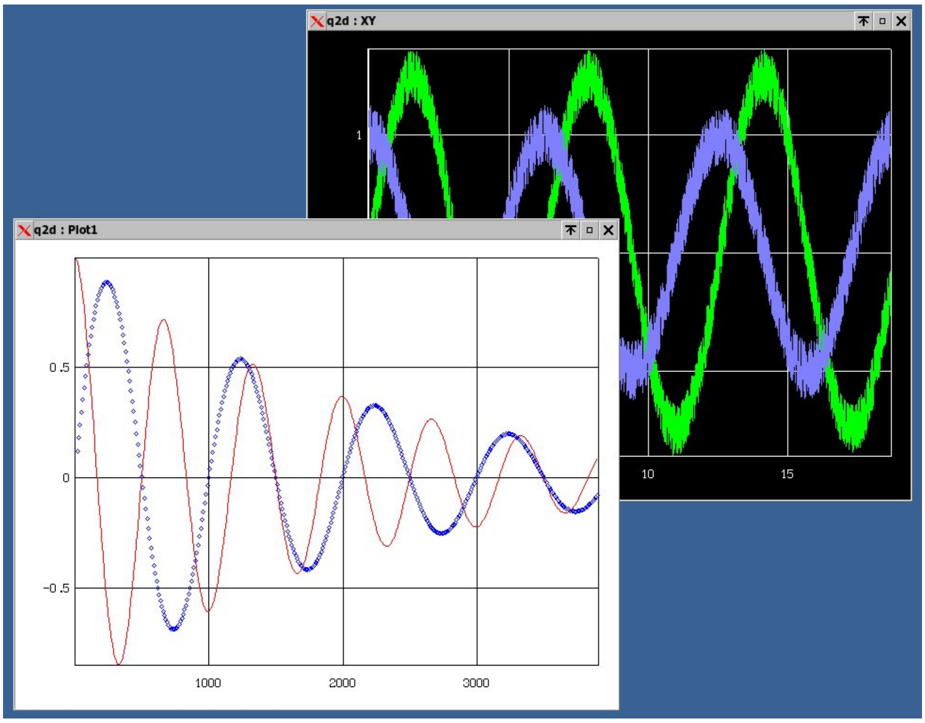

Quick 2d Plot

`Quick2dPlot`, or `q2d` for short, is an open source lightweight plotting program designed for live 2d graphical representation of data streams. The program may be useful for plotting outputs of different software programs, especially in case when you want to see the plot or a number of plots during calculations or a data acquisition process.

The program uses no widgets, it is completely command-driven. Commands and data could be taken from the standard input or one or several files. The user can, during a data plotting process or when it is finished, switch between plots, browse entire plot or any portion of it by shifting, expanding or stretching a visible portion of the plot with help of a few keyboard buttons. The user also can dynamically select one or several curves on the plot which he/she wants to see.

The program is written in C, it uses SDL library for plotting and seems to be reasonably fast. Currently it has been tested under Linux and Cygwin.

`Q2d` deals with objects of three kinds: *variables*, *curves* and *plots*. It has commands for creating, deleting or modifying any of them. *Variables* are actually expandable data buffers. By default, the size of a buffer is unbounded. *Curves* correspond to pairs of *variables*; they have some properties that define how to show them on the screen, such as color, line style, etc. *Plots* are collections of *curves*. They also have properties, such as type of axes and some others.

Data are read into *variables* line by line from the standard input or from a source defined by the argument of a `read` command.

Example of Usage

Suppose, your program `test_prog` calculates some data columns. All that you need to see any 2d plots representing these data, is to print the data line by line to the standard output in your program and prepare a separate file describing plots and curves that you wish to see.

Let us assume that the program `test_prog` calculates three variables t, x and y and prints them to the standard output line by line as follows:

```
<t value> <x value> <y value>
```

The values may be separated by spaces, commas, or semicolons. Suppose, you want see during calculations two plots:

> Plot1 with two curves *x(t)* and *y(t)*
>
> Plot2 with one curve *y(x)*

In this case the command file may contain the following:

```
# Plots One and Two
delay 2
win 800 600
curve first t x
curve second t y
curve third x y
plot One first second
plot Two third
limits One * -1 1
limits Two -1 1 -1 1
read - t x y
```

The text in the command file is almost self-explanatory. The first line is a comment. You may assign any (almost any) names to you variables, curves and plots. In this example variable names are *t, x, y*, the curves are named as *first*, *second*, and *third*, and the plots as *One* and *Two*.

Two `limits` commands define boundaries of initially visible area of plots, *min* and *max* values along horizontal and vertical axes respectively. The star sign (*) indicates that all the data must fit, in our case, for the plot named *One* along the horizontal axis. Here we assume that the variables x and y remain within (-1, 1) interval. If you don't know the intervals at all, `limits` commands may be omitted, and the program will dynamically fit all the data points to the window. As another option, you can adjust visible portion of the plot during or after calculations (see **Keys** section below).

The command `win 800 600` defines the size of the plot window. (800 and 600 are default values, so this line can be omitted.)

The command `delay 2` means that, each time after reading one line of data, `q2d` has to make 2 millisecond delay. Default delay is zero, so, if your program makes calculation fast, without a `delay` command you could see only the final result, not the calculation process.

Command `read` causes the data to be read from the standard input (denoted by symbol '-') into the variables `t, x, y`.

That is all. Save this command file e.g. as `test_prog.q2d`. Now you can run your program like this:

```
./test_prog | q2d test_prog.q2d
```

assuming that `test_prog` and `test_prog.q2d` are in your current directory.

You will see the window with the first plot named *One*. Using arrow keys and arrow keys with `CTRL` key held down, you can shift or stretch a visible part of the plot. Keys `PageUP` and `PageDOWN` allow to switch from one plot to another. Key `HOME` makes plot to fit all calculated data. Key `END` cancels this behaviour.

Some examples can be found under the `examples/` directory, which is `/usr/local/share/q2d/examples` by default. Note that you have to `cd` into a particular example directory before trying to run it.

More Examples

To get help on the command line arguments, start the program with `-h` or `--help` argument. To enter the command line mode, run the program without arguments by typing `q2d` followed by `ENTER` key in Linux command prompt. You will see a command prompt like this:

```
q2d>
```

Now you may enter `help` to inquire commands and keys. You may also enter some commands to see how they work or execute a command file.

To quit interactive mode press `q` (or `@q`, if the program waits for data), followed by `ENTER` key.

The simplest way to quickly test the program is to supply plotting commands as command line arguments, for instance, as follows:

```
echo -e "0\n1\n4\n9\n16\n25\n36\n"|q2d -e "curve # y;plot P C;read - # y"
```

You will see approximation of a parabola graph. Try to use arrow keys and arrow keys with `CTRL` modifier to see different portions of the plot.

List of Commands

All commands shall be in the following format:

```
   <command> [<arg 1> ... <arg n>]
```

Arguments must be space-separated; sometimes the star sign (*) can be used as an argument, often it means that the command assigns or uses default parameters. Any line that begins with a number sign (#) is a comment.

`append <plot> <curve 1> .. <curve n>`
Appends curves to a plot.

`axes (<plot>|*) <style>`
Selects axes style for a plot. A star (*) instead of a plot name means that the command affects default axes style, i.e., all the plots, declared after the command. Available styles are: `grid, boxed, frame, no_axes`

`axes_fmt "(<plot>|*) [<x format> <y format>]\\`Sets or shows a format string for data values along X and Y axes. If the first argument is the star symbol (*), the command affects default format strings, otherwise it deals with format strings for a given plot.
`bufsize <size> (<var 1> .. <var n > | * )`
Defines maximum size of a buffer. A star (*) sign instead of buffer names means that the command affects default value, i.e. influences all subsequent variable declarations. Default is unbounded buffer size (corresponds to zero size).

```
clear [<var 1>..<var n>]
```

Clears variables *var1 .. var n.*

```
colors [<bg color> [<axes color> <curve 1 color> .. <curve n color>]]
```

Selects a color mode (plot background). Three color modes are available: `white` (default), `black` and `custom`. To select white or black mode, only one argument shall be supplied to the command. To define custom color mode it is necessary to supply following colors as arguments: bg color, axes color and some curve colors. Colors can be: `white, black, darkgray, gray, lightgray, darkred, red, lightred, darkgreen, green, lightgreen, darkblue, blue, lightblue, yellow, purple, magenta, cyan, orange, lime, brown, pink`. Colors will be assigned in order of declaration of curves. With no arguments the command shows colors of all declared curves.

```
curve <name> <var x> <var y> [<style> [<mark>]]]
```

Declares curves. Shall be supplied a name of the curve, and names of two variables *x var* and *y var*, which correspond to horizontal and vertical axes. As optional arguments can be supplied *style* and *mark* Available curve styles are: `line`, `point` and `linepoint`. Available marks are: `dot, square, big_sqr, plus, diamond, open_sqr`.

```
del <object name>
```

Deletes an object - a variable, a curve or a plot.

```
delay [<t>]
```

Forces the program to make *t* millisecond delay (which can be fractional) after reading each data line. With no arguments the command shows a current delay value.

```
exec <file 1> .. <file n>
```

Executes command files *file 1 .. file n.*

```
frame_rate <n>
```

Defines how many times per second the current plot will be redrawn.
Default value is 10.

```
help [<command>]
```

Lists all commands, if no argument is supplied, or describes keys or a specific command.

```
limits (<plot>|*) [(<x min> <x max>|*) (<y min> <y max>|*)]
```

Defines horizontal (*x min*, *x max*), and vertical (*y min*, *y max*) boundaries of initially visible portion of a plot. A star sign (*) instead of a pair of values means that *x min*, *x max*, or *y min*, *y max* will be dynamically selected to fit data points. A star sign in place of a plot name makes the command to affect default values, i.e. all subsequent plot declarations. Instead of two stars as 2-nd and 3-d arguments it is acceptable to put one, which means that the size of a visible plot region will fit all data points. This is the default behavior. If the first value in a pair is prefixed with a dollar sign ($), the corresponding axis will be set to a ”fit last” mode, when the window dynamically shifts in order to accommodate all last placed points without changing the size, if possible. Note that no spaces between the dollar sign and the value are allowed. The command without arguments or with one argument, which can be a star (*) or a plot name, shows default limits or limits for a given plot.

`obj`
Prints some information about declared objects.

`plot <name> [<curve 1> .. <curve n>]`
Declares plot containing curves *curve 1 .. curve n.*

`quit, q`
Makes the program to quit.

`read ( . | - |<file>) <var 1>  .. <var n>]`
Reads data from a source indicated by the first argument into variables `var 1 .. var n`. A star (*) instead of a variable name forces the program to skip corresponding data column. If a variable name begins with a number sign (#), read command fills the buffer with data line numbers, starting from 0, instead of reading data into it. A point (.) as the first argument denotes the the current stream or file, a dash (-) denotes the standard input. Besides of data, read command understands any valid command, which must be prefixed with an at sign (`@`). There are although specific commands:

- `@clr` - clears all data buffers used by read command,
- `@abort` - cause read command to return,
- `@view` - stop reading data and go to the plot view mode,
- `@q` - quit the program (or go to the next command file).

`rd_abt_enable`
Enables aborting of data reading via keyboard. Enabled by default.

`rd_abt_disable`
Disables aborting of data reading via keyboard. By default aborting is enabled.

`remove <plot> <curve 1> .. <curve n>`
Removes curves from a plot.

`restart`
Deletes all objects and brings the program into initial state.

`save <plot> [<file>]`
Saves plot to a file in bmp format (not implemented yet).

`step <n>`
Makes the program to read every n-th data line only. Default value is 1.

`var <name 1> .. <name n>]`
Declares variables. Typically use of this command is not necessary, because corresponding objects will be created, if needed, with a curve declaration command.

`win [<width> <height> [<bpp>]]`
Defines size of the window and optionally bits per pixel.
Defaults are *width = 800, height = 600, bpp = 32.*

`write (<file>|-) [<var 1>..<var n>]`
Writes variables to a file or the standard output.

Keyboard Commands

`PAGE UP, PAGE DOWN`
Switch to the next or previous plot.

`UP, DOWN, RIGHT, LEFT`
Move a visible portion of a plot along vertical or horizontal axis.

`'+', '-'`
Expand or stretch a visible portion of the plot.

`CTRL-UP, CTRL-DOWN, CTRL-RIGHT, CTRL-LEFT`
Expand or stretch a visible portion of the plot along vertical or horizontal axes.

`HOME`
Make a visible portion of the plot to fit all data points ("fit all" mode).

`SPACE`
Pause/continue plotting, without data loss.

`ESCAPE`
Exit from a plot preview mode and go to the next command.

`Q, q`
Quit the program.

Please send bugs, comments, and suggestions to `lang21@yandex.ru`