```
//  file: GnuplotPipe.h
//
//  Header file for the GnuplotPipe C++ class.
//
//  Programmer:  Dick Furnstahl  furnstahl.1@osu.edu
//
//  Revision history:
//      02/06/06  original version, based on gnuplot_pipe
//      02/07/09  minor upgrades
//
//*****************************************************************
#ifndef GNUPLOTPIPE_H
#define GNUPLOTPIPE_H

// include files
#include <iostream>
#include <fstream>
#include <string>
using std::string;

class GnuplotPipe
{
 public:
  GnuplotPipe ( );   // constructor
  ~GnuplotPipe ( );  // destructor

  // accessor functions --- these set private variables
  void set_filename (const string &t_filename) {filename = t_filename;};
  void set_filename2 (const string &t_filename2) {filename2 = t_filename2;};
  void set_title (const string &t_title) {title = t_title;};
  void set_xlabel (const string &t_xlabel) {xlabel = t_xlabel;};
  void set_ylabel (const string &t_ylabel) {ylabel = t_ylabel;};
  void set_plot_title (const string &t_plot_title) {plot_title = t_plot_title;};
  void set_plot_title2 (const string &t_plot_title2)
                        {plot_title2 = t_plot_title2;};
  void set_xmin (const double t_xmin) {xmin = t_xmin;};
  void set_xmax (const double t_xmax) {xmax = t_xmax;};
  void set_ymin (const double t_ymin) {ymin = t_ymin;};
  void set_ymax (const double t_ymax) {ymax = t_ymax;};
  void set_delay (const int t_delay) {delay = t_delay;};

  int init ();                // initialize a plot
  int plot (const double x, const double y);  // plot a point on first line
  int plot2 (const double x, const double y);  // plot a point on second line
  int finish ();              // close down a gnuplot session
  int gnuplot_cmd (const string &plot_cmd);   // send a command to gnuplot

 private:
  FILE *gp_cmd;         // file handle for gnuplot process
  FILE *fileout;        // file handle for temporary output file 1
  string filename;      // file name for temporary output file 1
  FILE *fileout2;       // file handle for temporary output file 2
  string filename2;  // file name for temporary output file 2
  string title;      // overall plot title
  string xlabel;     // x-axis label
  string ylabel;     // y-axis label
  string plot_title;  // title for first line
  string plot_title2;  // title for second line
  double xmin;    // lower x-limit of graph
  double xmax;    // upper x-limit of graph
  double ymin;    // lower y-limit of graph
  double ymax;    // upper y-limit of graph
  int delay;      // plotting delay (in microseconds)
  int plot2_flag;       // flag to indicate whether 2nd plot has started
  string plot_cmd;      // command to plot only the first plot
  string plot_cmd2;     // command to plot both plots
};

#endif
```



```
//  file: GnuplotPipe.cpp
//
//  Definitions for the GnuplotPipe C++ class.
//
//  Programmer:  Dick Furnstahl  furnstahl.1@osu.edu
//
//  Revision history:
//      02/06/06  original version, based on gnuplot_pipe
//      02/07/09  minor upgrades
//      02/14/11  added #include <stdlib.h>
//
//  Notes:
//    * This is still rather kludgey, with ad hoc delays added
//       to make it work.
//
//  To do:
//    * There are many things that could be implemented, including
//       setting the style, outputting to postscript, resetting the plot
//    * Implement separate classes for the curves on an individual
//       plot as well as separate classes for different plot windows
//    * Figure out how to set an appropriate delay adjusted to the
//       speed of the computer
//
//
//*****************************************************************
// include files
#include <cmath>
#include <sstream>
#include <stdlib.h>
using std::ostringstream;

#include "GnuplotPipe.h"

//*****************************************************************

// Constructor for GnuplotPipe (add more)
GnuplotPipe::GnuplotPipe ( )
{

  // Set defaults for filehandles and file names
  gp_cmd = 0;
  fileout = 0;
  fileout2 = 0;
  filename = "gnupipe1.dat";
  filename2 = "gnupipe2.dat";

  // Set titles to blanks
  title = " ";
  xlabel = " ";
  ylabel = " ";
  plot_title = " ";
  plot_title2 = " ";

  // Set min and max defaults to use autoscaling
  xmin = 0.;
  xmax = 0.;
  ymin = 0.;
  ymax = 0.;

  delay = 10000;   // This delay is software/hardware dependent.
                   //  Can we do better?
}

// Copy constructor (needs to be written)

// Destructor for GnuplotPipe
GnuplotPipe::~GnuplotPipe ()
{
   // put an appropriate destructor here
```

```
}


int
GnuplotPipe::init ( )
{
  ostringstream cmd_stream;

  gp_cmd = popen ("gnuplot", "w");  // don't sleep  before running
  if (!gp_cmd)
  {
    std::cout << "Could not open gnuplot! " << std::endl;
    return(1);
  }
  usleep(int(0.5*delay));  // wait a bit to let the gnuplot window open

  fileout = fopen (filename.c_str(), "w");
  fileout2 = fopen (filename2.c_str(), "w");

  gnuplot_cmd ("set timestamp");

  cmd_stream.str ("");
  cmd_stream << "set title \"" << title << "\"";
  gnuplot_cmd (cmd_stream.str());

  cmd_stream.str ("");
  cmd_stream << "set xlabel \"" << xlabel << "\"";
  gnuplot_cmd (cmd_stream.str());

  cmd_stream.str ("");
  cmd_stream << "set ylabel \"" << ylabel << "\"";
  gnuplot_cmd (cmd_stream.str());

  if (xmin == xmax)    // autoscaling condition
  {
    gnuplot_cmd ("set autoscale x");
  }
  else
  {
    cmd_stream.str ("");
    cmd_stream << "set xrange [" << xmin << ":" << xmax << "]";
    gnuplot_cmd (cmd_stream.str());
  }
  if (ymin == ymax)   // autoscaling condition
  {
    gnuplot_cmd ("set autoscale y");
  }
  else
  {
    cmd_stream.str ("");
    cmd_stream << "set yrange [" << ymin << ":" << ymax << "]";
    gnuplot_cmd (cmd_stream.str());
  }

  // this string sets up the plots
  cmd_stream.str ("");
  cmd_stream << "plot \"" << filename
             << "\" using 1:2 title \"" << plot_title
             << "\"";
  plot_cmd = cmd_stream.str();
  cmd_stream.str ("");
  cmd_stream << "plot \"" << filename
             << "\" using 1:2 title \"" << plot_title
             << "\",\"" << filename2
             << "\" using 1:2 title \"" << plot_title2
             << "\"";
  plot_cmd2 = cmd_stream.str();
  plot2_flag = 0;
```



```
  return (0);
}

int
GnuplotPipe::plot (const double x, const double y)
{
  // print the x-y data to a file
  fprintf (fileout, "%e %e\n", x, y);
  fflush (fileout);  // flush the buffer so that gnuplot can read it

  //gnuplot_cmd ("replot");
  if (plot2_flag == 0)
  {
    gnuplot_cmd (plot_cmd);
  }
  else
  {
    gnuplot_cmd (plot_cmd2);
  }
  usleep (delay);

  return (0);
}

int
GnuplotPipe::plot2 (const double x, const double y)
{
  // print the x-y data to a file
  fprintf (fileout2, "%e %e\n", x, y);
  fflush (fileout2);  // flush the buffer so that gnuplot can read it

  //gnuplot_cmd ("replot");
  gnuplot_cmd (plot_cmd2);
  plot2_flag = 1;
  /* usleep (delay); */

  return (0);
}

int
GnuplotPipe::gnuplot_cmd (const string &plot_cmd)
{
  ostringstream cmd_stream;
  cmd_stream << plot_cmd << std::endl;  // add in a return
  // std::cout << "cmd: " << cmd_stream.str() << std::endl;

  fprintf (gp_cmd, cmd_stream.str().c_str());
  fflush (gp_cmd);

  return (0);
}

int
GnuplotPipe::finish ()
{
  pclose (gp_cmd);  // close a gnuplot handle
  fclose (fileout);  // close the first data file
  fclose (fileout2);  // close the second data file

  return (0);
}
```