2 プログラミング実習の準備

プログラミング実習の授業では，LINUX 系 OS である「Ubuntu Linux」を利用する．この 2 章の内容（LINUX のファイルの階層構造や基本的な使い方，エディタの使い方，C のコンパイル，gnuplot の使い方）については既知として授業を進めるので，よく予習しておくこと．

2.1 LINUX の起動

- 起動時に「Ubuntu」を選択して起動する．

起動すると，デフォルトで Windows になる．その場合，再起動する．

2.2 ターミナルの使い方

画面左のメニューから「端末」を選ぶとターミナルが起動する．「インストール済みアプリケーション」から「xterm」「uxterm」を選んでも同じ機能のターミナルである．

2.2.1 ターミナルの基本コマンド

		タイプ例
pwd	現在の作業ディレクトリを表示する	pwd
mkdir	新しくディレクトリを作成.	mkdir dir123
cd	現在の作業ディレクトリを移動.	cd dir123
	cd ~ 自分のホームディレクトリに戻る	cd~
	cd .. 1 つ上のディレクトリへ	cd ..
ls	現在の作業ディレクトリのファイル・ディレクトリを表示.	ls
	ls -F フォルダにスラッシュ付で表示	ls -F
	ls -l 詳しい情報付で表示	ls -l
	ls -a 不可視ファイルも表示	ls -a
cp	ファイルをコピー.	cp file1 file2
mv	ファイル名を変更.	mv file1 file2
	ファイルを移動.	mv file1 dir123
rm	ファイルを削除.	rm file1
	rm *.c ファイル名が「.c」で終わるファイルをすべて削除	rm *.c
	rm -r 下の階層的構造をすべて削除.	rm -r dir123
rmdir	ディレクトリを削除.	rmdir dir123

2.2.2 ターミナルでのパニック対処法

症状 1 ：画面の表示がおかしくなった。キーボードから入力した文字が表示されない。
対処法 **CTRL+q をタイプする。**
理由 UNIX/LINUX では、CTRL+s で表示が止まってしまう。それを解除する。

症状 2 プロンプトが出ない。キーボードからの入力は画面に表示される。
対処法 **CTRL+d をタイプする。**
理由 データ入力待ちのため止まっているので、強制終了させる。

その他: 次の対処法を順に試す。
対処法 1 **CTRL+c をタイプする。** 強制終了させる。
対処法 2 端末エミュレータのウィンドウを閉じる。
対処法 3 ログアウトして再度ログインする。

2.3 エディタ

プログラムを組んだり，単に文章を書いたりするとき，レイアウト機能は不要であるので，遅くて重くて独自ファイル仕様の「高機能ワープロソフト」より，速くて軽くて汎用性のある文字編集のみの「エディタ」を使う．

UNIX/LINUX 系では，Emacs と vi の 2 つのエディタが 2 大勢力となっている．この他，Ubuntu では「インストール済みアプリケーション」にある「テキストエディター」を使っても良い．今後の UNIX/LINUX 利用を考えるなら，Emacs は使いこなせると良い．

2.3.1 Emacs エディタの超基本

- `emacs filename` で，filename ファイルの編集へ．
- `C-` で説明されるコマンドは，control キーとその次のキーを「同時に」押す．
  `M-` で説明されるコマンドは，メタキー（ESC キー）とその次のキーを「順に」押す．
- `C-x C-c` (control と x　及び　control と c) で，emacs 終了．
- `Del` でカーソル直前の 1 文字削除，`C-d` でカーソル位置の文字削除．

Emacs 必須コマンド

C-x C-c	Emacs 終了　（ファイルを保存するかどうか確認を求められるので y/n）
C-x C-w	ファイルを別名で保存
C-x C-s	ファイルを保存
C-x u	UNDO 直前の操作を取り消す（さかのぼって操作を取り消すことができる）
C-g	コマンドラインでコマンド中断
M-<	ファイルの先頭へ
M->	ファイルの最後へ
C-s	単語の検索　順方向
C-r	単語の検索　逆方向
M-%	単語の置換 （置換する語を指定して，yes/no を指定しながら進む．q で終了．!ですべてを置換）
C-h C-h	画面の下半分がヘルプ表示になる（b を選ぶとすべてのキーマッピングを表示）
C-x b	バッファの一覧表示（return で切り替え）
C-x o	画面の上下を移動する（異なるバッファに移動する）
C-x 1	画面の分割を止める（C-x 2　は 2 画面に）
C-x C-f	同時にもう一つファイルを開く
C-k	キル（選んだ部分を消去し，バッファに一時格納） カーソルから行末まで削除．
C-y	ヤンク（キルした部分を復活させる）
C-space	選択領域指定開始 （カーソルを動かし，C-w で選択領域キル．M-w で消去せずバッファ格納）
M-!	サブシェルでコマンド実行

2.3.2 vi エディタの超基本

- `vi filename` で，filename ファイルの編集へ（コマンド入力モードへ）．
- 編集画面で `i` を打つと，上書きモードへ．最近は矢印キーで移動できるようになった．
- 上書きモードで，ESC を打つと，コマンド入力モードへ．コマンド入力モードでは，次のコマンドが使える（他にもたくさんある）．

x	カーソルの箇所を一文字消去．
dd	カーソルのある 1 行を消去．
/	単語の検索語入力．リターンで検索．
:q!	保存せず終了．
:wq	保存して終了．
:w	保存．
:w filename2	別名 filename2 として保存．

2.4 C プログラムのコンパイル

C 言語でプログラムを作成し，コンパイルし，実行する手順は次のようである．

1. プログラムを作成　エディタを利用して作成する．例えば，hello.c とする．
2. コンパイル　ターミナルで，例えば，gcc hello.c とする．プログラムに文法エラーがなければ，a.out という実行ファイルが生成される．実行ファイル名を指定するときは，オプションで追加してコンパイルすればよい．例えば，

   ```
   gcc hello.c -o hello
   ```

3. 実行　実行ファイルが hello であるとき，ターミナルで，

   ```
   ./hello
   ```

2.4.1 サンプルプログラム： calc.c

プログラムを作成して，実行せよ

```
#include <stdio.h>
int main(void)
{
  int a,b,c,d,e,f;
  a= 30;
  b= 4;
  c= a + b;
  d= a * b;
  e= a / b;
  f= a % b;
  printf("a+b= %d\n", c);
  printf("a*b= %d\n", d);
  printf("a/b= %d\n", e);
  printf("a%%b= %d\n", f);
  return 0;
}
```

- 1 行目の「#include ...」は，おまじないである．
- %という演算は，あまりを算出する．
- “　　”の中の「スラッシュ n」 は，改行を意味する．
- “　　”の中の「% d」 は，整数型の出力を意味する．

2.5 数値計算・シミュレーションの基礎

教科書　7.1 章「プログラムを組む方法」を読んでくることを推奨する．

次回の実習では，Euler 法による積分プログラムを配布するが，余力がある者は，冬休み中に自分で作成したり，さらに教科書に紹介しているより高級な積分法を使ったプログラムを作成してみるのも良いだろう．

2.6 gnuplot

gnuplot は，コマンド入力形式のグラフ作成ツールである．簡単に 2 次元や 3 次元のグラフが描けるので，計算結果を確かめるときなどに便利である．無料でダウンロードできる．

2.6.1 起動と終了

- 起動するときは，ターミナルで

  ```
  gnuplot
  ```

- 終了するときは，gnuplot のプロンプトのところで

  ```
  gnuplot> quit
  ```

2.6.2 関数のグラフ

関数をタイプして，グラフを描くことができる．試しに，次のグラフを描いてみよう．

```
gnuplot> plot sin(x)
gnuplot> plot [-pi:pi] sin(x)
gnuplot> plot sin(x), cos(x)
gnuplot> splot [-pi:pi] sin(x)
gnuplot> splot [-2:2] [0:2] exp(-x*x-y*y)
```

2.6.3 データファイルのグラフ

さて，実習では，ファイルに出力されたデータをグラフにする．ファイルがあり，data1 のファイルの中身が右のようになっているとき，

x	y	z
0.0	0.00	10.0
0.1	0.05	9.0
0.2	0.10	8.0
0.3	0.15	7.0
…		

```
gnuplot> plot "data1"
gnuplot> plot "data1" with lines
gnuplot> plot "data1" with linespoints
```

などととすると，1 列目を x 軸に，2 列目を y 軸にしたグラフになる．

```
gnuplot> plot "data1" using 1:3
```

とすると，1 列目を x 軸に，3 列目を y 軸にしたグラフになる．2 つのファイルがあるとき，

```
gnuplot> plot "data1", "data2"
gnuplot> plot "data1" with lines, "data2" with lines
```

などとすれば，両者を重ねて表示できる．
ちなみに，データファイルに空の行があると，2 本目の線になる．

2.6.4 グラフの保存

実習ではグラフを眺めて終了だが，結果のグラフを保存する方法には次のようにする．例えば，eps 形式で，`sample.eps` というファイルとして保存する場合，

```
gnuplot> set terminal postscript eps
gnuplot> set output "sample.eps"
gnuplot> plot "data1", "data2"
```

再び画面表示にするには，

```
gnuplot> set terminal x11
```

とすればよい．

3 C プログラムを用いた微分方程式実習

3.1 ファイルのダウンロードと展開

次の手順に従って，サンプルプログラム 2 つをダウンロードして，自分のディレクトリに準備する．

演習室での授業の初回に行う

1. ターミナルソフトウェアで，この授業専用のディレクトリを作成する．

```
cd ~
mkdir DE
cd DE
```

2. Firefox など，WWW ブラウザを起動し，本授業の web ページを開く．
ポータルサイトから「学習支援サイト (Learning Support Sites)」へ，情報科学部「情報システム学科」へ，そして「真貝」へ，「担当授業に関するページ」をクリックして進む．
あるいは以下の URL を指定する．
`http://www.oit.ac.jp/is/shinkai/lecture/`
さらに「微分方程式」を開き，`DE1.c` と `DE2.c` のファイルを 2 つダウンロードする．（それぞれファイル名を右クリックして「名前をつけてリンク先を保存」する）．
ダウンロードしたファイルは，先ほど作成した DE ディレクトリに入れる．

3. ターミナルで，`ls` して，2 つのファイル（`DE1.c` と `DE2.c`）があることを確認しよう．この 2 つのプログラムファイルは，何度も書き換えるので，不用意に破壊しないように，はじめにコピーをとっておくとよい．

```
cp DE1.c DE1_original.c
cp DE2.c DE2_original.c
```

3.2 基本的な利用方法

いくつかの課題に対して，次のことを行う．

1. プログラムファイル（`DE1.c`, `DE2.c`）を必要に応じて編集する．

`DE1.c`	1 階の微分方程式を Euler 法で解くプログラム．解析解もプロットできる．
`DE2.c`	2 階の微分方程式を Euler 法で解くプログラム．解析解もプロットできる．

2. プログラムをコンパイルする．
`DE1.c` のプログラムをコンパイルするときには，

```
gcc -o DE1.exe DE1.c -lm
```

`-lm` は，math.h をインクルードするためのオプションである．`-o` の直後は生成される実行ファイルの名前になる．

3. プログラムを実行する．
上記のコマンドを用いてコンパイルすると，実行ファイルは，`DE1.exe` になるので，

```
./DE1.exe
```

4. プログラムは，2 つのファイル（`output.numerical`, `output.analytic`）を結果として生成する．

`output.numerical`	Euler 法で解いた結果ファイル．式の入力，初期条件の設定が正しければ，それなりの正しい結果になるはず．
`output.analytic`	解析解ファイル．自分で解いた答えを関数として入力しておき，その数値を出力する．自分の解と入力が正しければ，数値解と一致するはず．

両者を gnuplot でグラフにして，一致しているかどうかを確かめる．gnuplot を開き，

```
gnuplot
gnuplot> plot "output.numerical", "output.analytic"
```

図を点ではなく，線で描くときには

```
gnuplot> plot "output.numerical" with line, "output.analytic" with line
```

あるいは

```
gnuplot> plot "output.numerical" w l, "output.analytic" w l
```

gnuplot を終了するときは

```
gnuplot> quit
```

3.3 DE1.c

プログラムファイル DE1.c である．解読せよ．

どこを書き換えたら良いか，を理解すること．

```
#include <stdio.h>                                  おまじない
#include <math.h>

#define Y0 2.0          /* Initial Value  y(0) */    Y0   は初期値.
#define X0 0.0          /* starting coord x0 */      X0   は始めの x の値.
#define XMAX 10.0       /* ending coord xmax */      XMAX は始めの x の値.

int main(void)
{
  char filename1[] = "output.numerical";
  char filename2[] = "output.analytic";
  FILE *fp1, *fp2;
  double x=0.0, dx=0.01;
  double y=0.0;
  double z=0.0;
  double dydx=0.0;
  // open files
  fp1 = fopen(filename1, "w");
  fp2 = fopen(filename2, "w");
  // x-Loop
  x = X0;
  y = Y0;
  while(x < XMAX){
    // *** Set your problem below
    //     dydx = right-hand side of the 1st order DE
    dydx = -0.5 * cos(x) * y;
    dydx = -0.5 * y + exp(-0.2 * x);              ここに問題となる微分方程式を
    dydx = -0.5 * y + sin(x);                     書き加える．いくつ書いても，
    dydx = -0.5 * y + 1.0;                        いちばん下の行のものが有効に
    dydx = -0.2 * x * y;                          なる．
    dydx = -0.2 * y;
    //     Set your problem  ... end
    // *** Write your analytic solution below
    z = exp(x) * Y0;                              ここに自分の解いた答えを書き
    z = exp(-0.2 * x) * Y0;                       加える．いちばん下の行が有効
    //     Analytic solution ... end              に．
// output
printf("%10.3f %11.5f %11.5f %12.8f \n", x,y,z,y-z);
fprintf(fp1,"%12.5f %12.5f\n", x,y);
fprintf(fp2,"%12.5f %12.5f\n", x,z);
    // Forward Difference
    y += dydx * dx;
// next x
    x += dx;
  } // end of x-loop
  // close files
  fclose(fp1);
  fclose(fp2);
  return 0;
}
```