

Controlling GNUPLOT from C++

Shohei NOBUHARA

初版 2003/02/19

概要

自分のプログラムから GNUPLOT を使うための簡単なクラスをつくる．車輪の再発明そのものなテーマだが，まあ練習ということで．

目次

1	目的	1
2	環境	1
3	方針	1
4	実装	1
5	サンプル	5
5.1	その1	5
5.2	その2	5

1 目的

C++から GNUPLOT[1] をつかってグラフを表示したり，それを eps で保存したり．

2 環境

g++と GNUPLOT が必要．

3 方針

とにかく GNUPLOT を起動し，それにコマンドを渡せばよい [3][2][4]．これには

1. パイプ経由
2. 一時ファイル+子プロセス

の2つの方法が考えられるが，前者は

- "pause -1"のコマンドが効かない？
- オプション"-persist"が無い場合，一瞬表示した後すぐに消える
- "-persist"としても GNUPLOT 本体のプロセスが終わる (?) ので，グラフのマウスによる回転などはできない

後者は

- "pause -1"は有効
- このときグラフのマウスによる回転などはできる
- 一時ファイルを用意する必要がある

と，それぞれ一長一短がある．

いずれにせよ

1. パイプか一時ファイルを開く (popen or fopen)
2. ファイルポインタに書き込む
3. 閉じる (pclose or fclose)

というように，初期化と後かたづけ部分が違うだけなので，

共通の基本クラスからパイプ版とファイル版を派生させる

という方向で実装する．

4 実装

ソースは以下の通り．もちろん宣言と定義をわけてもいいが，大した量でもないのですべてインライン関数にしてしまった方が楽かもしれない．

リスト 1: gnuplot.h

```

#ifndef GNUPLOT_H
#define GNUPLOT_H

#include <stdio>
#include <string>
#include <stdarg>

class Gnuplot_Base
{
protected:

    std::string exec;
    std::string cmdline;

    FILE * fp;

public:
    explicit Gnuplot_Base(const char * execname=NULL) : fp(NULL)
    {
        if(execname)
        {
            exec = execname;
        }
        else
        {
            exec = "gnuplot";
        }
    }

    virtual ~Gnuplot_Base()
    {
        end();
    }

    enum { GNUPLOT_OK = 0,
          GNUPLOT_NG,
          GNUPLOT_ALREADY_OPEN,
          GNUPLOT_NOT_OPEN,
          GNUPLOT_CANNOT_EXEC
    };

    virtual int begin(const char * cmd_opt=NULL, const char * reserved=NULL) = \
        0;
    virtual int end() {return 0;}

    int flush()
    {
        if( fp ) ::fflush(fp);

        return GNUPLOT_OK;
    }

    int begin_data()
    {
        if( ! fp ) return GNUPLOT_NOT_OPEN;
        return flush();
    }

    int end_data()
    {
        if( ! fp ) return GNUPLOT_NOT_OPEN;
        ::fprintf(fp, "e\n");
        return flush();
    }

    int command(const char * cmd, ...)

```

```

{
    if(!fp) return GNUPLOT_NOT_OPEN;
    if(!cmd) return GNUPLOT_OK;

    { va_list ap; va_start(ap, cmd); ::vfprintf(fp, cmd, ap); va_end(ap); }

    return GNUPLOT_OK;
}

int commandln(const char * cmd, ...)
{
    if(!fp) return GNUPLOT_NOT_OPEN;
    if(!cmd) return GNUPLOT_OK;

    { va_list ap; va_start(ap, cmd); ::vfprintf(fp, cmd, ap); va_end(ap); }
    ::fprintf(fp, "\n");

    return GNUPLOT_OK;
}
};

class Gnuplot_Pipe : public Gnuplot_Base
{
public:
    virtual int begin(const char * cmd_opt=NULL, const char * reserved=NULL)
    {
        if(fp) return GNUPLOT_ALREADY_OPEN;

        cmdline = exec;
        if(cmd_opt)
        {
            cmdline += " ";
            cmdline += cmd_opt;
        }

        fp = ::popen( cmdline.c_str(), "w" );
        if( !fp ) return GNUPLOT_CANNOT_EXEC;

        return GNUPLOT_OK;
    }

    virtual int end()
    {
        if( ! fp ) return GNUPLOT_OK;

        ::fflush(fp);
        ::pclose(fp);
        fp = NULL;

        return GNUPLOT_OK;
    }
};

class Gnuplot_Tmpfile : public Gnuplot_Base
{
protected:
    std::string tmpfilename;
    bool remove_file;

public:
    virtual int begin(const char * cmd_opt=NULL, const char * filename=NULL)
    {
        if(fp) return GNUPLOT_ALREADY_OPEN;

```

```

cmdline = exec;
if(cmd_opt)
{
    cmdline += " ";
    cmdline += cmd_opt;
}

if(filename)
{
    fp = ::fopen(filename, "w");
    tmpfilename = filename;
    remove_file = false;
}
else
{
    char buf[] = "/tmp/gnuplot_XXXXXX";
    int fd = ::mkstemp(buf);
    if( fd == -1 ) return GNUPLOT_NG;
    fp = ::fdopen( fd, "w" );
    tmpfilename = buf;
    remove_file = true;
}

if( !fp )
{
    tmpfilename.clear();
    return GNUPLOT_CANNOT_EXEC;
}

return GNUPLOT_OK;
}

virtual int end()
{
    if( ! fp ) return GNUPLOT_OK;

    ::fflush(fp);
    ::fclose(fp);
    fp = NULL;

    ::system( (cmdline + " " + tmpfilename).c_str() );
    if( remove_file ) ::remove( tmpfilename.c_str() );

    return GNUPLOT_OK;
}
};

// use pipe version as default
typedef Gnuplot_Pipe Gnuplot;

#endif //GNUPLOT_H

```

ポイントは、

- パイプを開くときは `popen/pclose` をつかう。このときの引数は実行するコマンドで、返値は `fopen` と同じく `FILE` 構造体へのポインタ。
- 一時ファイルを作るときは `mkstemp` をつかう。返値としてファイルディスクリプタが得られるので、これをさらに `fdopen` に渡すことで `FILE` 構造体へのポインタが得られる。

など。

5 サンプル

5.1 その1

GNU PLOT のバージョンが 3.8 とかでマウスパッチが当たっていれば、以下のように `splot` したときに、マウスでぐるぐる回転できるはず。なおパイプ版を使うと、回転してくれない。

```
#include "gnuplot.h"

int main(int argc, char *argv[])
{
    Gnuplot_Tmpfile gp;

    gp.begin();
    gp.commandln("splot '-' w l");
    gp.begin_data();
    for (double x = -10; x <= 10; x += 1)
    {
        for (double y = -10; y <= 10; y += 1)
        {
            gp.commandln("%f %f %f", x, y, x*x-y*y);
        }
        gp.commandln("");
    }
    gp.end_data();
    gp.commandln("pause -1");
    gp.end();

    return 0;
}
```

なお、

```
gp.begin();
```

を

```
gp.begin("", "a.gp");
```

のようにすると、一時ファイルではなく指定された"a.gp"を使い、`end()` で削除も行わない。

5.2 その2

[3]にあるサンプルを移植したもの。アニメーションする。

```
#include "gnuplot.h"

#include <cmath>
#include <unistd.h>

int main(int argc, char *argv[])
{
    Gnuplot gp;

    gp.begin();
    gp.commandln("set noautoscale x");
    gp.commandln("set noautoscale y");
    gp.commandln("set xrange [-2*pi:2*pi]");
```

```
gp.commandln("set yrange [-2:2]");
gp.commandln("plot sin(x)+sin(x*0.9)");

for (int i = 100; i > 0; i--)
{
    double b = M_PI * i / 5;
    double a = M_PI * 10 - b;

    b += M_PI * 10;
    usleep(10000);
    gp.commandln("set xrange [%5.2f:%5.2f]", a, b);
    gp.commandln("replot");
}

gp.commandln("pause -1");
gp.end();

return 0;
}
```

参考文献

- [1] Thomas Williams, Colin Kelley. GNUPLOT. <http://www.gnuplot.info/>.
- [2] 高橋隆史. C のプログラムから GNUPLOT を動かす方法. <http://tortoise1.math.ryukoku.ac.jp/~takataka/gnuplot/fromC.html>.
- [3] 山内千里. GPTCALL version 0.32. <http://phe.phyas.aichi-edu.ac.jp/~cyamauch/gptcall.html>.
- [4] 松田七美男. C 言語からの呼び出し. <http://ayapin.film.s.dendai.ac.jp/~matuda/Gnuplot/Tips/tips.html#callfrom>.