C 言語の学習
リダイレクト・パイプライン・gnuplot

山本昌志*

2004 年 7 月 7 日

1　本日の学習内容

本日の内容は、以下の通りである。

- UNIX の便利な機能であるリダイレクトとパイプ
- グラフ作成ソフトウェアー gnuplot の使い方と C 言語からの操作

2　リダイレクト

標準入出力 (キーボードとディスプレイ) 先をファイルに変更することをリダイレクトと言う。ディスプレイの出力をファイルに保存したい場合に有効である。

リダイレクトを用いるためには、UNIX のターミナルからの以下のようにコマンドを入力する。ここの `command` は、実行ファイル名や UNIX コマンドのことである。

- 標準出力を hogehoge というファイルに変更する場合 (ファイルは新規作成)

```
command > hogehoge
```

- 標準エラー出力を hogehoge というファイルに変更する場合 (ファイルは新規作成)

```
command 2> hogehoge
```

- 標準入力の変わりに、hogehoge というファイルを使う場合 (ファイルは新規作成)

```
command < hogehoge
```

例えば、カレントディレクトリーのファイルの情報を、file.txt というファイルに記録したければ、

```
ls -l > file.txt
```

とすれば良い。実際に実行してみて、その内容を確認せよ。

*独立行政法人　秋田工業高等専門学校　電気工学科

3 パイプライン

UNIX のコマンドの大部分は、標準入力 (キーボード) からデータを受け取り、標準出力に処理した結果を出力するようになっている。例えば、

```
ls -l
```

などである。このように、コマンドをフィルターと呼ぶ。

複数のコマンドを使って、処理したいデータがある場合、UNIX ではコマンドを接続することができる。標準出力から出てくるデータを次のコマンドの標準入力に渡すのである。例えば、

```
ls -l | sort -n +4
```

のようにするのである。最初のコマンドで、カレントディレクトリーのファイルとディレクトリーの情報を調べ、次のコマンドでファイル容量の順に並べている。

このようにコマンドを連結する機能をパイプラインという。そして、連結する | をパイプという。あたかも、パイプにデータが流れているかのようである。もちろん、2 個以上のコマンドの連結が可能である。

4 gnuplot

4.1 gnuplot とは

本講義のメインテーマである数値計算では、大量の数値を扱うことが多い。いちいち紙に書き写すことは不可能なので、ハードディスクに保存されるのが普通である。ハードディスクに保存されたデータは、適当に処理され、グラフや絵として出力されることが多い。本講義でもグラフを書くことが多々あり、そのプログラムを書く必要がある。グラフィックライブラリーを使うこともできるが、手間がかかる。そこで、グラフ作成ソフトウェアー gunplot を使うことにする。ここでは、その取り扱い方法を述べる。

gnuplot は簡単に 2D、3D のグラフが作成できるフリーのソフトウェーである。単純なグラフから、学術論文用の高品質なグラフまで作成可能で、世界中で使われている。本当の読み方は「ニュープロット」ではあるが、「グニュープロット」と呼ばれることも多い。これは、Free Software Foundation (FSF) が進めている GNU プロジェクト[1]とは関係が無い。

UNIX に限らず、Windows や Machintosh でも動作する。さらに、EXCEL とくらべものにならないくらい美しいグラフを書くことができる。しかも、フリーである。卒業研究のグラフ作成に使うのが良いだろう。

マニュアル類は、web にたくさんある。情報が必要になれば、以下のサイトを調べるのが良いだろう。

http://t16web.lanl.gov/Kawano/gnuplot/

http://lagendra.s.kanazawa-u.ac.jp/ogurisu/manuals/gnuplot-intro/

[1]Unix に似た フリーソフトウェアの完全なオペレーティングシステムの作成を目指す。

4.2 操作方法

簡単な操作方法を述べるが、本当は、先ほど示した web ページを見て各自学習する方が良い。

4.2.1 起動と終了

まずは、gunplot を立ち上げてみよう。以下のコマンドを端末に入力する。

```
$ gnuplot
```

すると、gnuplot が立ち上がり、コマンド入力画面になる。まずは、三角関数のグラフを書いてみよう。以下のコマンドを入力する。

```
gnuplot> plot sin(x)
```

sin 関数のグラフが描けただろう。次に、ヘルプを見たければ、

```
gnuplot> help
```

とする。web ページの方が圧倒的に分かり易いが、ネットに接続されていない環境の場合、このヘルプや役立つ。gnuplot を終了するときには、

```
gnuplot> exit
```

とする。

4.2.2 グラフの描画

以下のようにすると、いろいろなグラフがかける。練習せよ。

$x^3 + x + 1$	`plot x**3+x+1`
$x^{0.5}$	`plot x**0.5`
$\log_e(x)$	`plot log(x)`
$\log_{10}(x)$	`gnuplot> plot log10(x)`
e^x	`gnuplot> plot exp(x)`

3 次元グラフも簡単にかける。3 次元グラフの場合、右マウスでドラッグすると視点を変えることができるのでおもしろい。

$x^2 + y^2$	`splot x**2+y**2`
$x \sin(x + y)$	`splot x*sin(x+y)`

3次元グラフで隠線処理が必要であれば、set hidden3dとする。また、表示するデータ点は、set isosampleで設定する。たとえば、以下のようにすれば、隠線処理し、x 方向と y 方向とも 40 点のデータを出力する。

```
gnuplot> set hidden3d
gnuplot> set isosample 40,40
gnuplot> splot exp(0.5*(-x*x-y*y))*cos(x*x+y*y)
```

4.2.3 ファイルのデータの描画

ファイル処理を学習した時に作成した、三角関数表をグラフにする。表は、各行に θ と $\sin\theta$、$\cos\theta$、$\tan\theta$ の値が書き込まれていたはずである。ファイルになっているデータをグラフ化するときには、plot コマンドを使う。引き続いて、ダブルクォーテーションでファイル名を囲む。最後に、using を使って x 座標と y 座標が書かれている列を示す。具体的には、

```
gnuplot> plot "trifunc.txt" using 1:2
```

とする。各データ点を線で結びたければ、

```
gnuplot> plot "trifunc.txt" using 1:2 with line
```

とする。複数のデータを一度に描くためには、

```
gnuplot> plot "trifunc.txt" using 1:2 with line,
              "trifunc.txt" using 1:3 with line,
              "trifunc.txt" using 1:4 with line
```

とする。ただし、改行しないで (Enter キーを押さない) 記述する必要がある。プロットするレンジを変えたい場合は、set xrange[ymin:ymax] をつかう。

```
gnuplot> set yrange[-1.5:1.5]
gnuplot> plot "trifunc.txt" using 1:2 with line,
              "trifunc.txt" using 1:3 with line,
              "trifunc.txt" using 1:4 with line
```

4.3 gnuplot のコマンド

gnuplot は世界中で使われおり、便利な機能がたくさんある。使い方は、各自調べよ。

4.4 C 言語から gnuplot を操作する

4.4.1 パイプを使う方法

gnuplot を C 言語のプログラム制御するには、パイプを使うのが最も簡単である。C 言語のプログラムで、パイプを開いて、それを gnuplot に接続するのである。後は、C 言語のプログラムが gnuplot を操作するコマンドをパイプに流すのである。

UNIX では、パイプを使うことにより、かなり複雑な動作も簡単に記述できる。そのパイプを開くためには、ファイルポインターが必要である。そのための変数を用意する。パイプの先もファイルとして扱われるのである。

```
FILE *hoge;
```

次に gnuplot を立ち上げて、そこにパイプを接続する必要がある。パイプの情報のファイルポインターで示される。

```
hoge = popen("gnuplot -persist","w");
```

`popen()` 関数がパイプを開く命令である。

パイプを通して、gnuplot にコマンドを送るのは `fprintf()` 関数を使う。

```
fprintf(hoge, "plot sin(x)");
```

終了時には、開いたパイプは閉じるのが礼儀である。

```
pclose(hoge);
```

[**練習 1**] この一連の流れを、C 言語のプログラムで実現せよ。

4.4.2 プログラム例

リスト 1 に gnuplot を C 言語から制御したプログラム例を示す。

リスト 1: パイプを使った gnuplot の制御

```
1  #include <stdio.h>
2  #include <math.h>
3  void mk_triangle_data(char *a, double x1, double x2, int n);
4  void mk_graph(char *f, char *xlb, double x1, double x2,
5                char *ylb, double y1, double y2);
6
7  /*=========================================================*/
8  /*   main function                                         */
9  /*=========================================================*/
10 int main(void){
11
12   double pi = 4*atan(1);
13
14   mk_triangle_data("out.txt", -2*pi, 2*pi, 1000);
15   mk_graph("out.txt", "x", -2*pi, 2*pi, "y", -3, 3);
16
17   return 0;
```

```
18 }
19
20 /*============================================================*/
21 /*   make a data file                                         */
22 /*============================================================*/
23 void mk_triangle_data(char *a, double x1, double x2, int n){
24   double x, dx;
25   double y1, y2, y3;
26   int i;
27   FILE *out;
28
29   dx = (x2-x1)/n;
30
31   out = fopen(a, "w");
32
33   for(i=0; i<=n; i++){
34     x = x1+dx*i;
35     y1 = sin(x);
36     y2 = cos(x);
37     y3 = tan(x);
38
39     fprintf(out, "%e\t%e\t%e\t%e\n", x, y1, y2, y3);
40   }
41
42   fclose(out);
43 }
44
45 /*============================================================*/
46 /*   make a graph                                             */
47 /*============================================================*/
48 void mk_graph(char *f, char *xlb, double x1, double x2,
49               char *ylb, double y1, double y2)
50 {
51
52   FILE *gp;
53
54   gp = popen("gnuplot -persist","w");
55
56   fprintf(gp, "reset\n");
57   fprintf(gp, "set terminal postscript eps color\n");
58   fprintf(gp, "set output \"graph.eps\"\n");
59   fprintf(gp, "set grid\n");
60
61   /* -------  set x axis ---------*/
62
63   fprintf(gp, "set xtics 1\n");
64   fprintf(gp, "set mxtics 10\n");
65   fprintf(gp, "set xlabel \"%s\"\n", xlb);
66   fprintf(gp, "set nologscale x\n");
67   fprintf(gp, "set xrange[%e:%e]\n", x1, x2);
68
69   /* -------  set y axis ---------*/
70
71   fprintf(gp, "set ytics 1\n");
72   fprintf(gp, "set mytics 10\n");
73   fprintf(gp, "set ylabel \"%s\"\n", ylb);
74   fprintf(gp, "set nologscale y\n");
75   fprintf(gp, "set yrange[%e:%e]\n", y1, y2);
76
77         /* -------  plat graphs ---------*/
78
79   fprintf(gp, "plot \"%s\" using 1:2 with line,\
```

```
80                      \"%s\" using 1:3 with line,\
81                      \"%s\" using 1:4 with line\n", f, f, f);
82
83    fprintf(gp, "set terminal x11\n");
84    fprintf(gp, "replot\n");
85
86    pclose(gp);
87  }
```