Creating graphical output using X-11 graphics

B. Wilkinson, February 4, 2016

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. These notes only apply to using the provided Ubuntu virtual machine or a native Linux installation and an interactive connection a server (i.e. not a system with a job scheduler).

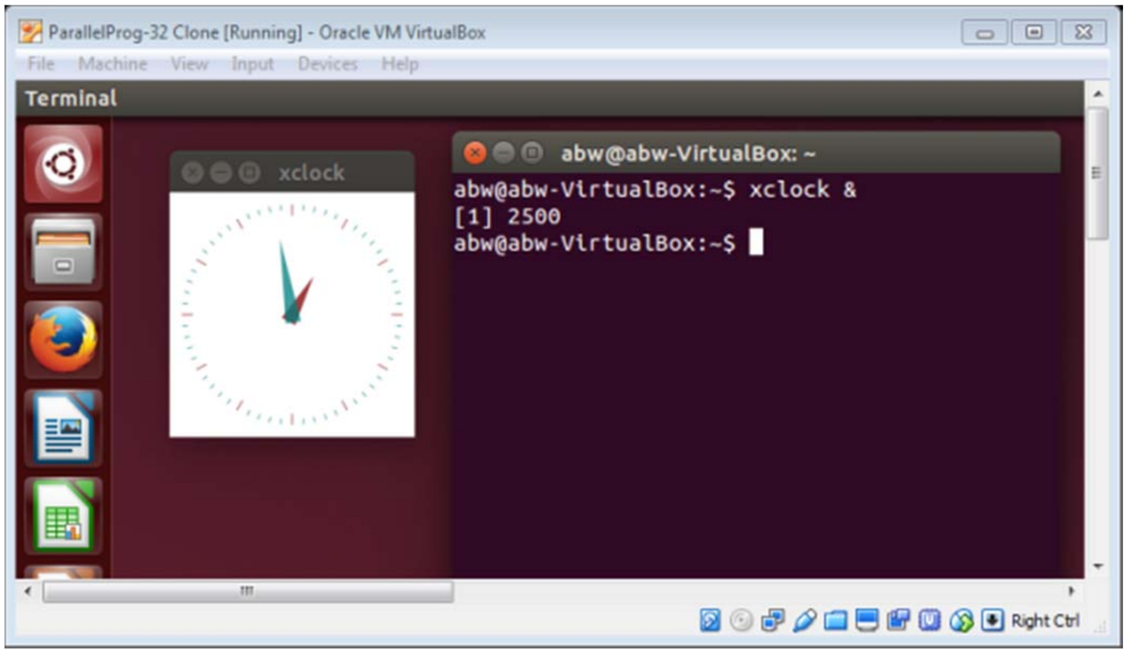
X-11 graphics

X-11 refers to version 11 of the X Window System first developed in the 1980's. It is chosen here because it is part of the Linux distribution, it is relatively easy to write simple graphics, and has server-client design that makes it easy to forward graphics to a client on another computer. An X-server is usually installed and already running if you are using a Linux distribution, including the course VM.

First test your X11 environment by running the X11 clock program in the background with the command[1]:

```
xclock &
```

The **&** specifies to run the command in background so that control is returned to the terminal.


ParallelProg-32 Clone [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal
xclock
abw@abw-VirtualBox: ~
Right Ctrl

```
abw@abw-VirtualBox:~$ xclock &
[1] 2500
abw@abw-VirtualBox:~$
```

[1] In the unlikely event the Xserver is not running issue the **startx** command to start it.

Adding graphics to your program

Before calling any X11 routines, you have to first include a rather long complicated sequence of code to set up the X window environment. To simplify the programming, a header file called **X11Macros.h** is provided, which includes a macro **initX11(x_resn, y_resn)** to insert the X11 code for the X11 environment and set the x and y display resolution. Also in **X11Macros.h** are predefined colors. A sample program, **sample.c**, is given below:

```
#include <stdio.h>
#include <stdlib.h>

#include "X11Macros.h"              // X11 macros
#define X_RESN 800                  // x resolution
#define Y_RESN 800                  // y resolution


int main (int argc, char **argv ) {
   int x,y;
/* --------------------------- X11 graphics setup ------------------------------ */

         initX11(X_RESN,Y_RESN);                  // includes the X11 initialization code

    for (x = 0; x < 800; x = x + 1) {
         y = x;

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

         XSetForeground(display,gc,RED);          // color of foreground (applies to object to be drawn)

         //XDrawPoint (display, win, gc, x, y);   // draw point at location x, y in window

         XFillArc (display,win,gc,x-25,y-25,50,50,0,23040);  // draw circle of size 50x50 at location (x,y)

         XFlush(display);                                    // necessary to write to display

         usleep(100000);                                     // provide a delay beween each drawing

    }

         return 0;
}
```

This program will display red filled in circle that starts at the top left corner (x = 0, y = 0) and moves diagonally downwards. The origin of the display is at the top left corner with x and y across and down respectively. The program can be found in **~/ParallelProg/X11/** directory on the VM. The red statements are needed to set up the X11 environment. The X11 routines shown in the program are:

```
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, not used here
XFillArc (display,win,gc,x,y,width,height,angle1,angle2);   // draw arc/circle
XFlush (display);                                    // necessary to write to display
```

which are probably sufficient for courses assignments although you can find a complete list of available X11 routines on-line.

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 (hexadecimal) would give Crimson. As a convenience, the names BLACK, BLUE, BROWN, CYAN, GRAY, GREEN, MAGENTA, ORANGE, PINK, PURPLE, RED, TURQUOISE, VIOLET, WHITE, YELLOW are defined in **X11Macros.h** as pre-defined colors so that you do not need to use 24-bit integers. 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

Make sure **X11Macros.h** is in the same directory as your source code. Compile your program with the X11 libraries in addition to any other libraries such the Math libraries (-lm), e.g. to compile **sample.c**:

```
cc -o sample sample.c -lm -lX11
```

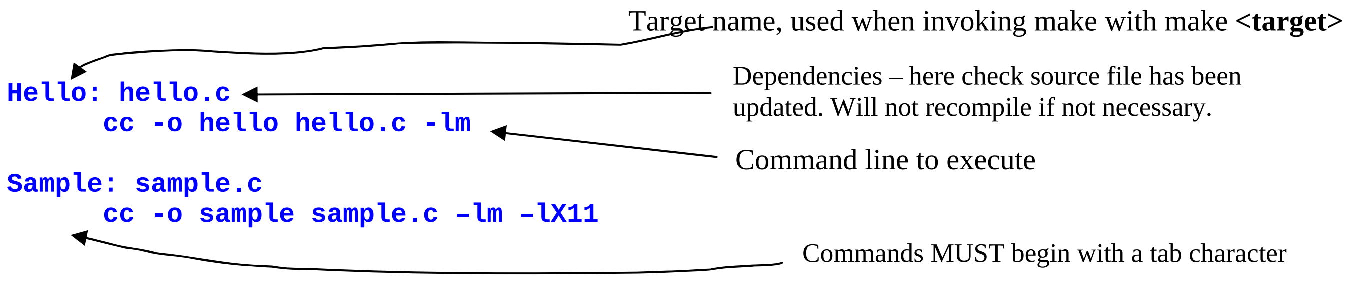
Order of libraries on command line can be important. Libraries must follow the source file. Symbols are resolved from left to right.

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

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

Make file

A make file is most convenient to issue the compilation command especially if the compilation command gets long. For example, a file called **makefile** with the contents shown in blue below:



```
Hello: hello.c
	cc -o hello hello.c -lm

Sample: sample.c
	cc -o sample sample.c –lm –lX11
```

Target name, used when invoking make with make <target>
Dependencies – here check source file has been updated. Will not recompile if not necessary.
Command line to execute
Commands MUST begin with a tab character

will compile a regular C program, **hello.c**, with the command:

```
make Hello
```

or an X11 program **sample.c** with the command:

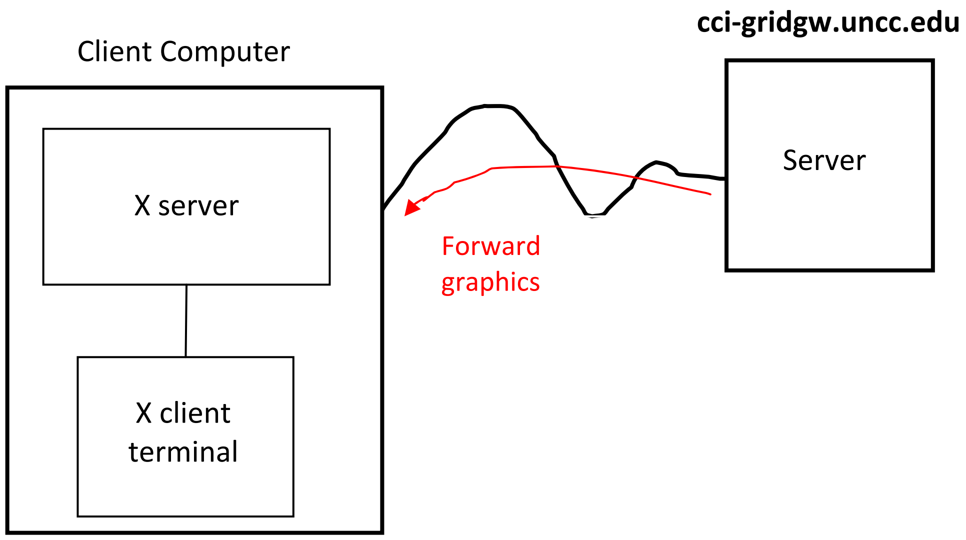
```
make Sample
```

A make file for sample is provided. The program is executed as an executable, i.e.:

```
./sample
```

Using a remote server

To execute X11 programs on a remote server and see the graphical output on your computer, you have to forward the graphics. This relies on an X-server running on your computer:



(For a Windows client computer, an X server would need to be installed such as Xming.)

To forward X11 graphics from a terminal on your computer, connect to the remote server with the –X option:

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

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

Test the connection and forwarding by running **xclock** in the background (with &):

```
xclock &
```

You should see clock appear on your computer.

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 –X ccigrid05
```

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.)

Useful references

Wikipedia X Window System http://en.wikipedia.org/wiki/X_Window_System
Wikipedia entry: X-11 color names http://en.wikipedia.org/wiki/X11_color_names
XLib Manual http://tronche.com/gui/x/xlib/
X11 graphics routines http://tronche.com/gui/x/xlib/graphics/

Appendix
X11Macros.h

```
// set up and initialization macro for X11 graphics. B. Wilkinson June 1, 2015
// x_resn, y_resn are the resolutions in x and y direction provided by programmer

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

#define initX11(x_resn,y_resn) \
Window             win;\
unsigned int width, height,win_x,win_y,border_width,display_width, display_height,screen;\
char               *window_name = "N-body with X11 graphics, Nbody-G.c", *display_name = NULL;\
GC                 gc;\
unsigned long      valuemask = 0;\
XGCValues          values;\
Display            *display;\
XSizeHints         size_hints;\
Pixmap             bitmap;\
XPoint             points[800];\
XSetWindowAttributes attr[1];\
if ((display = XOpenDisplay (display_name)) == NULL ) {  \
          fprintf (stderr, "drawon: cannot connect to X server %s\n",XDisplayName (display_name) );\
          exit (-1);\
}\
screen = DefaultScreen (display);\
display_width = DisplayWidth (display, screen);\
display_height = DisplayHeight (display, screen);\
width = x_resn;\
height = y_resn;\
win_x = 0;\
win_y = 0;\
border_width = 4;\
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);\
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);\
usleep(1000)


#define BLACK (long) 0x000000
#define BLUE (long) 0x0000FF
#define BROWN (long) 0xA52A2A
#define CYAN (long) 0x00FFFF
#define GRAY (long) 0xBEBEBE
#define GREEN (long) 0x00FF00
#define MAGENTA (long) 0xFF00FF
#define ORANGE (long) 0xFFA500
#define PINK (long) 0xFFC0CB
#define PURPLE (long) 0xA020F0
#define RED (long) 0xFF0000
#define TURQUOISE (long) 0x40E0D0
#define VIOLET (long) 0xEE82EE
#define WHITE (long) 0xFFFFFF
#define YELLOW (long) 0xFFFF00
```