University of Bologna - Department of Mathematics

Piazza di Porta S.Donato, 5 - 40127 - Bologna

xtools library
Programming Guide - Version 2.0

S. Bonetti G. Casciola

Department of Mathematics
University of Bologna

Bologna 2000

Abstract

This report describes the *xtools* library. It is designed to produced graphics interfaces in the simplest and easiest way, instead of using the more complex and larger Xwindow tools. This Version 2.0 includes new useful functions.

S. Bonetti G. Casciola
Department of Mathematics, University of Bologna, P.zza di Porta S.Donato 5, Bologna, Italy. E-mail: casciola@dm.unibo.it.

Contents

Contents **i**

1 What is xtools? **1**

2 The objects of xtools **3**

3 How to make an interface **5**

4 Classes of functions **7**
4.1 AddType(par) 7
4.2 SetTypeEnable(par) 8
4.3 SetTypeLabel(par) 9
4.4 Event Types and their managing 9
4.4.1 SetTypesBPScript(par) 10
4.4.2 SetTypeBPScript(par) 10
4.4.3 SetTypeBRScript(par) 11
4.4.4 SetTypeLFScript(par) 11
4.4.5 SetTypeMNScript(par) 11
4.4.6 SetTypeEXScript(par) 11
4.4.7 SetTypeCNScript(par) 12
4.5 SetTypeHelp(par) 12
4.6 SetTypeValue(par) 12
4.7 return_type *GetTypeValue(par) 12

5 Functions for objects **13**
5.1 Methods for textbox tools 13
5.2 Methods for bar tools 13
5.3 Methods for box tools 14
5.4 Methods for checkpoint tools 15
5.5 Methods for window tools 15
5.6 Methods for message-box tools 16
5.7 Method for file requester 16

6 Generic functions **17**

7 A programming example **19**

8 How to compile **29**

List of Figures **31**

Bibliography **33**

Chapter 1

What is xtools?

xtools is a library made for Unix and Xwindow environments. It is designed to produce graphic interfaces in the simplest and easiest way, instead of using the more complex and larger Xwindow tools (see [NYOR92]). Xtools is designed for a limited number of objects (see chapter 2) and the use of three classes of functions: create, execute and able/unable. Creation defines the name and shape of the object, execute creates the event-object-function association in order to execute a function when an event occurs on an object. Able/unable allows the object to be active or not. Therefore, xtools is very fast and uses little memory, while still being efficient and effective.

Chapter 2

The objects of xtools

These can be divided into two categories and are illustrated in fig.1.

1.primitives are the basis for interface construction and these consist of the following:

bar is a white rectangle (bar), within which a cursor can move horizontally or vertically (depending on whether the bar itself is horizontal or vertical). The position of the cursor within the bar "corresponds" to the value of a variable, which is obviously discrete;

box is a white rectangle containing a list of items;

button is a grey rectangle which performs an action when pressed;

checkbox is a white rectangle, which can be crossed. It represents an option;

checkpoint is a grey rectangle which, when pressed, contains a dark square. These are mutually exclusive objects, i.e. in a group of checkpoints, when one is selected the others are automatically excluded. However, there may be more than one group of checkpoints in a window;

cursor is a group of four direction buttons: top, bottom, left and right;

label is a string of characters that can be positioned at any point on the window.

textbox is a white rectangle that can contain data;

window is a real X window that cannot be resizeble i.e. its size is fixed for the entire execution;

Rwindow is a real X window that can be resizeble, but cannot be used to contain xtools objects;

2.derivates are constructed from primitives and are the actual programs of xtools to carry out specific functions. They consist of the following:

help text is a string of text which appears at the bottom of the window. If it is combined with an object, when the mouse passes over this object, a message appears;

file request is used to insert a file;

message box provides the user with informations

As we can see from the description, some of these objects can be used as actual variables within user programs. This is true of bars, that are discrete representations of real variables (also integer), boxes, which enable a list of strings to be managed interactively, checkboxes, that can be viewed as boolean variables, and checkpoints as a choice made from a menu. Textboxes are variables that may be classified as required. They may be a string, an integer or a real; the type is chosen when created and cannot be changed.

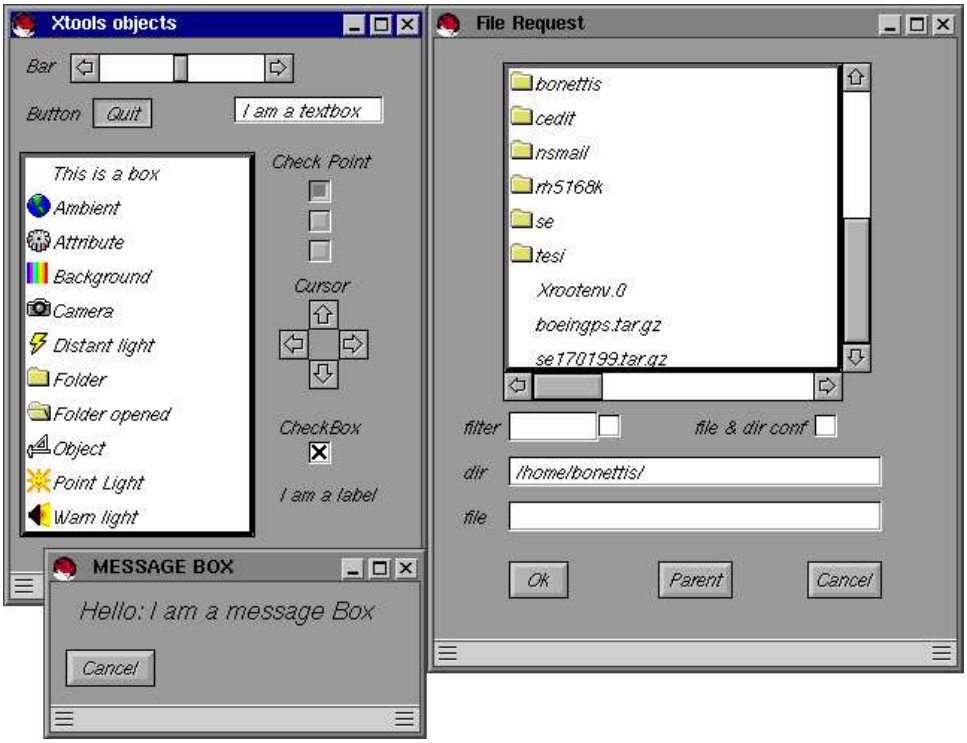

Figure 2.1: All the xtools's objects

Chapter 3

How to make an interface

Every program using xtools has the following structure:

```
#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include "xtools/xtools.h"

GUI_t inter; /* the GUI id*/
Win_t Win;   /* the window id */

int main(void) {

  Initgraph();
  InitGUI(&inter);
  CreateWin();    /* function defined by the user */
  AddWindowToGUI(&inter, &Win);
  RunGUI(&inter, &Win);
  return(0);
}
```

Since xtools uses the Xlib library the first two include lines must ever be included. The function which the GUI programmer has to write is **CreateWin**, in which, by calling xtools functions, a window is created, objects are positioned in the window, are abled/unabled and user functions are associated with events. **AddWindowToGUI**, whose second parameter is the window defined in CreateWin, enables to add the window to the GUI.

Chapter 4

Classes of functions

The use of xtools functions is analysed by class. Every function belongs to one of three classes: create, execute, able/unable and is associated with one of the primitive objects.

4.1 AddType(par)

These functions define and add objects in a window. All the functions in this class have the following parameters:

-win the window in which the objects must be viewed;

-id the "name" by which the objects are identified;

-x,y the object position in the window;

-w,h the object size (the labels have a fixed size).

```
void AddBar(Win_t *win, int id, int x, int y,
            unsigned int w,unsigned int h,
            float minval, float maxval,
            float step, float value);
void AddBox(Win_t *win, int id, int x, int y,
            unsigned int w,unsigned int h,
            Array_Entry_t *table);
void AddButton(Win_t *win, int id, int x, int y,
               unsigned int w, unsigned int h,
               char *cap);
void AddCheckBox(Win_t *win, int id, int x, int y,
                 unsigned int w, unsigned int h,
                 int value);
void AddCheckPoint(Win_t *win, int gid, int nid,
                   int x, int y,
```

```
                    unsigned int w, unsigned int h,
                    int value);
    void AddCursor(Win_t *win, int id, int x, int y,
                   unsigned int w, unsigned int h);
    void AddLabel(Win_t *win, int id, int x, int y,
                  unsigned long col, char *cap);
    void AddTextBox(Win_t *win, int id, int x, int y,
                    unsigned int w, unsigned int h,
                    char *text, int type,
                    unsigned int maxchar, int ie);
```

Every function has its own characteristics: e.g. in bar the dimensions of the interval of the variable it represents must be given, as well as the step (which is the rate at which the cursor moves along the bar) and the initial value; box needs a list of strings to be viewed; in button, the selected string must be given; in checkbox the initial value; in checkpoint the group to which it belongs (checkpoints must be grouped together); in label the string to be viewed and its colour must be indicated; finally, in textbox the first string must be setted (it may also be empty), and the type of data which must be one of the following:

```
__XT_FLOAT  /* textbox float */
__XT_TEXT   /* textbox char */
__XT_INT    /* textbox int */
__XT_DOUBLE /* textbox double */
```

as well as the maximum number of characters that it can contain and if it possible to make insertions (boolean parameter ie).

4.2 SetTypeEnable(par)

These functions make the object identified by win and id able to be managed by events, "abled" or "unabled". For example, if a bar is FALSE, it will be invisible to the event manager, and its appearance will be different. Even the pointer will be different, becoming an X, when it passes over the bar.

```
void SetBarEnable(Win_t *win, int id, BOOLEAN status);
void SetBoxEnable(Win_t *win, int id, BOOLEAN status);
void SetButtonEnable(Win_t *win, int id, BOOLEAN status);
void SetCheckBoxEnable(Win_t *win, int id, BOOLEAN status);
void SetCursorEnable(Win_t *win, int id, BOOLEAN status);
void SetLabelEnable(Win_t *win, int id, BOOLEAN status);
void SetTextEnable(Win_t *win, int id, BOOLEAN status);
void SetCheckPointEnable(Win_t *win, int gid, int nid, BOOLEAN status);
void SetCheckPointsEnable(Win_t *win, int gid, BOOLEAN status);
```

4.3 SetTypeLabel(par)

These functions allow the user to link the object identified by win and id to an id label (lid) of the same window, so that the SetTypeEnable of an object ables or unables the linked label too. In the case of a group of checkpoints, the function SetCheckPointsEnable() changes the status of all the labels linked to the checkpoints belonging to that group according to the group and single checkpoint states.

```
void SetBarLabel(Win_t *win, int id, int lid);
void SetBoxLabel(Win_t *win, int id, int lid);
void SetCheckBoxLabel(Win_t *win, int id, int lid);
void SetCheckPointLabel(Win_t *win, int gid, int nid, int lid);
void SetCursorLabel(Win_t *win, int id, int lid);
void SetTextBoxLabel(Win_t *win, int id, int lid);
```

4.4 Event Types and their managing

The event types foreseen by xtools are:

Button Press (BP)

Button Release (BR)

Lost Focus (LF)

Motion Notify (MN)

Expose (EX)

Configure Notify (CN)

Each of these events is associated with the mouse and not with the object to be managed. When a BP is managed on a button, the event considered is the mouse pressed on a button. Their names are self-explanatory. Some objects cannot be connected with these events, e.g. the BR in a box cannot be scheduled. Lost focus expresses the transition of the mouse and the only object that can be connected with it is textbox (when the insertion is finished, do not use the return key, but move the mouse pointer outside the textbox). The same is true of motion notify, which expresses a transition, when a button is pressed. The only objects that can be matched with this type of event are bars. Finally, the event Expose only belongs to the windows and it is what makes them visible, while the event Configure Notify is linked to a Rwindow and expresses its resize.

4.4.1 SetTypesBPScript(par)

These functions serve to match a BP event, that has been carried out on an object, with a user function . In other words, they manage the BP event of an object of GUI. The plural expressed in the name of the objects (e.g. Bars) is important to note. This means that the function in question will be invoked whenever a BP occurs in all the objects of that type defined in that window. The associated functions have void parameters and return void.

```
void SetBarsBPScript(Win_t *win,
                     void (*Script)(void));
void SetBoxesBPScript(Win_t *win,
                      void (*Script)(void));
void SetButtonsBPScript(Win_t *win,
                        void (*Script)(void));
void SetCheckBoxesBPScript(Win_t *win,
                           void (*BPScript)(void));
void SetCheckPointsBPScript(Win_t *win,
                            void (*BPScript)(void));
void SetCursorsBPScript(Win_t *win,
                        void (*Script)(void));
void SetTextsBPScript(Win_t *win,
                      void (*BPScript)(void));
```

4.4.2 SetTypeBPScript(par)

```
void SetBarBPScript(Win_t *win, int id,
                    void (*Script)(void));
void SetBoxBPScript(Win_t *win, int id,
                    void (*Script)(void));
void SetButtonBPScript(Win_t *win, int id,
                       void (*Script)(void));
void SetCheckBoxBPScript(Win_t *win, int id,
                         void (*BPScript)(void));
void SetCheckPointBPScript(Win_t *win, int gid,int nid,
                           void (*Script)(void));
void SetCursorBPScript(Win_t *win, int id,
                       void (*Script)(int));
void SetTextBPScript(Win_t *win, int id,
                     void (*Script)(void));
void SetWindowBPScript(Win_t *win,
                       void (*Script)(void));
```

As for the above, these also match the BP of a specific object within the window with a user function (see id field). As for the above, the return value

and the list of parameters are void, except for the cursors. In fact, these enable the function to have an int argument representing the pressed cursor button and represented by one of the following constants:

```
__XT_CRS_UP
__XT_CRS_RIGHT
__XT_CRS_DOWN
__XT_CRS_LEFT
```

4.4.3 SetTypeBRScript(par)

As for the above, except that the managed event is ButtonReleased (BR).

```
void SetBarBRScript(Win_t *win, int id,
                    void (*Script)(void));
void SetButtonBRScript(Win_t *win, int id,
                       void (*Script)(void));
void SetCursorBRScript(Win_t *win, int id,
                       void (*Script)(int));
void SetWindowBRScript(Win_t *win,
                       void (*Script)(void));
```

4.4.4 SetTypeLFScript(par)

As for the above, except that the managed event is LostFocus (LF).

```
void SetTextLFScript(Win_t *win, int id,
                     void (*Script)(void));
```

4.4.5 SetTypeMNScript(par)

As for the above, except that the managed event is MotionNotify (MN).

```
void SetBarMNScript(Win_t *win, int id,
                    void (*Script)(void));
void SetButtonMNScript(Win_t *win, int id,
                       void (*Script)(void));
void SetTextMNScript(Win_t *win, int id,
                     void (*Script)(void));
void SetWindowMNScript(Win_t *win,
                       void (*Script)(void));
```

4.4.6 SetTypeEXScript(par)

```
void SetWindowEXScript(Win_t *win,
                       void (*Script)(void));
```

sets the function to be performed when the event Expose (EX) of the window win occurs.

4.4.7 SetTypeCNScript(par)

```
void SetWindowCNScript(Win_t *win,
                       void (*Script)(void));
```

sets the function to be performed when the event Configure Notify (CN) of the Rwindow win occurs.

4.5 SetTypeHelp(par)

```
void SetBarHelp(Win_t *win, int id, char *msg);
void SetBoxHelp(Win_t *win, int id, char *msg);
void SetButtonHelp(Win_t *win, int id, char *msg);
void SetCursorHelp(Win_t *win, int id, char *msg);
void SetTextHelp(Win_t *win, int id, char *msg);
```

allow a help string for the object in question to be viewed along the bottom of the window. This message will be visible whenever the cursor moves over an object.

4.6 SetTypeValue(par)

```
void SetBarValue(Win_t *win, int id, float value);
void SetCheckBoxValue(Win_t *win, int id, int value);
void SetTextValue(Win_t *win, int id, void *value);
```

allow the object to be set with a specific value.

4.7 return_type *GetTypeValue(par)

```
float GetBarValue(Win_t *win, int id);
BOOLEAN GetCheckBoxValue(Win_t *win, int id);
char *GetTextValue(Win_t *win, int id);
```

return the value of the specified object to be known.

Chapter 5

Functions for objects

Specific functions are presented for each object.

5.1 Methods for textbox tools

```
BOOLEAN GetTextInputEnable(Win_t *win, int id);
```

returns the status of the textbox to be known, whether it can receive input or not.

```
void SetTextInputEnable(Win_t *win, int id, int status);
```

allows the able/unable of the input in the textbox.

5.2 Methods for bar tools

```
void SetBarLimits(Win_t *win, int id, float min,
                  float max, float step, float value);
void SetBarStep(Win_t *win, int id, float step);
```

allow the bar parameters, the range (min, max), the actual value (value) and the step (step), to be changed during the execution.

```
float GetBarMin(Win_t *win, int id);
float GetBarMax(Win_t *win, int id);
float GetBarStep(Win_t *win, int id);
```

return the value of the variable bar (id) to be known.

```
void SetBarTxtLink(Win_t *win, int id, int txtid);
```

every bar can be linked to a textbox that views its actual value.

5.3 Methods for box tools

```
void SetBoxBarlink(Win_t *win, int boxid,
                   int vbarid, int hbarid);
```

every box may be linked to two bars (even a single one, in which case vbarid or hbarid will be worth __XT_NO_ID), which are required for horizontal and vertical scrolling.

```
unsigned int GetBoxRow(Win_t *win, int id);
unsigned int GetBoxCol(Win_t *win, int id);
```

return the row and column values of a box.

```
void SetBoxRow(Win_t *win, int id, unsigned int row);
void SetBoxCol(Win_t *win, int id, unsigned int col);
void SelectBoxRow(Win_t *win, int boxid, unsigned int row);
```

set the row and column values of a box.

```
void ReDrawBox(Win_t *win, int id, unsigned int row,
               unsigned int col);
```

draws the pixmap of a box at a particular row and column of the window.

```
void UpdateBox(Win_t *win, int id);
```

serves to update the viewing of a box, to be used only after updating the string list of the box.

The following functions serve to manage the list of strings in the box. Their names are self-explanatory.

```
void InitTable(Array_Entry_t *t);
void InsertEntry(Array_Entry_t *table,
                 unsigned int idx_ico,
                 int indent, char *str);
void InsertNEntry(Array_Entry_t *table, unsigned int n,
                  unsigned int idx_ico, int indent,
                  char *str);
void InsertInTop(Array_Entry_t *table,
                 unsigned int idx_ico,
                 int indent, char *str);
void InsertInBottom(Array_Entry_t *table,
                    unsigned int idx_ico,
                    int indent, char *str);
void Append(Entry_t *main, Entry_t *secondary);
void PrintTable(Array_Entry_t *t);
void SetFirst(Array_Entry_t *table, unsigned int idx_ico,
```

```
                int indent, char *str);
void SetLast(Array_Entry_t *table, unsigned int idx_ico,
             int indent, char *str);
void SetN(Array_Entry_t *table, unsigned int n,
          unsigned int idx_ico, int indent, char *str);
void DeleteInTop(Array_Entry_t *table);
void DeleteInBottom(Array_Entry_t *table);
void DeleteNEntry(Array_Entry_t *table, unsigned int n);
void DeleteAll(Array_Entry_t *table);
char *GetFirst(Array_Entry_t *table);
char *GetLast(Array_Entry_t *table);
char *GetN(Array_Entry_t *table, unsigned int n);
unsigned int Verify(Array_Entry_t *table, char *name);
```

5.4 Methods for checkpoint tools

```
int GetCheckPointOn(Win_t *win, int gid);
```

returns the value of the checkpoint.

```
void SetCheckPointOn(Win_t *win, int gid, int nid);
```

sets the value of the checkpoint.

5.5 Methods for window tools

```
void CreateWindow(Win_t *win, int x, int y,
                  unsigned int w, unsigned int h,
                  unsigned int bordcol,
                  unsigned long backcol,
                  Colormap colormap,
                  char *title);
```

creates a window. It is the instruction that precedes the definition of the objects in a window.

```
void ShowWindow(Win_t *win);
```

makes a window to be visible.

```
void HideWindow(Win_t *win);
```

makes a window to be invisible.

```
void SetCursor(Win_t *win, unsigned int cid);
```

sets the cursor image of the mouse.

```
void CancelCursor(Win_t *win);
```

resets the mouse cursor.

```
void SetBackground(Win_t *win, XImage *XImageBuffer,
                   unsigned int w, unsigned int h);
```

sets the background of a window.

5.6 Methods for message-box tools

```
int MsgBox(char *msg, int mode);
```

views a window containing the message msg. This window has at least one button according to the mode value:

```
__XT_NORMAL
__XT_NOYES
```

In the first case, it will be a message window that provides information, in the second, it will be a window that provides a question, in other words, there will be three buttons: dismiss, yes and no. The given value can only be:

```
__XT_DISMISSKEY
__XT_YESKEY
__XT_NOKEY
```

```
void SetMsgBoxBackground(XImage *image, unsigned int w,
                         unsigned int h);
```

sets a background image for the message.

5.7 Method for file requester

What follows is not an actual object of xtools, but, like the msgbox, it is a window that is processed in a particular way.

```
char *FileReq(char *str_dir, BOOLEAN file_conf,
              BOOLEAN filter, char *str_flt);
```

creates a file requester that initially views the directory dir. If this does not exist, it sends an error msg and views the $HOME; file_conf shows that the files beginning with a "." are to be viewed; filter shows the status of the textbox of the filter, that can be selected by the user; str_flt shows the extension of the files to be viewed, and considers the regular expressions consisting only of characters and $*$.

Chapter 6

Generic functions

```
void Initgraph(void);
void Closegraph(void);
```

The first starts up and the last closes the X connection.
The functions that follow are self-explanatory and carry out the same operations as X11.

```
void CreatePixmap(Pixmap *pixmap, unsigned int w,
                  unsigned int h);
void ClearPixmapArea(Pixmap pixmap, int x, int y,
                     unsigned int w, unsigned int h);
void PutPixmap(Pixmap pixmap, Win_t *win, int x, int y,
               unsigned int w, unsigned int h);
void DrawText(Window w, char *txt, int px, int py);
void DrawLine(int ax, int ay, int bx, int by, Window w);
void DrawColorLine(int ax, int ay, int bx, int by,
                   int col, Drawable d);
void DrawPoint(int x, int y, Drawable d);
void DrawColorPoint(int x, int y, int col, Drawable d);
void DrawBox(Win_t *win, int x0, int y0, unsigned int h,
             unsigned int w, int d1);
```

The following functions manage the Graphics User Interface.

```
void InitGUI(GUI_t *gui);
```

sets the GUI.

```
void AddWindowToGUI(GUI_t *gui, Win_t *w);
```

adds a window to the GUI.

```
void RunGUI(GUI_t *gui, Win_t *Wstart);
```

executes the GUI.

```
void StopGUI(GUI_t *gui);
```

stops the GUI.

Chapter 7

A programming example

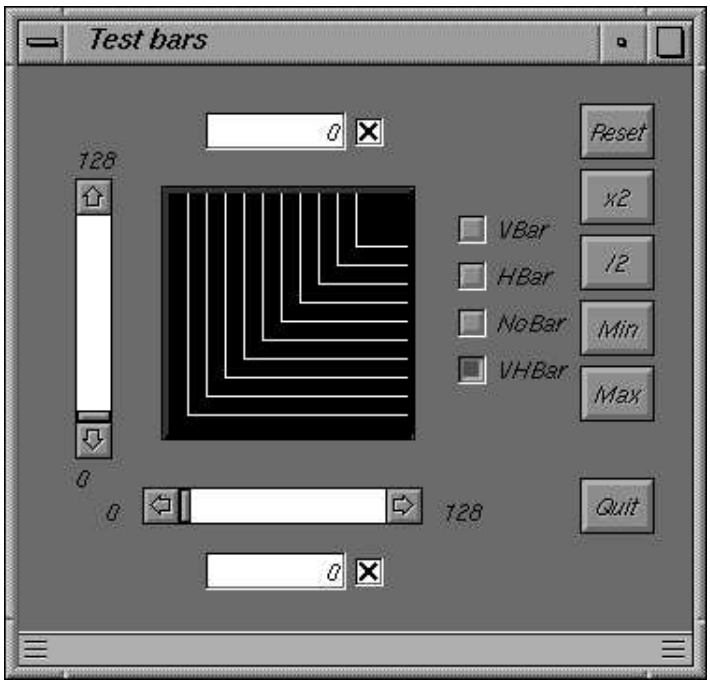

Figure 7.1: GUI for tbar program

The following code refers to tbar program; the GUI made in this program is shown in fig. 7.1.

```
/* tbar.c */
#include <X11/Xlib.h>
#include <X11/Xutil.h>

#include <stdio.h>
```

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

#include "xtools/xtools.h"
#include "xtools/test/tbar.h"

GUI_t inter;
Win_t Win;
Pixmap Pix;
int px,py,ppx,ppy;

int main(void) {
  Initgraph();
  InitGUI(&inter);
  CreateWin();
  AddWindowToGUI(&inter,&Win);
  RunGUI(&inter,&Win);
  printf("That's All Folks\n");
  return(0);
}

void CreateWin(void) {
int x;
  CreateWindow(&Win,200,100,360,300,color[0],color[13], mapGUI,
               "Test bars");

  AddTextBox(&Win, TBV, 100, 25, 75, 20, "0", __XT_INT, 5, TRUE);
  AddCheckBox(&Win, CKBV, 180, 28, 15, 15, TRUE);
  AddBar(&Win, BARV, 30, 60, 20, 150, 0, WSIDE, 1, 0);
  SetBarTxtLink(&Win, BARV, TBV);
  AddLabel(&Win, L1BARV, 30, 55, black, "128");
  AddLabel(&Win, L2BARV, 30, 225, black, "0");

  AddTextBox(&Win, TBH, 100, 260, 75, 20, "0", __XT_INT, 5, TRUE);
  AddCheckBox(&Win, CKBH, 180, 263, 15, 15, TRUE);
  AddBar(&Win, BARH, 66, 225, 150, 20, 0, WSIDE, 1, 0);
  SetBarTxtLink(&Win, BARH, TBH);
  AddLabel(&Win, L1BARH, 46, 243, black, "0");
  AddLabel(&Win, L2BARH, 226, 243, black, "128");

  AddCheckPoint(&Win, CKP, CKP1, 235, 80, 15, 15, FALSE);
  AddLabel(&Win, L1CKP, 255, 92, black, "VBar");
  AddCheckPoint(&Win, CKP, CKP2, 235, 105, 15, 15, FALSE);
  AddLabel(&Win, L2CKP, 255, 117, black, "HBar");
```

```
AddCheckPoint(&Win, CKP, CKP3, 235, 130, 15, 15, FALSE);
AddLabel(&Win, L3CKP, 255, 142, black, "NoBar");
AddCheckPoint(&Win, CKP, CKP4, 235, 155, 15, 15, TRUE);
AddLabel(&Win, L4CKP, 255, 167, black, "VHBar");

AddButton(&Win, RESET, 300,  20, 40, 30, "Reset");
AddButton(&Win, X2,    300,  55, 40, 30, "x2");
AddButton(&Win, DIV2,  300, 90, 40, 30, "/2");
AddButton(&Win, BMIN,  300, 125, 40, 30, "Min");
AddButton(&Win, BMAX,  300, 160, 40, 30, "Max");

AddButton(&Win, QUIT,  300, 220, 40, 30, "Quit");

px=80;
py=68;
CreateBorder(&Win, px, py, WSIDE, WSIDE, color[0]);
CreatePixmap(&Pix, PSIDE, PSIDE);
ClearPixmapArea(Pix, 0, 0, PSIDE, PSIDE);
ppx=0;
ppy=WSIDE;
for (x=10; x<=100; x=x+10)
   XDrawRectangle(dispGUI,Pix,gcGUI,x,x,PSIDE-2*x,PSIDE-2*x);
XCopyArea(dispGUI,Pix,Win.win,gcGUI,ppx,ppy,WSIDE,WSIDE,px,py);

SetButtonHelp(&Win,X2,"doubles the active bar step");
SetButtonHelp(&Win,DIV2,"halves the active bar step");
SetButtonHelp(&Win,RESET,"resets the bar values");

SetTextHelp(&Win,TBV,"enables to set the vertical bar value");
SetTextHelp(&Win,TBH,"enables to set the horizontal bar value");

SetCheckBoxBPScript(&Win, CKBV, input_en_dis);
SetCheckBoxBPScript(&Win, CKBH, input_en_dis);

SetTextLFScript(&Win,TBV,fun_v);
SetTextLFScript(&Win,TBH,fun_h);

SetCheckPointsBPScript(&Win, bar_selection);

SetBarMNScript(&Win, BARV, fun_v);
SetBarBRScript(&Win, BARV, fun_v);
SetBarMNScript(&Win, BARH, fun_h);
SetBarBRScript(&Win, BARH, fun_h);
```

```
  SetButtonBRScript(&Win, QUIT, quit);
  SetButtonBRScript(&Win, X2, x2);
  SetButtonBRScript(&Win, DIV2, div2);
  SetButtonBRScript(&Win, BMAX, max_bars);
  SetButtonBRScript(&Win, BMIN, min_bars);
  SetButtonBRScript(&Win, RESET, reset);
}

void quit(void) {
  StopGUI(&inter);
}

void fun_v(void) {
int aux;
  aux = GetBarValue(&Win, BARV);
  ppy=WSIDE-aux;
  XCopyArea(dispGUI,Pix,Win.win,gcGUI,ppx,ppy,
            WSIDE,WSIDE,px,py);
}

void fun_h(void) {
int aux;
  aux = GetBarValue(&Win, BARH);
  ppx=aux;
  XCopyArea(dispGUI,Pix,Win.win,gcGUI,ppx,ppy,
            WSIDE,WSIDE,px,py);
}

void x2(void) {
int id,step;
  id = GetCheckPointOn(&Win, CKP);
  switch (id) {
    case CKP1:
     step=GetBarStep(&Win, id) * 2.0;
     if (step<=WSIDE)
       SetBarStep(&Win, id, step );
    break;
    case CKP2:
     step=GetBarStep(&Win, id) * 2.0;
     if (step<=WSIDE)
       SetBarStep(&Win, id, step );
    break;
    case CKP3:
    break;
```

```
    case CKP4:
     step=GetBarStep(&Win, CKP1) * 2.0;
     if (step<=WSIDE)
       SetBarStep(&Win, CKP1, step );
     step=GetBarStep(&Win, CKP2) * 2.0;
     if (step<=WSIDE)
       SetBarStep(&Win, CKP2, step );
    break;
  }
}

void div2(void) {
int id,step;
  id = GetCheckPointOn(&Win, CKP);
  switch (id) {
    case CKP1:
      step=GetBarStep(&Win, id) / 2.0;
      if (step>=2)
       SetBarStep(&Win, id,step );
    break;
    case CKP2:
      step=GetBarStep(&Win, id) / 2.0;
      if (step>=2)
       SetBarStep(&Win, id,step );
    break;
    case CKP3:
    break;
    case CKP4:
      step=GetBarStep(&Win, CKP1) / 2.0;
      if (step>=2)
       SetBarStep(&Win, CKP1,step );
      step=GetBarStep(&Win, CKP2) / 2.0;
      if (step>=2)
       SetBarStep(&Win, CKP2,step );
    break;
   }
}

void reset(void) {
  SetCheckPointOn(&Win, CKP, CKP4);
  SetBarEnable(&Win, BARV, TRUE);
  SetBarValue(&Win, CKP1, 0 );
  fun_v();
  SetBarEnable(&Win, BARH, TRUE);
```

```
  SetBarValue(&Win, CKP2, 0 );
  fun_h();
  SetCheckBoxValue(&Win, CKBV, TRUE);
  SetCheckBoxEnable(&Win, CKBV, TRUE);
  SetTextInputEnable(&Win, TBV, TRUE);
  SetCheckBoxValue(&Win, CKBH, TRUE);
  SetCheckBoxEnable(&Win, CKBH, TRUE);
  SetTextInputEnable(&Win, TBH, TRUE);
}

void input_en_dis(void) {
  SetTextInputEnable(&Win, TBV,GetCheckBoxValue(&Win, CKBV) );
  SetTextInputEnable(&Win, TBH,GetCheckBoxValue(&Win, CKBH) );
}

void bar_selection(void) {
int id,aux;
  id = GetCheckPointOn(&Win, CKP);
  switch (id) {
    case CKP1:
      SetCheckBoxEnable(&Win, CKBV, TRUE);
      SetTextInputEnable(&Win, TBV, TRUE);
      SetBarEnable(&Win, BARV, TRUE);
      aux = GetBarValue(&Win, BARV);
      SetTextValue(&Win, TBV, &aux);
      SetCheckBoxEnable(&Win, CKBH, FALSE);
      SetTextInputEnable(&Win, TBH, FALSE);
      SetBarEnable(&Win, BARH, FALSE);
      break;
    case CKP2:
      SetCheckBoxEnable(&Win, CKBV, FALSE);
      SetTextInputEnable(&Win, TBV, FALSE);
      SetBarEnable(&Win, BARV, FALSE);
      SetCheckBoxEnable(&Win, CKBH, TRUE);
      SetTextInputEnable(&Win, TBH, TRUE);
      SetBarEnable(&Win, BARH, TRUE);
      aux = GetBarValue(&Win, BARH);
      SetTextValue(&Win, TBH, &aux);
      break;
    case CKP3:
      SetCheckBoxEnable(&Win, CKBV, FALSE);
      SetTextInputEnable(&Win, TBV, FALSE);
      SetBarEnable(&Win, BARV, FALSE);
      SetCheckBoxEnable(&Win, CKBH, FALSE);
```

```
      SetTextInputEnable(&Win, TBH, FALSE);
      SetBarEnable(&Win, BARH, FALSE);
      break;
    case CKP4:
      SetCheckBoxEnable(&Win, CKBV, TRUE);
      SetTextInputEnable(&Win, TBV, TRUE);
      SetBarEnable(&Win, BARV, TRUE);
      aux = GetBarValue(&Win, BARV);
      SetTextValue(&Win, TBV, &aux);
      SetCheckBoxEnable(&Win, CKBH, TRUE);
      SetTextInputEnable(&Win, TBH, TRUE);
      SetBarEnable(&Win, BARH, TRUE);
      aux = GetBarValue(&Win, BARH);
      SetTextValue(&Win, TBH, &aux);
      break;
    default:
      break;
  }
}

void min_bars(void) {
int id;
  id = GetCheckPointOn(&Win, CKP);
  switch (id) {
    case CKP1:
     SetBarValue(&Win, id, 0 );
     fun_v();
    break;
    case CKP2:
     SetBarValue(&Win, id, 0 );
     fun_h();
    break;
    case CKP3:
    break;
    case CKP4:
     SetBarValue(&Win, CKP1, 0 );
     fun_v();
     SetBarValue(&Win, CKP2, 0 );
     fun_h();
    break;
   }
}

void max_bars(void) {
```

```
int id;
  id = GetCheckPointOn(&Win, CKP);
  switch (id) {
    case CKP1:
     SetBarValue(&Win, id, WSIDE );
     fun_v();
    break;
    case CKP2:
     SetBarValue(&Win, id, WSIDE );
     fun_h();
    break;
    case CKP3:
    break;
    case CKP4:
     SetBarValue(&Win, CKP1, WSIDE );
     fun_v();
     SetBarValue(&Win, CKP2, WSIDE );
     fun_h();
    break;
   }
}

/* tbar.h */
#define WSIDE 128
#define PSIDE 2*WSIDE

#define BARV 1
#define BARH 2

#define L1BARV  1
#define L2BARV  2
#define L1BARH  3
#define L2BARH  4
#define L1CKP   5
#define L2CKP   6
#define L3CKP   7
#define L4CKP   8

#define TBV 1
#define TBH 2

#define CKBV 1
#define CKBH 2
```

```
#define CKP 1

#define CKP1 1
#define CKP2 2
#define CKP3 3
#define CKP4 4

#define QUIT   1
#define X2     2
#define DIV2   3
#define RESET  4
#define BMIN   5
#define BMAX   6

void CreateWin(void);
int main(void);
void quit(void);
void x2(void);
void div2(void);
void reset(void);
void input_en_dis(void);
void bar_selection(void);
void max_bars(void);
void min_bars(void);
void fun_v(void);
void fun_h(void);
```

Chapter 8

How to compile

This section describes how to compile, link and execute a program in C language with Unix Operating System. A program, to call the xtools library functions, should include the file xtools.h, that is, it should have the line

```
#include "xtools/xtools.h"
```

Then the following compiling and linking command must be given:

```
gcc -Idirectory_of_includes -o prova prova.c\\
    -Ldir_of_lib -lxtools -L/usr/X11/lib -lX11 -lXpm
```

Obviously, the available c compiler should be inserted in the place of the gcc. Now it is possible to execute the program with:

```
./prova
```

List of Figures

2.1 All the xtools's objects 4

7.1 GUI for tbar program 19

Bibliography

[JON89] O.Jones, Introduction to the Xwindow system, Prentice Hall (1989).

[NYE92] A.Nye, Xlib programming manual, O'Reilly and associates, Volume one (1992).

[NYOR92] A.Nye, T.O'Reilly, X Toolkit programming manual, O'Reilly and associates, Volume one (1992).

[xpm89] Xpm Format Library,
http://www.inria.fr/koala/lehors/xpm.html

[lehnah91] A. Le Hors, C. Nahaboo, XPM The X PixMap Format, (1991).

[XTOOLS99] S.Bonetti, G.Casciola, *xtools* library, Programming Guide - Version 1.0, (1999)
http://www.dm.unibo.it/∼casciola/html/xcmodel.html