4 Ruby のプログラムを書いてみる

最後に簡単なおみくじソフトを作って Ruby プログラミングの世界を覗いてみましょう.

4.1 エディタでソースを打ち込む

次のプログラムを打ち込んでセーブします．ファイル名は `omikuji.rb` としてください.

```
puts "Omikuji : To quit: Q RET"
while s = gets
  break if s =~ /[Qq]/
  case rand(10)
    when 0
    print "Happiest!"
    when 1 .. 7
    print "Happy"
    else
    print "Awful!"
  end
end
```

ちゃんとセーブされたことを `ls` で確認してください.

4.2 実行する

Ruby はインタプリタなので，そのまま実行できます[*7]．コマンドラインから次のように入力してください.

```
> ruby omikuji.rb
```

すると次のように表示されます.

おみくじプログラム：リターンキーで次のくじ，Q で終了

ここでリターンキーを押してみてください．次々に占いの結果が表示されます．このプログラムの場合，`rand(10)` というのが 0〜9 までの乱数を発生させるので,「大吉」は 10%,「吉」は 70%,「凶」は 20% の確率で現れます．確率を変えたり,「大凶」を追加したりすることも簡単にできます.

[*7] C や Java ではコンパイルという手間がかかります.

4.3 ちょっとしたプログラム

練習のための簡単なプログラムのソースをいくつか紹介します．何が起こるのか，打ち込んで走らせてみてください．

4.3.1 多数回の繰り返し

```
# lovelove.rb
1000.times do
  print "love!"
end
```

プログラムのファイル名は，先頭のコメント行の lovelove.rb とします．エディタで C-x C-f としてからファイル名を入力し，開いた画面にソースを打ち込んでください．他の手順もすでにやったとおりです．

4.3.2 簡単な時計

```
# clock.rb
while true
  puts Time.now
  sleep 1
end
```

このプログラムはいつまでも繰り返しが止まらない無限ループになっています．止めるには Cygwin ターミナルで C-c を入力してください．

4.3.3 簡単な賭け

走らせると，0 か 1 の数を入力するように促されます．コンピュータが予め用意していた数と一致すればあなたの勝ち，外れたら負けで嫌味を言われます．2 以上の数を入力すると注意されます．

```
# nim.rb
my_number = rand(2)
puts "Hi! Guess my number. (0,1)"
print "> "
str = gets
if str.empty?
```

```
  puts "bye"
end
your_number = str.to_i
if your_number > 1 then
  puts "Input 0 or 1!"
elsif your_number == my_number then
  puts "You won!"
else
  puts "Hm, my number is #{my_number}. wwwww"
end
```

4.3.4 誕生日は何曜日？

```
# countdays.rb
wdays = ["Sun","Mon","Tues","Wednes","Thurs","Fri","Satur"]
if ARGV.size < 3
  puts "Input your birthday. YYYY MM DD"
  puts "Example: 1998 11 03"
  exit
end
y,m,d = ARGV.map{|v| v.to_i }
date = Time.local(y,m,d)
nw =  date.wday
puts "You were born on #{wdays[nw]}day."
```

このプログラムは，シェルのコマンドラインで次のように日付を与えます．すると，それが何曜日かを教えてくれます．

```
> ruby countdays.rb 1998 11 3
```