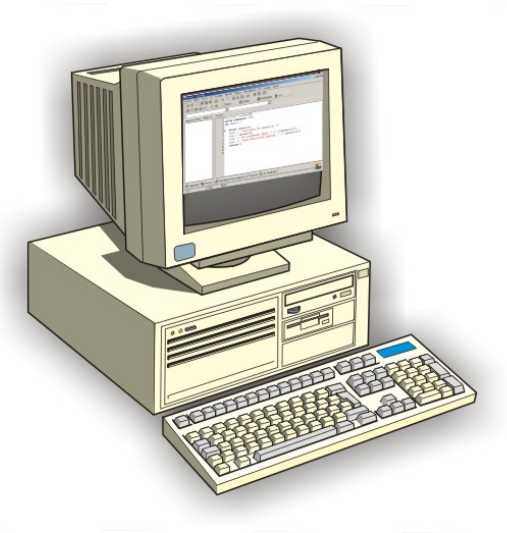

Guión 1

Introducción a X-Windows

Octubre de 2011

DECSAI
Departamento de Ciencias de la Computación e I.A.
Universidad de Granada

Nuevas Tecnologías de la Programación

Curso 2011/2012

Índice

Utilizaremos linux fedora 6 para este guión. Este guión servirá de introducción a algunos aspectos que serán útiles para la realización de la práctica 1 de la asignatura.

1. Ejecución de un programa X en un ordenador remoto

En primer lugar veremos cómo ejecutar una aplicación X en un ordenador remoto, visualizándola en el ordenador local (dónde estamos trabajando). Debido a las particularidades de la configuración de linux en las aulas de prácticas de la ETSIIT, no existe actualmente ningún servidor (ordenador remoto) al que nos podamos conectar con `telnet` o `ssh` para porder ejecutar algún programa en remoto. Sin embargo, podemos usar como ordenador remoto, otro ordenador del aula, arrancado en linux fedora 6, con tu mismo login y password. No es necesario entrar en el entorno de ventanas en el ordenador remoto. Supongamos que el nombre de tal ordenador es **ei150150**. Tal nombre suele estar escrito en una pegatina encima del monitor del ordenador o de la CPU. También lo puedes averiguar ejecutando el siguiente comando en una consola del ordenador remoto:

```
uname -a
```

Ejecutaremos una aplicación X en **ei150150** (ordenador remoto) y la visualizaremos en nuestra máquina realizando los siguientes pasos:

- Conectarnos a **ei150150** mediante el comando *ssh* (el comando `telnet` no se puede utilizar en los ordenadores de las aulas por motivos de seguridad). En el ordenador local, abre una shell y ejecuta:

  ```
  ssh ei150150.ugr.es
  ```

  Introduce tu login y password para conectarte a **ei150150**.

  En la forma en que nos hemos conectado al ordenador remoto, las aplicaciones X no se visualirán en el ordenador local en el que estáis trabajando con su teclado, pantalla y ratón (el que está ejecutando el servidor X). Podeís comprobar esto según se indica a continuación.

- En la consola en la que estás conectado al ordenador remoto, ejecuta alguna aplicación X de **ei150150** como por ejemplo *xclock*. Se mostrará un mensaje de errror indicándonos que no se puede abrir el display. Esto es debido a que se intenta usar como display el mismo ordenador remoto, que ni si quiera tiene iniciado el servidor X.

- Para que el display sea el ordenador local, volveremos a conectarnos al ordenador remoto desde otra consola, de la siguiente forma:

  ```
  ssh -X ei150150.ugr.es
  ```

Con esto, nos abremos conectado al ordenador remoto, y además el display a usar será el de el ordenador local.

- Ejecuta de nuevo alguna aplicación X de **ei150150** como por ejemplo *xclock*. La ventana de esta aplicación debería aparecer en tu ordenador.

2. Algunos comandos relacionados con X

Existen algunos comandos de Unix que están relacionados con X. Prueba a ejecutarlos en tu ordenador:

- **xhost**: Se usa para dar o quitar permiso de uso del display local si nos conectamos con `telnet` al ordenador remoto. Es posible usarlo de otras formas:
  - **xhost +**: Da permiso a cualquier ordenador a usar el display.
  - **xhost -**: Quita el permiso para usar el display a todos los ordenadores.
- **xdpyinfo**: Nos muestra información sobre el display que estamos utilizando.
- **xwininfo**: Una vez ejecutado si pinchamos con el ratón en una ventana nos da información sobre esa ventana.
- **xprop**: Una vez ejecutado si pinchamos con el ratón en una ventana nos muestra las propiedades de esa ventana.
- **xlsfonts**: Nos muestra una lista con los fonts instalado en nuestro ordenador.
- **xfd**: Nos abre una ventana mostrando el aspecto de un determinado font. Ejemplo:

```
xfd -fn 9x15
```

3. Compilación programas X

En el directorio `/fenix/depar/ccia/ntp/Xlib/XlibEjemplos` y en la página web `http://decsai.ugr.es/~acu/NTP/otros.html` existen algunos programas construidos con *Xlib*. Copia por ejemplo el fichero `basicwin.c`, que se encuentra en el subdirectorio *basic*, a un directorio de tu cuenta. Copia también el fichero `icon_bitmap` que se encuentra en el subdirectorio `bitmaps`. Este fichero es el icono del programa. Antes de compilar el programa, tenemos que modificar modificar el fichero `basicwin.c`. Editalo con algún editor de texto y cambia la línea

```
#include "../bitmaps/icon_bitmap"
```

por la línea

```
#include "icon_bitmap"
```

Cuando nuestro programa lo compone sólo un fichero fuente, podemos obtener el ejecutable en un solo paso o bien en dos pasos.

- En un solo paso:

```
gcc basicwin.c -o basicwin -lX11 -I/usr/X11R6/include
    -L/usr/X11R6/lib
```

- En dos pasos:
  - Compilación del fichero fuente:
    `gcc -c basicwin.c -o basicwin.o  -I/usr/X11R6/include`
  - Enlazado con la biblioteca Xlib:
    `gcc basicwin.o -o basicwin -lX11 -L/usr/X11R6/lib`

En ambos casos, el programa ejecutable queda almacenado en el fichero `basicwin`

Cuando nuestro programa está formado por más de un fichero fuente entonces es conveniente obtener el ejecutable compilando uno a uno los ficheros fuente, y luego linkando todos los ficheros objeto con la librería `Xlib`.

Para compilar un programa resulta más cómodo construir un fichero *makefile*. Por ejemplo el programa anterior podría compilarse con el siguiente fichero makefile:

```
basicwin: basicwin.o
        gcc basicwin.o -o basicwin -lX11 -L/usr/X11R6/lib
basicwin.o: basicwin.c
        gcc -c basicwin.c -o basicwin.o  -I/usr/X11R6/include
```

Una vez construido el fichero *makefile* el ejecutable se obtiene ejecutando simplemente el comando **make**.

En la página web de prácticas de la asignatura **Metodología de la Programación II** puedes encontrar una documentación más completa sobre la compilación de programas C y C++, así como de la construcción de ficheros makefile en el enlace *Introducción a la compilación de programas en C++*:
http://decsai.ugr.es/mp2/practicas.html

4. Construcción de un programa sencillo

En esta sección construiremos un programa Xlib en el que aparecerá una ventana con el fondo en color blanco. Dentro de la ventana se mostrará un círculo de color negro, inicialmente situado en la esquina superior izquierda de la ventana. El círculo se situará en otra posición si el

usuario pincha con el botón izquierdo del ratón en una posición de la ventana. Si el usuario arrastra el ratón teniendo pulsado el botón izquierdo, el círculo seguirá la posición del cursor del ratón. Si el usuario pincha con el botón derecho del ratón entonces el nuevo radio del círculo será la distancia entre el antiguo círculo y la posición donde se ha pinchado. Si arrastra el ratón teniendo pulsado el botón izquierdo, el radio del círculo se modificará a medida que se modifica la distancia entre el centro del círculo y la posición actual del cursor del ratón.

4.1. Variables necesarias

En los programas con un interfaz gráfico debemos pensar en primer lugar las estructuras de datos necesarias para guardar el estado actual de los datos del programa. En este caso los únicos datos que necesitamos guardar son los que representan el círculo (posición y radio actual). Cuando la estructura de datos es más o menos compleja, conviene separar el código que controla esta estructura en un fichero fuente aparte o clase si programamos en C++. En este caso la estructura de datos para almacenar el estado del círculo es muy simple y no la pondremos en un fichero fuente distinto: necesitamos tres números enteros, dos enteros para la posición del centro y otro para el radio. Podemos declarar estas variables de la siguiente forma dentro de la función `main`:

```
int currentX=0,currentY=0,currentR=10;
```

Mientras se arrastra el ratón con un botón pulsado necesitamos saber si debemos cambiar la posición del centro del círculo, o bien cambiar su radio. Para ello usaremos una variable `mode` que nos indicará el modo en que estamos (`CHANGERADIO`, `CHANGECENTER` o `NOCHANGE`). Incluimos en nuestro fichero fuente las tres constantes simbólicas siguientes:

```
#define CHANGECENTER 0
#define CHANGERADIO 1
#define NOCHANGE 2
```

y la variable `mode` la incluimos dentro del `main`, que inicializamos con `NOCHANGE`.

```
int mode=NOCHANGE;
```

El programa también necesitará una serie de variables relacionadas directamente con el interfaz gráfico que pasamos a describir en los siguientes párrafos.

En las funciones de Xlib se necesita siempre usar los parámetros `Display` y *número de screen*. Declararemos para ello dos variables globales en nuestro programa:

```
Display *display;
int screen_num;
```

Otras variables locales que declararemos dentro de la función `main` son las siguientes:

```
Window win;
unsigned int width, height, x, y; /* anchura de ventana y posición
unsigned int borderwidth = 4;     /* 4 pixels */
unsigned int display_width, display_height;
unsigned long foreground_pixel,
              background_pixel; /* valores de pixel (color) */
XEvent report;
GC gc;
XGCValues values;
```

El significado de estas variables es el siguiente:

- `Window win`: La ventana de la aplicación.
- `unsigned int width, height, x, y`: Anchura, altura y posición de la ventana.
- `unsigned int display_width, display_height`: Serán usadas para obtener la anchura y altura del display (pantalla).
- `long foreground_pixel`: Color con el que dibujaremos en la ventana el círculo.
- `background_pixel`: Color del *background* de la ventana.
- `XEvent report`: Usado para ir leyendo un evento de la cola de eventos.
- `GC gc`: El contexto gráfico usado para dibujar el círculo.
- `XGCValues values`: Máscara usada en la creación del contexto gráfico.

4.2. Creación del programa

Una vez analizadas las variables que vamos a necesitar comencemos a crear el programa. Crearemos un fichero vacío que llamaremos `circulo.c` con cualquier editor ASCII. Para usar funciones de *Xlib* tenemos que incluir el fichero `X11/Xlib.h`. También incluimos los ficheros `stdio.h` y `math.h` pues usaremos funciones de estos ficheros de cabecera. Por ahora el fichero podría quedar como sigue:

```
#include <X11/Xlib.h>
#include <stdio.h>
#include <math.h>

#define CHANGECENTER 0
#define CHANGERADIO 1
#define NOCHANGE 2

Display *display;
int screen_num;
int main(int argc,char **argv)
```

```
{
  int currentX=0,currentY=0,currentR=10;
  int mode=NOCHANGE;
  Window win;
  unsigned int width, height, x, y; /* anchura de ventana y posición */
  unsigned int borderwidth = 4;            /* 4 pixels */
  unsigned int display_width, display_height;
  unsigned long foreground_pixel,
                background_pixel; /* valores de pixel (color) */
  XEvent report;
  GC gc;
  XGCValues values;
}
```

El primer paso que hará el programa será la conexión con el display X mediante la función `XOpenDisplay()`:

```
if ( (display=XOpenDisplay(NULL)) == NULL ){
     (void) fprintf(stderr,"circulo: no puedo conectar al servidor X %s\\n");
     exit( -1 );
}
```

A continuación inicializamos la variable `screen_num` con:

```
screen_num = DefaultScreen(display);
```

Ahora calculamos anchura, altura y posición de la ventana en la siguiente forma:

```
display_width = DisplayWidth(display, screen_num);
display_height = DisplayHeight(display, screen_num);
x = display_width/3, y = display_height/3;
width = display_width/3, height = display_height/4;
```

Los colores usados como *foreground* y *background* los definimos de la siguiente forma:

```
background_pixel=WhitePixel(display,screen_num);
foreground_pixel=BlackPixel(display,screen_num);
```

Creamos la ventana de la aplicación:

```
win = XCreateSimpleWindow(display, RootWindow(display,screen_num), x, y,
         width, height, borderwidth, BlackPixel(display,screen_num),
         background_pixel);
```

Seleccionamos los eventos `Expose` (pérdida de contenidos en la ventana), `ButtonPress` (pulsación de un botón del ratón) y `MotionNotify` (movimiento del cursor del ratón) en la ventana de la aplicación:

```
XSelectInput(display, win, ExposureMask | ButtonPressMask|
             ButtonMotionMask);
```

Creamos el contexto gráfico que será usado para dibujar el círculo en la ventana:

```
gc = XCreateGC(display, win, 0 , &values);
```

Mapeamos la ventana en el display (hacemos que sea visible en pantalla):

```
XMapWindow(display, win);
```

El programa tendría por ahora el siguiente contenido:

```
#include <X11/Xlib.h>
#include <stdio.h>
#include <math.h>

#define CHANGECENTER 0
#define CHANGERADIO 1
#define NOCHANGE 2

Display *display;
int screen_num;
int main(int argc,char **argv)
{
  int currentX=0,currentY=0,currentR=10;
  int mode=NOCHANGE;
  Window win;
  unsigned int width, height, x, y; /* anchura de ventana y posición */
  unsigned int borderwidth = 4;            /* 4 pixels */
  unsigned int display_width, display_height;
  unsigned long foreground_pixel,
                background_pixel; /* valores de pixel (color) */
  XEvent report;
  GC gc;
  XGCValues values;

  if ( (display=XOpenDisplay(NULL)) == NULL ){
      (void) fprintf(stderr,"circulo: no puedo conectar al servidor X %s\\n");
      exit( -1 );
  }
  screen_num = DefaultScreen(display);
  display_width = DisplayWidth(display, screen_num);
  display_height = DisplayHeight(display, screen_num);
  x = display_width/3, y = display_height/3;
  width = display_width/3, height = display_height/4;
  background_pixel=WhitePixel(display,screen_num);
  foreground_pixel=BlackPixel(display,screen_num);
  win = XCreateSimpleWindow(display, RootWindow(display,screen_num), x, y,
         width, height, borderwidth, BlackPixel(display,screen_num),
         background_pixel);
  XSelectInput(display, win, ExposureMask | ButtonPressMask|
               ButtonMotionMask);
  gc = XCreateGC(display, win, 0 , &values);
  XMapWindow(display, win);
}
```

Ya sólo nos queda construir el *bucle de eventos* del programa. Es un bucle infinito en el que en cada pasada se lee un evento, se ve de qué tipo es y se realiza una u otra acción dependiendo del tipo. El esqueleto de este bucle a falta de incluir el código para cada evento quedaría de la siguiente forma:

```
while (1)  {
  XNextEvent(display, &report);
  switch  (report.type) {
    case Expose:
         break;
    case ButtonPress:
         break;
    case MotionNotify:
         break;
  }
}
```

El código para el evento `Expose` debe dibujar el círculo en su posición actual con su radio correspondiente:

```
case Expose:
     XSetFunction(display, gc, GXcopy);
     XSetForeground(display,gc,foreground_pixel);
     XDrawArc(display,win,gc,currentX-currentR,currentY-currentR,
              currentR*2,currentR*2,0,360*64);
     break;
```

En el evento `ButtonPress` debemos ver si hemos pinchado en el botón izquierdo (`Button1`) o el botón derecho (`Button3`). Si se ha pinchado con el botón izquierdo pasaremos a modo `CHANGECENTER`, borramos el círculo de su posición anterior, cambiamos la posición del centro a la posición del cursor, y dibujamos el círculo en la nueva posición. Si se ha pinchado con el botón derecho pasaremos a modo `CHANGERADIO`, borramos el círculo de su posición anterior, asignamos al radio del círculo la distancia entre el centro del círculo y la posición del cursor del ratón, y dibujamos el círculo con el nuevo radio:

```
case ButtonPress:
     if(report.xbutton.button==Button1){
         mode=CHANGECENTER;
         XSetFunction(display, gc, GXxor);
         XSetForeground(display,gc,foreground_pixel^background_pixel);
         XDrawArc(display, win, gc, currentX-currentR,
                  currentY-currentR,currentR*2,currentR*2,0,360*64);
         currentX=report.xbutton.x;
         currentY=report.xbutton.y;
         XDrawArc(display, win, gc, currentX-currentR,
                  currentY-currentR,currentR*2,currentR*2,0,360*64);
     }
     else if(report.xbutton.button==Button3){
         mode=CHANGERADIO;
         XSetFunction(display, gc, GXxor);
         XSetForeground(display,gc,foreground_pixel^background_pixel);
         XDrawArc(display, win, gc, currentX-currentR,
                  currentY-currentR,currentR*2,currentR*2,0,360*64);
         currentR=distancia(report.xbutton.x,report.xbutton.y,
                            currentX,currentY);
         XDrawArc(display, win, gc, currentX-currentR,
                  currentY-currentR,currentR*2,currentR*2,0,360*64);
     }
     else
         mode=NOCHANGE;
     break;
```

Se deja como ejercicio incluir el código para el evento `MotionNotify`.

4.3. Funciones auxiliares

En este sencillo programa sólo hemos utilizado una función auxiliar. Construiremos una función para calcular la distancia euclídea entre dos puntos:

```
int distancia(int x1,int y1,int x2, int y2){
  return sqrt((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1));
}
```

Ahora puedes compilar el programa y ver si funciona correctamente.

5. Creación del fichero icono de una aplicación

Los ficheros usados por las aplicaciones X como iconos son ficheros de texto. Por ejemplo el contenido del fichero `icon_bitmap` usado en la aplicación que hemos compilado anteriormente es:

```
#define icon_bitmap_width 40
#define icon_bitmap_height 40
static char icon_bitmap_bits[] = {
```

```
   0xc3, 0xc3, 0x7f, 0x00, 0x78, 0x00, 0x00, 0x00, 0x00, 0xc0, 0x00, 0x00,
   0x00, 0x00, 0x80, 0x38, 0x00, 0x40, 0x00, 0x80, 0x24, 0x00, 0x00, 0x00,
   0x80, 0x44, 0x00, 0x00, 0x00, 0x80, 0x44, 0x00, 0x00, 0x00, 0x80, 0x74,
   0x00, 0x0f, 0x0c, 0x00, 0x7c, 0x3e, 0x41, 0x0e, 0x00, 0x44, 0x22, 0x41,
   0x02, 0x00, 0x84, 0x22, 0x46, 0x02, 0x00, 0x9c, 0x26, 0xcc, 0x02, 0x00,
   0x78, 0x3c, 0xcd, 0x36, 0x80, 0x00, 0x20, 0x06, 0x0c, 0x80, 0x01, 0x00,
   0x00, 0x00, 0x80, 0x01, 0x02, 0x40, 0x00, 0x80, 0x01, 0x06, 0x40, 0x00,
   0x80, 0x01, 0x04, 0x20, 0x00, 0x80, 0x01, 0x04, 0x20, 0x01, 0x80, 0x01,
   0x04, 0x20, 0x00, 0x80, 0x01, 0x04, 0x22, 0x00, 0x80, 0x01, 0x04, 0x33,
   0xf1, 0x81, 0x01, 0x88, 0x12, 0x31, 0x03, 0x01, 0x88, 0x12, 0x11, 0x02,
   0x00, 0x88, 0x12, 0x11, 0x02, 0x00, 0x48, 0x1a, 0x11, 0x02, 0x00, 0x70,
   0x04, 0x19, 0x82, 0x01, 0x00, 0x00, 0x00, 0x80, 0x01, 0x00, 0x00, 0x38,
   0x80, 0x01, 0x00, 0x00, 0xce, 0x80, 0x01, 0x00, 0x00, 0x83, 0x81, 0x81,
   0x07, 0x80, 0x01, 0x81, 0xe1, 0x04, 0xc0, 0x00, 0x83, 0x31, 0x08, 0x40,
   0x00, 0x82, 0x10, 0x08, 0x20, 0x00, 0x82, 0x19, 0x10, 0x30, 0x00, 0x86,
   0x0c, 0x30, 0x18, 0x00, 0x84, 0x04, 0x60, 0x0e, 0x00, 0xdc, 0x02, 0x80,
   0x03, 0x00, 0x70, 0x00, 0x00, 0x00, 0x00, 0x00};
```

Para editar visualmente este tipo de ficheros puedes utilizar una aplicación editora de bitmaps como por ejemplo la aplicación *bitmap*:

```
bitmap icon_bitmap
```