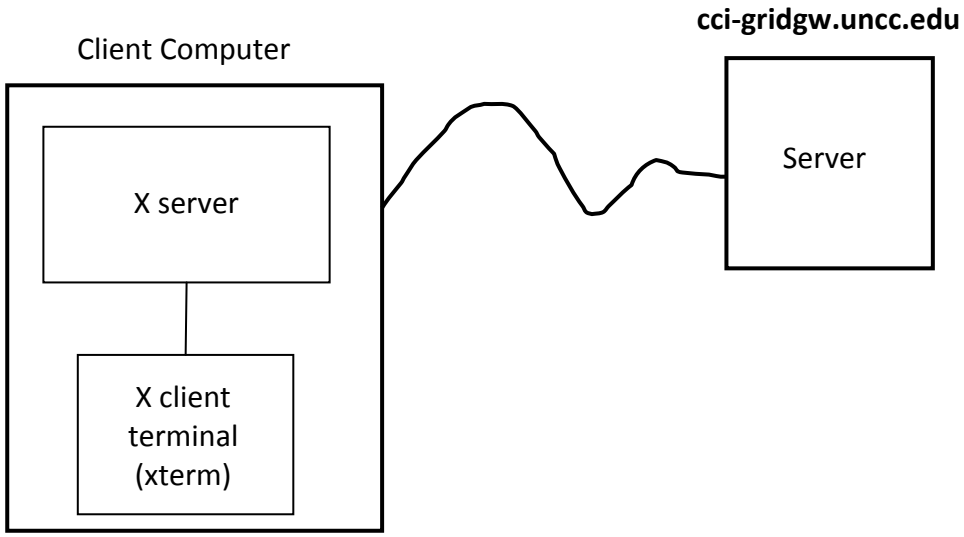
Notes on creating graphical output - X-11 graphics

B. Wilkinson, October 31, 2013

Assignments and projects may require graphical output, which is especially relevant for problems such as the heat distribution problem to show the heat contours or the *N*-body problem to show movement of bodies. When executing programs on a remote server such as **cci-gridgw.uncc.edu**, the graphical output has to be forwarded to the client computer for display. In these notes, we will explain how to create and forward basic X11 graphics.[1] Note these notes only apply to an interactive connection a server (not batch through a job scheduler.)

X-11 graphics

X-11 refers to version 11 of the X Window System first developed in the 1980's. It is chosen because it is part of the Linux distribution, it is relatively easy to write simple graphics, and easy to forward to a client. It is said to be "almost universally" used on Unix-like systems [Wikipedia X Window System]. The ability to forward the graphics is critical in our application. X is a client-server model that relies on an X-server running on the client:



Making the Connection

Windows PC as client

For a Windows PC, it is necessary to get an X server installed on the PC. (A Mac already has this – simply start xterm to create an X client terminal.)

Xming. Perhaps the easiest way to install an X server on a PC at no cost is to install Xming X Window server, which is small and very quick to install. Then use a PuTTY terminal configured to forward X11. X11 forwarding is enabled in PuTTY by first selecting SSH > X11. Check "enable X11 forwarding" with X display location "localhost:0" and then going back to Session and entering the host name. (*Note: DOT NOT load a previously saved host name first as it will not include X11 forwarding. It can be saved afterwards to include X11 forwarding.*) Start the local Xserver before connecting to the remote sever.

[1] Note: Rather than use the basic X 11 libraries directly as described here, one could use other graphics libraries that use X 11 forwarding such as Cairo if the libraries are installed.

Cygwin. Alternatively you can install Cygwin with the X server (Cygwin/X) at no cost and operate the PC as a Linux-like system. This installation takes long time (hours) when you select all the packages. Make sure you select the X11 libraries. Once installed, start the X server and xterm connecting to the X server and login through the xterm. To forward X-11 graphics from the xterm, include the –X option:

```
ssh cci-gridgw.uncc.edu –X –l <username>
```

You will also need to specify your username on the remote server with the –l option if it different to the local PC.

Test X11 forwarding

Test the connection and forwarding by running xclock in the background:

```
xclock &
```

Servers without an external Internet connection

On the UNCC cluster, internal nodes such as **cci-grid05** are not accessible directly and one needs to first ssh into **cci-gridgw.uncc.edu** remembering to forward X11 graphics (-X option) and then ssh from **cci-gridgw.uncc.edu** to the internal node, again remembering to forward X11 graphics, e.g. from **cci-gridgw.uncc.edu** to **cci-grid05**:

```
ssh ccigrid05 –X
```

Test the connection and forwarding by running xclock in the background. The clock graphics should forward back through the two servers and to your client machine. (You would get two clocks if you also forwarded one from the first server.)

Servers that are not accessible from off-campus. A similar procedure is necessary to reach servers such as **coit-grid06.uncc.edu** from off campus.

X-11 drawing code

Before calling any X 11 routine to draw a figure, you have to first do a rather long sequence of code to set up the X window environment, which is given below and you can just copy:

```
...
#include <X11/Xlib.h>          // X11 library headers
#include <X11/Xutil.h>
#include <X11/Xos.h>

#define X_RESN 800             /* x resolution */
#define Y_RESN 800             /* y resolution */
...

int main (int argc, char **argv ) {

...

/* --------------------------- X11 graphics setup ------------------------------ */

  Window         win;                /* initialization for a window */
  unsigned int   width, height,      /* window size */
                 win_x,win_y,        /* window position */
```

```
                border_width,        /* border width in pixels */
                display_width, display_height,      /* size of screen */
                screen;                             /* which screen */

  char         *window_name = "My graphics program", *display_name = NULL;
  GC            gc;
  Unsigned long valuemask = 0;
  XGCValues     values;
  Display       *display;
  XSizeHints    size_hints;
  Pixmap        bitmap;
  XPoint        points[800];
  FILE          *fp, *fopen ();
  char          str[100];

  XSetWindowAttributes attr[1];

  if (  (display = XOpenDisplay (display_name)) == NULL ) {   /* connect to Xserver */
     fprintf (stderr, "Cannot connect to X server %s\n",XDisplayName (display_name) );
  exit (-1);
  }

  screen = DefaultScreen (display);                          /* get screen size */
  display_width = DisplayWidth (display, screen);
  display_height = DisplayHeight (display, screen);

  width = X_RESN;                                            /* set window size */
  height = Y_RESN;
  win_x = 0;   win_y = 0;                                    /* set window position */

  border_width = 4;                                          /* create opaque window */
  win = XCreateSimpleWindow (display, RootWindow (display, screen),
        win_x, win_y, width, height, border_width,
        BlackPixel (display, screen), WhitePixel (display, screen));

  size_hints.flags = USPosition|USSize;
  size_hints.x = win_x;
  size_hints.y = win_y;
  size_hints.width = width;
  size_hints.height = height;
  size_hints.min_width = 300;
  size_hints.min_height = 300;

  XSetNormalHints (display, win, &size_hints);
  XStoreName(display, win, window_name);

  gc = XCreateGC (display, win, valuemask, &values);         /* create graphics context */

  XSetBackground (display, gc, WhitePixel (display, screen));
  XSetForeground (display, gc, BlackPixel (display, screen));
  XSetLineAttributes (display, gc, 1, LineSolid, CapRound, JoinRound);

  attr[0].backing_store = Always;
  attr[0].backing_planes = 1;
  attr[0].backing_pixel = BlackPixel(display, screen);

  XChangeWindowAttributes(display, win, CWBackingStore | CWBackingPlanes | CWBackingPixel, attr);

  XMapWindow (display, win);
  XSync(display, 0);

/* --------------------------- End of X11 graphics setup ----------------------------- */


... /* continue with N-body code */
```

Useful X11 routines

Once the code to set up the X window environment is in place, you get down to the business of drawing an image, using routines such as

```
XClearWindow(display, win);                                  // clear window for next drawing
```

```
XSetForeground(display,gc,(long) color);                    // color of foreground (object to be drawn)
XDrawPoint (display, win, gc, x, y);                        // draw point at location x, y in window
XFillArc (display,win,gc,x,y,width,height,angle1,angle2);   // draw arc/circle
XFlush (display);                                           // necessary to write to display
```

The long integer color is a 24-bit number that specifies the color, as give in the Wikipedia entry for X-11 color names. For example, 0xDC143C would give Crimson. (Note: the number can be given as a hexadecimal number.) To create a circle with **XFillArc()**, the start and end angles would be 0 and 23040 (degrees x 64). You drawing routines can be repeated in a loop to display movement. Include **usleep()** or **sleep()** to get the appropriate speed for the motion.

Compiling C code with X-11 graphics

You will need to compile with the X11 libraries in addition to any other libraries such the Math libraries, e.g. to compile **NbodyG.c**: [2]

cc -o NbodyG NbodyG.c -lm –lX11

On some systems (Macs e.g.), you may need to provide the full path to the X11 libraries:

cc -o NbodyG NbodyG.c -lm -L/usr/X11R6/lib -lX11

Make file

A make file is most convenient especially if the compilation command is getting long. For example, a file called **makefile** with the contents:

Target name, used when invoking make with make **<target>**

Dependencies -- check source file has been updated.
Will not recompile if not necessary.

Command line to execute

```
Nbody: Nbody.c
	cc -o Nbody Nbody.c -lm

NbodyG: NbodyG.c
	cc -o NbodyG NbodyG.c –lm –lX11
```

will compile either a regular C program, **Nbody.c**, or a graphics output version **NbodyG.c** with the commands:

```
make Nbody
make NbodyG
```

Note the commands in the make file, (cc … in the example) MUST begin with a tab character.

Useful references

Wikipedia entry: X-11 color names http://en.wikipedia.org/wiki/X11_color_names
Xming software: http://sourceforge.net/projects/xming/
Xming home page: http://www.straightrunning.com/XmingNotes/
Cygwin home page http://cygwin.com/
XLib Manual http://tronche.com/gui/x/xlib/
X11 graphics routines http://tronche.com/gui/x/xlib/graphics/

[2] Order of libraries on command line is important. Libraries must follow the sourcefile. Symbols are resolved from left to right.