コマンドとコンパイル（数学購読）

４月１８日

1　コマンドについて

ＵＮＩＸシステムは（Windows も Mac も実はそうなのだが）計算機に実行させたい処理や計算機能に対応する「コマンド」と呼ばれる実行ファイルの集まりによって構成されている。

最近はマウスベースでアイコンをダブルクリックして、こうした「コマンド」を実行する。いわゆる市販ソフトウェアももとをたどればコマンドである。

ＵＮＩＸでは歴史的な経緯もあって、「ターミナル」と呼ばれるコマンドを使って、その上で様々な「コマンド」を入力して処理を行うことが多い。

標準的によく用いられるコマンドの一例

- ls (List: ファイルの一覧リストを見る)
- cd（Change Directory:カレントディレクトリを変更する)
- mv（ファイル名の変更・ファイルのフォルダを移動する）
- cp（ファイルをコピーする）
- mkdir（ディレクトリを作成する）
- rm（ファイルを消去する）
- man（オンラインマニュアルを見る）
- emacs（ファイル編集エディタ）

- cc（標準Ｃコンパイラ）
- gcc（GNU　Ｃコンパイラ）

多くの「コマンド」がＵＮＩＸには用意されている。その一つ一つがどのような処理をするかについては、それぞれ使い方が決まっているので、本やネットで調べたりすることでわかる。（通常のソフトウェアに取り扱い説明書があるのと同様である。）

ふつう UNIX の標準的なコマンドにはオンラインマニュアルが内蔵されており man コマンドでそれを見ることができる。たとえば

```
man ls
```

と入力してみれば ls コマンドの使い方がかかれており、それを元に使うことができる。

コマンドに対する一般的注意

1. コマンドには「オプション」がつく。たとえば　ls コマンドにもいろいろなオプションがあって

```
ls -la
```

などそれに応じてコマンドの処理に違いがでる。コマンドにどのようなオプションがあるかも man で確かめることができる。

2. コマンドは標準的に用意されているものや、市販されているコマンド（アプリケーションソフトということもある）を購入して使うもの、さらに自分でコマンドを作ることもできる。（我々がこの購読でやろうとしていることはまさに「コマンドの自作」である）

2　GCC コマンドによるプログラムのコンパイル

CC コマンドやＧＣＣコマンドによって我々は自分たちの思い通りに計算機に処理を行わせる「コマンド」を自作できる

「コマンド」は計算機が実行するモノなので、計算機にしか理解できない方法で記述されている（0 や 1 の羅列）ので、直接 01 を打ち込んでコマンドを作成するのは至難の業である。

そこで人間が理解しやすい言葉（これをプログラミング言語という）でプログラムを作成し、それを計算機が理解できる形式に変換することでコマンドが作成される。(この変換をコンパイルという)

この変換を行うコマンドが「コンパイラ」である。ＵＮＩＸではほとんどすべてのコマンドはＣ言語という言語によって作成されて出来たものである。よってすべてのＵＮＩＸにはＣコンパイラとよばれるコマンド“ cc ”が必ず存在している。

ただ cc は機能的にやや劣る面があるため、通常は GNU というプロジェクトでフリーコマンドとして提供されている gcc コマンドによってプログラムのコンパイルを行う。(我々も gcc を使う)

gcc の基本使い方

Step 1. prog1-1.c の説明

Step 2. emacs などのエディタで作成したＣ言語によるプログラムファイル (ここでは例として prog1-1.c と名付ける）をコンパイルするには次のようにすればよい。

```
gcc prog1-1.c
```

この段階でプログラムファイルに文法エラーがあれば、エラーメッセージがでてコンパイルは中断される。このときはそのエラーメッセージから文法ミスの内容を推定して、エディタで再び prog1-1.c を修正する（これをデバッグという)。

Step 3. エラーもなくコンパイルが終了した場合に ls コマンドでその中身をチェックすると a.out というファイルが新たに作成されていることがわかる。これが gcc によって作成されて「コマンド」である。これはコマンドなので、実際に実行することができる a.out とターミナルに入力してみよう。そうすると prog1-1.c の内容が実行される。

Step 4. しかし、a.out というのはコンパイラが自動的に命名したコマンドなので、プログラムファイルをコンパイルしてもすべて出力は a.out になってしまう。

Step 5. これでは使いにくくて仕方がないので、コンパイラコマンドには出力ファイルを指定するオプションがある。それが-c オプションである。

```
gcc -c prog1-1 prog1-1.c
```

としてコンパイルしてみよう。今度は prog1-1 というコマンドができているはずである。このようにしてコマンド名は自分で指定して作成するのが普通である。

Step 6. gcc コンパイラには他にもたくさんのコンパイルオプションがある。これも man gcc としてオンラインマニュアルで確認すればよい。

Step 7. 次に prog1-2.c の説明

Step 8. prog1-2.c ファイルをコンパイルしてみよう

```
gcc -c prog1-2 prog1-2.c
```

そうするとエラーがでてコンパイルは中断されてしまう。これは数学関数 sin の定義されている場所がわからないということが原因である。このようにC言語の中であらかじめ用意されて用いられる関数は「ライブラリ」ファイルを結合（リンク）することによって完了する。そこで次のようなコンパイルオプションを使う。

```
gcc -c prog1-2 prog1-2.c -lm
```

-lm というのは ライブラリファイルをリンクするというオプション-l と数学関数ライブラリを示す m をくっつけたものである。よって他のライブラリファイルをリンクするときは、それに応じた名称を-l につける必要がある（プログラムの中で使う様々な関数に応じてリンクするライブラリが異なるので、その時々に応じて対応しなければならないが、これ以上のことはここでは学ばない。）

Step 8. 以後残りのファイルをコンパイルして実行してみよう。

- prog1-3.c のコンパイルと実行（プログラムの解説）
- prog1-4.c のコンパイルと実行（プログラムの解説）
- prog1-5.c のコンパイルと実行（プログラムの解説）

3　４月１８日課題

課題１. 関数 $y = \mathrm{e}^x$ の値を区間 $[0, 1]$ を１００等分して得られる点の上で計算し、x,y の順に出力するプログラムを書け

課題２. 関数 $y = \sin x$ の値を区間 $[0, 2\pi]$ で１００等分して得られる点の上で計算し、x,y の順に出力するプログラムを書け。ただし π の値は 4Arctan1 の値を用いよ。

課題３. 関数 $y = \sin x$ の０の周りの三次までのテイラー級数で近似し、その値を $[0, 2\pi]$ の区間を１００等分した点の上で計算し、x,y の順に出力するプログラムを書け。

課題 4. 関数 $z = x^2 + y^2 + 1$ の値を領域 $(x, y) \in [0, 1] \times [0, 1]$ の各区間をそれぞれ２０等分して得られる４００点の格子上で計算し、x,y,z の順に出力するプログラムを書け。

課題 5. 複素関数 $w = z^2$ を領域 $z = \{|\mathrm{Re}z| \leq 1, |\mathrm{Im}z| \leq 1\}$ で実部と虚部をそれぞれ２０等分して得られる格子点の上で計算し、$\mathrm{Re}z, \mathrm{Im}z, \mathrm{Re}w, \mathrm{Im}w$ の順に出力するプログラムを書け。