NAME

Xinterface - a dynamically configurable process level interface to the X window system.

SYNOPSIS

Xinterface [*toolkitoption* ...] [<option> ...]

DESCRIPTION

The *Xinterface* program is a process level interface to the X window system. It allows a programmer having little or no knowledge of the X window system to create an interface for new applications, existing applications, and shell scripts. *Xinterface* is a skeleton client based on the Motif Resource Manager's support for the User Interface Language (UIL). The initial interface is specified by a compiled UIL file and augmented by command line arguments and default resource files (eg .Xdefaults). It is invoked interactively by a user or by an application as a shell process. After initialisation, *Xinterface* enters an event loop waiting for interaction from the user or the caller. Communication is by message strings over the standard input and output channels of *Xinterface*. Application messages to *Xinterface* may be:

POSTSCRIPT
: code for display on a canvas.

SETARG
: commands to modify object parameters which define the interface appearance and behaviour.

SETVAR
: commands to modify the value of variables stored in the object.

CALLBACK
: commands to invoke a method for an object.

Xinterface events are generated by application messages, user actions on interface objects, or messages from other objects. Messages from *Xinterface* to the application are generated by callback methods and are expected to be interpreted by the application.

OPTIONS

Xinterface accepts the following standard X Toolkit command line options:

bg	background colour
background	background colour
bw	width of border
borderwidth	width of border
bd	colour of border
bordercolour	colour of border
display	server to use
fg	foreground colour
foreground	foreground colour
fn	font name
font	font name
geometry	size and position

–iconic	start as an icon
–name	name of application
–reverse	reverse video
–rv	reverse video
+rv	reverse video off
–selectionTimeout	selection timeout
–synchronous	synchronous debug mode on
+synchronous	synchronous debug mode off
–title	title of application
–xrm	resource

In addition to these arguments, Xinterface supports the following options:

–help This option indicates that a brief summary of the allowed options should be printed on the standard error.

–uid *<file>*
This option specifies a compiled User Interface Language (**UIL**) file which contains all or part of the definition of the interface. If no **–uid** arguments are used then the documentation interface is used.

–arg "*<communication type> <object_name> <arguments>*"
This option denotes a communication string which will be made available for retrieval to the interface under the *object_name* specified. Valid communication types are:

POSTSCRIPT
SETARG
SETVAR
CALLBACK

Object names are as specified in the UIL file. The arguments to the *POSTSCRIPT* command are simply valid postscript. The arguments to the *SETARG* command are the name of the resource to be set and the value to set it to. The arguments to the *SETVAR* command are the name of the variable to be set and the value to set it to. The argument to the *CALLBACK* command is the name of the callback method to invoke.

–input This option specifies that communication with the caller is desired.

ENVIRONMENT

The environment variable *XINTERFACE_PATHS* (in the same format as PATH) is appended to a set of default paths. One of the entries in this combined set must be the path to the compiled UIL file specified with the –uid option.

The environment variable *POSTSCRIPTLIB* may be used to cause the postscript interpreter to access an alternate macro file. If this is not set a default library is used.

The environment variable *HOME* is used to access the .postscript file which augments the macro library,

The environment variable *HOME* is used to access the .Xdefaults resource file.

OVERVIEW

Xinterface is a high-level interface tool. UIL (User Interface Language) provides a mechanism to describe the "look and feel" of an interface. This UIL specification is compiled into a UID (User Interface Description) format which can be used in conjunction with the Motif Resource Manager (MRM) to provide that interface.

A series of MRM procedures allow Xinterface to load in the UID files and install the base hierarchy. The UIL/MRM section which follows discusses the facilities provided by the Motif Resource Manager in conjunction with UIL. It is shown how to incorporate Xinterface functions into a UIL specification. For a more detailed look at UIL the UIL Programmers Guide is included in the "Building Xinterfaces" section of this documentation.

In addition to installing the UID interface, however, Xinterface provides additional functionality. Xinterface provides the following key features:

- accommodates dynamic command line argument access.
- dynamic modification by the calling process.
- control over internal modification.
- support for inter-object communication.
- event communication to the calling process.
- callback triggering from the calling process.
- independant control.
- run time loading of text arguments from files.

Xinterface is event driven. That is, interaction on the part of the user will generate an X event. The handling of this event will depend on the UID specification provided to Xinterface. For example: pressing a mouse button inside of a PushButton object will cause an X event which will be handled as specified by the XmNactivateCallback procedure list for that object. This allows an interface specification to capture the user's actions and respond accordingly.

Since Xinterface, in general, only provides an interface it is often necessary to communicate the user's actions back to the underlying program. This is accomplished through UNIX pipes. Thus the calling program must provide some means of listening to its interface. The section on "Existing Xinterfaces" demonstrates how this can be done.

Of course, it is also necessary to accommodate communication from the caller to Xinterface. This allows the caller to cause changes in the interface to inform the user of the state of the process. Xinterface supports this as an input event. Input events cause either specification changes, or callback triggering.

Xinterface also provides support for utilising the object-oriented nature of X and UIL. A callback function called "xinterface_iprintf" allows one object to internally print a string to another object.

Xinterface also provides dynamic command line argument access. That is, an object may request its arguments at any time during execution. This allows the interface to dynamically configure itself to the users specifications. Command line arguments are handled almost identically to inter-object communications or caller->Xinterface communications. The exception being that the arguments:

- must be requested (via xinterface_args).
- are still available for later access.

The communication protocol in all cases is as follows:

```
communication ::=
        communication_type_1
```

```
        | communication_type_2
        | communication_type_3
        | communication_type_4

    communication_type_1 ::=
        CALLBACK  object_name  callback_argument

    communication_type_2 ::=
        SETARG  object_name  argument_name  argument_value

    communication_type_3 ::=
        SETVAR  object_name  variable_name  variable_value

    communication_type_4 ::=
        POSTSCRIPT  object_name  postscript_code

    object_name ::=
        {the registered name of the object}

    callback_argument ::=
        {the name of the callback reason for the object}

    argument_name ::=
        {the name of the argument to set in the object}

    argument_value ::=
        {the new value for argument_name}

    variable_name ::=
        {the name of the variable to set in the object}

    variable_value ::=
        {the new value for variable_name}

    postscript_code ::=
       {the postscript code to process}
```

In all cases a communication is passed as a single string with the arguments delimited by white space. Xinterface parses the string into its components.

Mesages can only be sent to objects who have registered themselves. This provides a public/private feature for objects.

Since objects requests command line arguments there need not be a correspondence between the actual or registered object name and the name used in the command line arguments. The object whose name corresponds with the name used for retrieval will receive the arguments. Thus one object may cause another to receive its command line arguments.

To enable its arguments to be used an object must register itself with xinterface_register_widget(object_name) and then any object may call xinterface_args(object_name) to retrieve all arguments for object_name. These procedures may be invoked by the createCallback for startup initialisation, or by any other callback for dynamic user defined modification.

External Modification

Xinterface provides a method by which the calling process can dynamically modify the interface. Communication is through UNIX pipes (Xinterface receives events through pipes specified with the "-input" argument), so the calling process can send messages to Xinterface by writing over this pipe.

Internal Modification

Xinterface provides a method by which an active object can dynamically modify any other object. Inter-object communication is performed through the callback function "xinterface_iprintf", so the sender must be active. A message can be sent to any registered widget (to its registered address).

Communication with Caller

The calling process may establish communication with Xinterface in either or both directions.

The callback function "xinterface_fprintf" will print a string to Xinterface's stdout; the calling process may use this mechanism to receive messages from Xinterface. The caller need only read the messages coming over the pipe. Most languages provide some mechanism to handle the event of incoming messages. Xgmacs (see Existing Xinterfaces) demonstrates the use of this communication system.

Communication from the calling process may be established via the "-input" argument. Once an input pipe has been specified, messages can be written over it and will generate events for Xinterface.

Independant Control

Xinterface provides a number of callback routines which allow a degree of independance from the caller. When an interface object is activated the user is indicating that some action is desired. Provided that the action to be taken involves modifying the interface, or can be performed by the operating system, Xinterface can proceed without the necessity of notifying the calling process.

This is, in fact, a powerful enough mechanism to allow some simple Xinterfaces to be written as standalone programs. Xmemo (see 'Existing Xinterfaces') is an example of such a program.

When carried to a lesser extreme this allows the interface to act in parallel to the caller performing some of the simpler tasks and communicating to the caller only when needed.

In particular, Xinterface allows dynamic modification of the interface. Thus if the action to be taken when an object is pressed is merely to modify the interface then the caller need not be notified.

User Configurability

The X window system has attempted to "provide mechanism not policy". To this end many facilities have been provided for allowing the user to specify their preferences as to the "look and feel" of an interface.

Xinterface has incorporated these ideas into its design. All interfaces run by Xinterface belong to the class "XInterface". By default the name of the client for resource lookup is "Xinterface"; however this name may be overridden by passing the -name argument to Xinterface. Indeed this is how each of the demos work. The resource files are then used to define the "look and feel" of the interface. Only the functionality of the interface need be specified in the UIL description. Thus a user may use the resource facility to control the appearance (but not the functionality) of all XInterface interfaces.

For example, by placing the following line:

```
Xdialog*.dialog_quit_button.mappedWhenManaged: true
```

in your .Xdefaults file you can cause Xdialog to map (make visible) the QUIT button.

Examples Using Xinterface

XDIALOG

NAME

Xdialog - X Window Systems dialog box client.

SYNOPSIS

Xdialog [Xinterface arguments]

DESCRIPTION

Xdialog is an *Xinterface* dialog box utility. It is intended for use as a prompt/retrieve mechanism. Xdialog is simply the following shell script:

```
#! /bin/csh
Xinterface  $argv:q -name Xdialog -uid Xdialog.uid
```

This may be used to prompt/retrieve text arguments from an existing application or shell script, or from the shell by placing the call in back-quotes. The latter of these is useful for prompt/retrieval of arguments for programs which need arguments but are started from a window manager menu.

OPTIONS

Xdialog simply passes all arguments on to *Xinterface*.

X DEFAULTS

Resource values for this program are read for the client name *Xdialog* and client class *XInterface*. The default values are defined in the file */home/resstaff/lib/X11/app-defaults/XInterface* but may be redefined in *$HOME/.Xdefaults*.

The current set of defaults is as follows:

```
Xdialog*.background: black
Xdialog*.dialog_label.labelString: Dialog Box
Xdialog*.dialog_label.x: 160
Xdialog*.dialog_label.y: 15
Xdialog*.dialog_ok_button.background: green
Xdialog*.dialog_ok_button.x: 15
Xdialog*.dialog_ok_button.y: 120
Xdialog*.dialog_ok_button.labelString: _OK_
Xdialog*.dialog_quit_button.background: red
Xdialog*.dialog_quit_button*mappedWhenManaged: true
Xdialog*.dialog_quit_button.x: 370
Xdialog*.dialog_quit_button.y: 120
Xdialog*.dialog_quit_button.labelString: QUIT
```

```
Xdialog*.dialog_editor.x: 15
Xdialog*.dialog_editor.y: 50
Xdialog*.dialog_editor_text.foreground: black
Xdialog*.dialog_editor_text.background: white
Xdialog*.dialog_editor_text.width: 365
```

UIL FILE

This gives a simple example of how to construct a UIL description for use with Xinterface .

The actual code begins with the statement module xinterface. This is an arbitrary name and is not used for anything.

Notice the "include file" statement. This causes the named file to be read in and compiled as if placed in line. This file contains the declaration of Xinterface callback procedures.

Note that the base of the hierarchy: the bulletin board on which everything is posted is named "xinterface_main". This is necessarily true.

Besides "xinterface_main" there are five other objects:
- a LABEL
- an OK button
- a QUIT button
- a framed BulletinBoard
- a TEXT widget

These are all posted on the main bulletin board.

The "dialog_quit_button" and the "dialog_editor_text" objects both register themselves upon creation. This registered name can later be used to send a message to the object by either another object or the calling process. Both of these schemes are intended for Xdialog. The inter-object communication scheme is used by the OK button which must send a message to the TEXT widget to print its contents (the OK button does not have access to the contents directly). Communication from the calling process may be used to invoke the QUIT button when the desired information is obtained.

Any feature of a registered object can be dynamically modified and any callback associated with the object can be invoked.

If you start Xdialog with the –input option then you can demonstrate this feature. The QUIT button is mapped by default. To unmap it simply type the following after starting Xdialog with –input:

```
SETARG dialog_quit_button mappedWhenManaged false
```

Even when the button is unmapped its functionality is fully defined and the button can be used simply by invoking its activate callback. Type the following into the xterm:

```
CALLBACK dialog_quit_button activateCallback
```

The Xdialog process will terminate and the xterm will disappear.

The dialog box is intended to be used as a prompt-retrieve device. When text is typed into the text window will be returned to the caller when the dialog_editor_text object is activated (see the dialog_editor object -- notice that the XmText object that it controls registers itself as "dialog_editor_text").

The "dialog_editor_text" object can be activated in three ways:

1 – by default a single line XmText object activates when a return key is pressed in it.

2 – pressing the OK button causes the dialog_ok_button activateCallback to execute; but this

calls xinterface_iprintf with a CALLBACK trigger for the "dialog_editor_text" object which will now be activated.

3 – if the -input argument was used at startup the caller can send the message:
"CALLBACK dialog_editor_text activateCallback"
which will cause the "dialog_editor_text" object to activate.

In some cases it may be desirable to have Xdialog exit whenever a the TEXT widget is activated. This would allow Xdialog to be used to prompt/retrieve information for a passive client. This is accomplished by the "xinterface_args" call in the activate callback for "dialog_editor_text". "xinterface_args" causes dynamic retrieval of command line arguments -- by providing a command line argument: -arg "CALLBACK dialog_quit_button activateCallback" the QUIT button will be activated whenever the TEXT widget is activated.

```
! ============================================================
! Xdialog.uil

module xinterface
      names = case_sensitive

include file
          'Xinterface.uil';

object
   xinterface_main : XmBulletinBoard {
          controls {
             XmLabel                  dialog_label;
             XmPushButton    dialog_ok_button;
             XmPushButton    dialog_quit_button;
             XmFrame                  dialog_editor;
          };
   };



object
   dialog_label : XmLabel {
   };



object
   dialog_ok_button : XmPushButton {
          callbacks {
             XmNactivateCallback = procedures {
                xinterface_iprintf("\
                            "CALLBACK dialog_editor_text activateCallback"");
             };
          };
   };
```

```
object
   dialog_quit_button : XmPushButton {
         callbacks {
            XmNactivateCallback = procedures {
               xinterface_exit('"0"');
            };
            MrmNcreateCallback = procedures {
               xinterface_register_widget('"dialog_quit_button"');
            };
         };
   };



object
   dialog_editor : XmFrame {
         controls {
            XmBulletinBoard {
               controls {
                     XmText dialog_editor_text;
                  };
            };
         };
   };



object
   dialog_editor_text : XmText {
         callbacks {
            MrmNcreateCallback = procedures {
               xinterface_register_widget('"dialog_editor_text"');
            };
            XmNactivateCallback = procedures {
               xinterface_TextGetString('"');
               xinterface_fprintf('stdout, "\n"');
                  xinterface_args('"dialog_quit_button"');
            };
         };
   };


end module;
```

FILES

/home/resstaff/bin/X11/Xdialog /home/resstaff/lib/X11/app-defaults/XInterface

SEE ALSO

X(1), Xinterface(1), uil(7)

XMEMO

NAME

Xmemo - Xinterface memo editor utility.

SYNOPSIS

Xmemo [Xinterface arguments]

DESCRIPTION

Xmemo is an *Xinterface* memo editor utility. It is allows the user to edit text and send it to themselves in the mail at any given time. A single line text editor allows the entry of a time specification. The time specification is in the Unix *at* format. A multi line text editor allows a memo to be entered. When the message is ready to send the send button can be activated and the memo will be entered in the "at" queue. If the time specification is in error the *at* will issue an appropriate message. When the message is sent *at* will inform the user that a queue entry has been made. If the user wishes to cancel the operation during editing they simply press the quit button causing *Xmemo* to terminate.

Xmemo is simply the following shell script:

```
#! /bin/csh
Xinterface $argv:q -name Xmemo -uid Xmemo.uid
```

OPTIONS

All options are passed to *Xinterface*.

X DEFAULTS

Resource values for this program will be read for client name *Xmemo* and client class *XInterface*. The default values are defined in the file */home/resstaff/lib/X11/app-defaults/XInterface* but may be redefined in *$HOME/.Xdefaults*.

The current set of defaults is as follows:

```
Xmemo*.background: black
Xmemo*.xinterface_main.XmLabel.x: 15
Xmemo*.xinterface_main.XmLabel.y: 10
Xmemo*.xinterface_main.XmLabel.labelString: Memo Editor
Xmemo*.memo_send_button.background: gray
Xmemo*.memo_send_button.x: 300
Xmemo*.memo_send_button.y: 10
Xmemo*.memo_send_button.labelString: SEND
Xmemo*.memo_quit_button.background: red
Xmemo*.memo_quit_button.x: 400
Xmemo*.memo_quit_button.y: 10
Xmemo*.memo_quit_button.labelString: QUIT
Xmemo*.memo_time_editor.x: 15
Xmemo*.memo_time_editor.y: 60
Xmemo*.memo_time_editor.XmBulletinBoard.XmLabel.x: 15
Xmemo*.memo_time_editor.XmBulletinBoard.XmLabel.y: 15
Xmemo*.memo_time_editor.XmBulletinBoard.XmLabel.labelString: time
Xmemo*.memo_time_editor_text.background: white
Xmemo*.memo_time_editor_text.foreground: black
```

```
Xmemo*.memo_time_editor_text.x: 60
Xmemo*.memo_time_editor_text.y: 15
Xmemo*.memo_time_editor_text.width: 365
Xmemo*.memo_message_editor.x: 15
Xmemo*.memo_message_editor.y: 120
Xmemo*.memo_message_editor_text.background: white
Xmemo*.memo_message_editor_text.foreground: black
Xmemo*.memo_message_editor_text.x: 60
Xmemo*.memo_message_editor_text.y: 15
Xmemo*.memo_message_editor_text.scrolledHorizontal: true
Xmemo*.memo_message_editor_text.scrolledVertical: true
Xmemo*.memo_message_editor_text.editMode: MULTI_LINE_EDIT
Xmemo*.memo_message_editor_text.height: 200
Xmemo*.memo_message_editor_text.width: 400
```

UIL FILE

This gives a simple example of how to construct a UIL description for an interface for Xinterface.

The Memo Editor is a standalone program. That is, the interface is self contained -- all the functionality is contained in the interface.

The actual code begins with the statement module xinterface. This is an arbitrary name and is not used for anything.

Notice the "include file" statement. This causes the named file to be read in and compiled as if placed in line. This file contains the declaration of Xinterface callback procedures.

Note that the base of the hierarchy: the bulletin board on which everything is posted is named "xinterface_main". This is necessarily true.

Besides "xinterface_main" there are five other objects:
- a LABEL
- a SEND button
- a QUIT button
- a framed TEXT window to edit the time string.
- a framed TEXT window to edit the memo message.

These are all posted on the main bulletin board.

The "xinterface_system" procedure is used to send the message. The UNIX 'at' command (execute a job AT a given time) is used to schedule the memo. The 'at' command is given the time from the time editor and is given a script to execute. The script consists of mailing the memo text from the memo editor to the user. Inter-object communication is used to ensure that the text from both the time editor and the memo editor has been written (using tmp files). These are then used as input to the script.

Absolutely NO format checking is performed on the time format. The backquote notation is used both to get the time from the file and to find the user name.

After 'at' command is submitted the temporary files are deleted.

```
! ============================================================
! Memo Editor
```

```
module xinterface
    names = case_sensitive

include file
    'Xinterface.uil';

object
  xinterface_main : XmBulletinBoard {
    controls {
      XmLabel {
      };
      XmPushButton      memo_send_button;
      XmPushButton      memo_quit_button;
      XmFrame           memo_time_editor;
      XmFrame           memo_message_editor;
    };
  };



object
  memo_send_button : XmPushButton {
    callbacks {
      XmNactivateCallback = procedures {
        xinterface_iprintf('\
            "CALLBACK memo_time_editor_text activateCallback"');
        xinterface_iprintf('\
            "CALLBACK memo_message_editor_text activateCallback"');
        xinterface_system('"at -c 'cat Xmemo_time' <<EOIN1\n\
            /usr/ucb/mail -v -s memo 'whoami' <<EOIN2\n\
              'cat Xmemo_text' \n\
EOIN2\n\
EOIN1\n"');
        xinterface_args('"memo_quit_button"');
      };
    };
  };


object
  memo_quit_button : XmPushButton {
    callbacks {
      MrmNcreateCallback = procedures {
        xinterface_register_widget('"memo_quit_button"');
      };
      XmNactivateCallback = procedures {
        xinterface_system('"/bin/rm -f Xmemo_time"');
        xinterface_system('"/bin/rm -f Xmemo_text"');
        xinterface_exit('"0"');
      };
    };
  };
```

```
object
   memo_time_editor : XmFrame {
      controls {
         XmBulletinBoard {
            controls {
               XmLabel {
               };
               XmText memo_time_editor_text;
            };
         };
      };
   };

object
   memo_time_editor_text : XmText {
      callbacks {
         MrmNcreateCallback = procedures {
            xinterface_register_widget("'memo_time_editor_text"');
         };
         XmNactivateCallback = procedures {
            xinterface_system("'/bin/rm -f Xmemo_time"');
            xinterface_TextGetString("'Xmemo_time"');
         };
      };
   };


object
   memo_message_editor : XmFrame {
      controls {
         XmBulletinBoard {
            controls {
               XmScrolledText memo_message_editor_text;
            };
         };
      };
   };


object
   memo_message_editor_text : XmScrolledText {
      callbacks {
         MrmNcreateCallback = procedures {
            xinterface_register_widget("'memo_message_editor_text"');
         };
         XmNactivateCallback = procedures {
            xinterface_system("'/bin/rm -f Xmemo_text"');
            xinterface_TextGetString("'Xmemo_text"');
         };
      };
   };

end module;
```

FILES

/home/resstaff/bin/X11/Xmemo
/home/resstaff/lib/X11/app-defaults/XInterface

SEE ALSO

X(1), Xinterface(1), uil(7)

XBIFF

NAME

Xbiff - Xinterface mailbox flag

SYNOPSIS

Xbiff [-Xinterface options] [-options]

DESCRIPTION

Xbiff is an *Xinterface* utility which monitors for incoming mail. When no mail is present a suitable icon is displayed in a window which is insensitive. When mail is present the icon changes to indicate that mail has arrived and the window becomes sensitive to mouse presses. Pressing a mouse button when the Xbiff button window is sensitive issues a command to Unix intended to deal with the mail. The default system command is "gmacs -f rmail -f rmail-summary" but this may be redefined in your .Xdefaults file.

OPTIONS

All arguments are forwarded to *Xinterface* except the following:

–help display the usage of the command.

–test next argument denotes the test to perform on the mail file. Default is */usr/ucb/mail –v –e*

–update next argument denotes how often in seconds to check for mail. Default is 60.

–bell next argument denotes the string to print when mail is received. Default is "^G" which rings the bell.

X DEFAULTS

Resource values for this program will be read for client name *Xbiff* and client class *XInterface*. The default values are defined in the file */home/resstaff/lib/X11/app-defaults/XInterface* but may be redefined in *$HOME/.Xdefaults*.

The current set of defaults is as follows:

```
Xbiff*.flagdown.x: 0
Xbiff*.flagdown.y: 0
Xbiff*.flagdown.width: 50
Xbiff*.flagdown.height: 50
Xbiff*.flagdown.backgroundPixmap: /usr/include/X11/bitmaps/flagdown
Xbiff*.flagdown.pushButtonEnabled: true
Xbiff*.flagdown.sensitive: false
Xbiff*.flagup.x: 0
Xbiff*.flagup.y: 0
Xbiff*.flagup.width: 50
```

```
Xbiff*.flagup.height: 50
Xbiff*.flagup.backgroundPixmap: /usr/include/X11/bitmaps/flagup
Xbiff*.flagup.pushButtonEnabled: true
Xbiff*.flagup.sensitive: true
Xbiff*.flagup.mappedWhenManged: false
Xbiff*.flagup.userData: gmacs -f rmail -f rmail-summary
```

UIL FILE

```
! ============================================================
! Xbiff.uil

module xinterface
      names = case_sensitive

include file
       'Xinterface.uil';

object
   xinterface_main : XmBulletinBoard {
      callbacks {
         MrmNcreateCallback = procedures {
            xinterface_register_widget('"main"');
            xinterface_create_widget('"bitmap flagdown"');
            xinterface_iprintf('"SETVAR flagdown other flagup"');
            xinterface_create_widget('"bitmap flagup"');
            xinterface_iprintf('"SETVAR flagup other flagdown"');
         };
      };
   };

object
   bitmap : XmDrawnButton {
      callbacks {
         XmNactivateCallback = procedures {
            xinterface_fprintf('stdout, \
                        "activateCallback %s\n", \
                        Resources().userData');
         };
         XmNhelpCallback = procedures {
            xinterface_iprintf('"SETARG %s mappedWhenManaged true", \
                        Registry().other');
            xinterface_iprintf('"SETARG %s mappedWhenManaged false", \
                        Registry().name');
         };
         XmNexposeCallback = procedures {
            xinterface_fprintf('stdout, \
                        "EXPOSE\n"');
         };
      };
   };

end module;
```

FILES
/home/resstaff/lib/X11/app-defaults/XInterface

SEE ALSO
X(1), Xinterface(1), xbiff(1), mail(1).

XCHEZ
NAME
Xchez - Chez Scheme with X Window Systems support via. Xinterface.

SYNOPSIS
Xchez <Xfile.ss> [Chez Scheme arguments ...]

DESCRIPTION
Xchez is a shell script which starts Chez Scheme with the interface defined in *Xfile.ss*. The Xfile.ss file is simply a Chez Scheme definition as follows:

```
(define process-string "Xinterface [–options ... ]")
```

The Xchez shell script uses this as the first argument to Chez Scheme. The second argument is another Scheme file which uses the definition of process–string to start Xinterface and establish the communication ports. Communication with *Xinterface* is through the input/output ports *X-IN*, *X-OUT* respectively.

CHEZ SCHEME OPTIONS
The Chez Scheme arguments are passed on without processing.

XINTERFACE OPTIONS
All options to Xinterface are passed on without processing.

X DEFAULTS
Resource values for this program are read for the client name specified as a *–name* argument to *Xinterface* with the client class name *XInterface*. By default the client name is *Xinterface*. The default values are defined in the file */home/resstaff/lib/X11/app-defaults/XInterface* but may be redefined in *$HOME/.Xdefaults*.

FILES
/home/resstaff/bin/X11/Xchez /home/resstaff/lib/X11/app-defaults/XInterface

SEE ALSO
X(1), Xinterface(1), scheme(1)

XCALC
NAME
Xcalc - a simple calculator client based on Chez Scheme and Xinterface.

SYNOPSIS

Xcalc

DESCRIPTION

Xcalc is an *Xinterface* calculator utility. It is intended as an example of the use of Xinterface from Chez Scheme and uses the Xchez utility. Xcalc is simply the following shell script:

```
#! /bin/csh
Xchez /home/resstaff/lib/X11/Xinterface/Xcalcdef.ss\
    /home/resstaff/lib/X11/Xinterface/Xcalc.ss\
    > /dev/null
```

OPTIONS

Xcalc has no options.

X DEFAULTS

Resource values for this program are read for the client name *Xcalc* and client class *XInterface*. The default values are defined in the file */home/resstaff/lib/X11/app-defaults/XInterface* but may be redefined in *$HOME/.Xdefaults*.

The current set of defaults is as follows:

```
Xcalc*.xinterface_main.background: black
Xcalc*.calc_text.y: 30
Xcalc*.calc_text.x: 20
Xcalc*.calc_text.editable: false
Xcalc*.calc_1_button.labelString: 1
Xcalc*.calc_1_button.width: 20
Xcalc*.calc_1_button.height: 20
Xcalc*.calc_1_button.x: 10
Xcalc*.calc_1_button.y: 100
Xcalc*.calc_2_button.labelString: 2
Xcalc*.calc_2_button.width: 20
Xcalc*.calc_2_button.height: 20
Xcalc*.calc_2_button.x: 50
Xcalc*.calc_2_button.y: 100
Xcalc*.calc_3_button.labelString: 3
Xcalc*.calc_3_button.width: 20
Xcalc*.calc_3_button.height: 20
Xcalc*.calc_3_button.x: 90
Xcalc*.calc_3_button.y: 100
Xcalc*.calc_4_button.labelString: 4
Xcalc*.calc_4_button.width: 20
Xcalc*.calc_4_button.height: 20
Xcalc*.calc_4_button.x: 10
Xcalc*.calc_4_button.y: 150
Xcalc*.calc_5_button.labelString: 5
Xcalc*.calc_5_button.width: 20
Xcalc*.calc_5_button.height: 20
Xcalc*.calc_5_button.x: 50
Xcalc*.calc_5_button.y: 150
```

```
Xcalc*.calc_6_button.labelString: 6
Xcalc*.calc_6_button.width: 20
Xcalc*.calc_6_button.height: 20
Xcalc*.calc_6_button.x: 90
Xcalc*.calc_6_button.y: 150
Xcalc*.calc_7_button.labelString: 7
Xcalc*.calc_7_button.width: 20
Xcalc*.calc_7_button.height: 20
Xcalc*.calc_7_button.x: 10
Xcalc*.calc_7_button.y: 200
Xcalc*.calc_8_button.labelString: 8
Xcalc*.calc_8_button.width: 20
Xcalc*.calc_8_button.height: 20
Xcalc*.calc_8_button.x: 50
Xcalc*.calc_8_button.y: 200
Xcalc*.calc_9_button.labelString: 9
Xcalc*.calc_9_button.width: 20
Xcalc*.calc_9_button.height: 20
Xcalc*.calc_9_button.x: 90
Xcalc*.calc_9_button.y: 200
Xcalc*.calc_0_button.labelString: 0
Xcalc*.calc_0_button.width: 20
Xcalc*.calc_0_button.height: 20
Xcalc*.calc_0_button.x: 10
Xcalc*.calc_0_button.y: 250
Xcalc*.calc_lp_button.labelString: (
Xcalc*.calc_lp_button.width: 20
Xcalc*.calc_lp_button.height: 20
Xcalc*.calc_lp_button.x: 50
Xcalc*.calc_lp_button.y: 250
Xcalc*.calc_rp_button.labelString: )
Xcalc*.calc_rp_button.width: 20
Xcalc*.calc_rp_button.height: 20
Xcalc*.calc_rp_button.x: 90
Xcalc*.calc_rp_button.y: 250
Xcalc*.calc_sp_button.labelString:
Xcalc*.calc_sp_button.width: 20
Xcalc*.calc_sp_button.height: 20
Xcalc*.calc_sp_button.x: 10
Xcalc*.calc_sp_button.y: 300
Xcalc*.calc_eq_button.labelString: =
Xcalc*.calc_eq_button.width: 20
Xcalc*.calc_eq_button.height: 20
Xcalc*.calc_eq_button.x: 50
Xcalc*.calc_eq_button.y: 300
Xcalc*.calc_add_button.labelString: +
Xcalc*.calc_add_button.width: 20
Xcalc*.calc_add_button.height: 20
Xcalc*.calc_add_button.x: 140
Xcalc*.calc_add_button.y: 100
Xcalc*.calc_sub_button.labelString: -
Xcalc*.calc_sub_button.width: 20
Xcalc*.calc_sub_button.height: 20
```

```
Xcalc*.calc_sub_button.x: 140
Xcalc*.calc_sub_button.y: 150
Xcalc*.calc_mul_button.labelString: *
Xcalc*.calc_mul_button.width: 20
Xcalc*.calc_mul_button.height: 20
Xcalc*.calc_mul_button.x: 140
Xcalc*.calc_mul_button.y: 200
Xcalc*.calc_div_button.labelString: /
Xcalc*.calc_div_button.width: 20
Xcalc*.calc_div_button.height: 20
Xcalc*.calc_div_button.x: 140
Xcalc*.calc_div_button.y: 250
Xcalc*.calc_off_button.labelString: off
Xcalc*.calc_off_button.x: 130
Xcalc*.calc_off_button.y: 300
```

UIL FILE

```
! ============================================================
! Xcalc UIL file

module xinterface
      names = case_sensitive

include file
      'Xinterface.uil';

object
   xinterface_main : XmBulletinBoard {
      controls {
         XmText calc_text;
         XmPushButton calc_off_button;
      };
      callbacks {
         MrmNcreateCallback = procedures {
            xinterface_create_widget("'calc_button calc_0_button"');
            xinterface_create_widget("'calc_button calc_1_button"');
            xinterface_create_widget("'calc_button calc_2_button"');
            xinterface_create_widget("'calc_button calc_3_button"');
            xinterface_create_widget("'calc_button calc_4_button"');
            xinterface_create_widget("'calc_button calc_5_button"');
            xinterface_create_widget("'calc_button calc_6_button"');
            xinterface_create_widget("'calc_button calc_7_button"');
            xinterface_create_widget("'calc_button calc_8_button"');
            xinterface_create_widget("'calc_button calc_9_button"');
            xinterface_create_widget("'calc_button calc_sp_button"');
            xinterface_create_widget("'calc_button calc_eq_button"');
            xinterface_create_widget("'calc_button calc_lp_button"');
            xinterface_create_widget("'calc_button calc_rp_button"');
            xinterface_create_widget("'calc_button calc_add_button"');
            xinterface_create_widget("'calc_button calc_sub_button"');
            xinterface_create_widget("'calc_button calc_mul_button"');
            xinterface_create_widget("'calc_button calc_div_button"');
         };
```

```
        };
    };


object
    calc_off_button : XmPushButton {
        callbacks {
            XmNactivateCallback = procedures {
                xinterface_exit('"0"');
            };
        };
    };


object
    calc_button : XmPushButton {
        callbacks {
            XmNactivateCallback = procedures {
                xinterface_fprintf('stdout, "%s", Resources().labelString');
            };
        };
    };



object
    calc_text : XmText {
        callbacks {
            MrmNcreateCallback = procedures {
                xinterface_register_widget('"calc_text"');
            };
        };
    };


end module;
```

FILES

/home/resstaff/bin/X11/Xcalc /home/resstaff/lib/X11/app-defaults/XInterface

SEE ALSO

X(1), Xinterface(1), Xchez(1)

XCANVAS

NAME

Xcanvas - a general interface utility which is dynamically configurable.

SYNOPSIS

Xcanvas [–options ...]

DESCRIPTION

Xcanvas is a shell script which uses *Xinterface* to provide a process level configurable Motif style interface. The UIL definition provides a definition for each UIL object (accept gadgets) which may be created and modified dynamically.

Xcanvas is simply the following shell script:

```
#! /bin/csh
Xinterface $argv:q -name Xcanvas -uid Xcanvas.uid -input
```

Each composite widget's helpCallback is defined to call *xinterface_create_widget* with the userData string. This is also true for xinterface_main which is registered as "main". By setting the userData resource of an existing composite widget and then calling the help callback the named widget will created. For example, the following sequence of communication strings:

```
SETARG main userData SelectionDialog sd1
CALLBACK main helpCallback
SETARG sd1 listItems item1,item2,item3,item4,item5
SETARG sd1 listItemCount 5
SETARG sd1 userData PushButton pb1
CALLBACK sd1 helpCallback
```

will cause a SelectionDialog to pop up, place the five named items in it and display them, and add a PushButton to the SelectionDialog. Try it.

Some callabck reasons are provided with special procedures; however in general a callback will simply print out the reason name and the object name. The createCallback as well as the mapCallback print only the callback reason, since the object name is unknown until creation is complete.

OPTIONS

Xcanvas passes all options to Xinterface.

X DEFAULTS

Resource values for this program are read for the client name *Xcanvas* and client class *XInterface*. The default values are defined in the file */home/resstaff/lib/X11/app-defaults/XInterface* but may be redefined in *$HOME/.Xdefaults*.

The current set of defaults is as follows:

```
Xcanvas*.xinterface_main.height: 500
Xcanvas*.xinterface_main.width: 500
```

UIL FILE

```
! ============================================================
! Xcanvas UIL file

module xinterface
    names = case_sensitive
```

```
include file
        'Xinterface.uil';

object
    xinterface_main : XmBulletinBoard {
        callbacks {
            MrmNcreateCallback = procedures {
                xinterface_register_widget('"main"');
            };
            XmNhelpCallback = procedures {
                xinterface_create_widget('"%s", \
                        Resources().userData');
            };
        };
    };


object
    ArrowButton : XmArrowButton {
        callbacks {
            XmNactivateCallback = procedures {
                xinterface_fprintf('stdout, \
                        "activate %s\n", \
                        Registry().name');
            };
            XmNarmCallback = procedures {
                xinterface_fprintf('stdout, \
                        "arm %s\n", \
                        Registry().name');
            };
            MrmNcreateCallback = procedures {
                xinterface_fprintf('stdout, \
                        "create\n"');
            };
            XmNdestroyCallback = procedures {
                xinterface_destroy_widget('"%s", \
                        Registry().name');
            };
            XmNdisarmCallback = procedures {
                xinterface_fprintf('stdout, \
                        "disarm %s\n", \
                        Registry().name');
            };
            XmNhelpCallback = procedures {
                xinterface_fprintf('stdout, \
                        "help %s\n", \
                        Registry().name');
            };
        };
    };


object
```

```
    BulletinBoard : XmBulletinBoard {
        callbacks {
            MrmNcreateCallback = procedures {
                xinterface_fprintf('stdout, \
                        "create\n"');
            };
            XmNdestroyCallback = procedures {
                xinterface_destroy_widget('"%s", \
                        Registry().name');
            };
            XmNfocusCallback = procedures {
                xinterface_fprintf('stdout, \
                        "focus %s\n", \
                        Registry().name');
            };
            XmNhelpCallback = procedures {
                xinterface_create_widget('"%s", \
                        Resources().userData');
            };
            XmNmapCallback = procedures {
                xinterface_fprintf('stdout, \
                        "map\n"');
            };
            XmNunmapCallback = procedures {
                xinterface_fprintf('stdout, \
                        "unmap %s\n", \
                        Registry().name');
            };
        };
    };

    object
        BulletinBoardDialog : XmBulletinBoardDialog {
            callbacks {
                MrmNcreateCallback = procedures {
                    xinterface_fprintf('stdout, \
                            "create\n"');
                };
                XmNdestroyCallback = procedures {
                    xinterface_destroy_widget('"%s", \
                            Registry().name');
                };
                XmNfocusCallback = procedures {
                    xinterface_fprintf('stdout, \
                            "focus %s\n", \
                            Registry().name');
                };
                XmNhelpCallback = procedures {
                    xinterface_create_widget('"%s", \
                            Resources().userData');
                };
                XmNmapCallback = procedures {
                    xinterface_fprintf('stdout, \
```

```
                "map\n"');
        };
        XmNunmapCallback = procedures {
            xinterface_fprintf('stdout, \
                "unmap %s\n", \
                Registry().name');
        };
    };
};


object
    CascadeButton : XmCascadeButton {
        callbacks {
            XmNactivateCallback = procedures {
                xinterface_fprintf('stdout, \
                    "activate %s\n", \
                    Registry().name');
            };
            XmNcascadingCallback = procedures {
                xinterface_fprintf('stdout, \
                    "cascading %s\n", \
                    Registry().name');
            };
            MrmNcreateCallback = procedures {
                xinterface_fprintf('stdout, \
                    "create\n"');
            };
            XmNdestroyCallback = procedures {
                xinterface_destroy_widget('"%s", \
                    Registry().name');
            };
            XmNhelpCallback = procedures {
                xinterface_create_widget('"%s", \
                    Resources().userData');
            };
        };
    };


object
    Command : XmCommand {
        callbacks {
            XmNcommandChangedCallback = procedures {
                xinterface_fprintf('stdout, \
                    "commandChanged %s\n", \
                    Registry().name');
            };
            XmNcommandEnteredCallback = procedures {
                xinterface_fprintf('stdout, \
                    "commandEntered %s\n", \
                    Registry().name');
            };
```

```
            MrmNcreateCallback = procedures {
               xinterface_fprintf('stdout, \
                     "create\n"');
            };
            XmNdestroyCallback = procedures {
               xinterface_destroy_widget('"%s", \
                     Registry().name');
            };
            XmNhelpCallback = procedures {
               xinterface_fprintf('stdout, \
                     "help %s\n", \
                     Registry().name');
            };
         };
      };


   object
      DrawingArea : XmDrawingArea {
         callbacks {
            MrmNcreateCallback = procedures {
               xinterface_fprintf('stdout, \
                     "create\n"');
            };
            XmNdestroyCallback = procedures {
               xinterface_destroy_widget('"%s", \
                     Registry().name');
            };
            XmNexposeCallback = procedures {
               xinterface_fprintf('stdout, \
                     "expose %s\n", \
                     Registry().name');
            };
            XmNhelpCallback = procedures {
               xinterface_create_widget('"%s", \
                     Resources().userData');
            };
            XmNinputCallback = procedures {
               xinterface_fprintf('stdout, \
                     \
                     "%s %d %d %d %d\n", \
                     Resources().userData, \
                     XmAnyCallbackStruct.XEvent.type, \
                     XmAnyCallbackStruct.XEvent.button, \
                     XmAnyCallbackStruct.XEvent.x, \
                     XmAnyCallbackStruct.XEvent.y');
            };
            XmNresizeCallback = procedures {
               xinterface_fprintf('stdout, \
                     "resize %s\n", \
                     Registry().name');
            };
         };
```

```
    };


object
    DrawnButton : XmDrawnButton {
        callbacks {
            XmNactivateCallback = procedures {
                xinterface_fprintf('stdout, \
                        "activate %s\n", \
                        Registry().name');
            };
            XmNarmCallback = procedures {
                xinterface_fprintf('stdout, \
                        "arm %s\n", \
                        Registry().name');
            };
            MrmNcreateCallback = procedures {
                xinterface_fprintf('stdout, \
                        "create\n"');
            };
            XmNdestroyCallback = procedures {
                xinterface_destroy_widget('"%s", \
                        Registry().name');
            };
            XmNdisarmCallback = procedures {
                xinterface_fprintf('stdout, \
                        "disarm %s\n", \
                        Registry().name');
            };
            XmNexposeCallback = procedures {
                xinterface_fprintf('stdout, \
                        "expose %s\n", \
                        Registry().name');
            };
            XmNhelpCallback = procedures {
                xinterface_fprintf('stdout, \
                        "help %s\n", \
                        Registry().name');
            };
            XmNresizeCallback = procedures {
                xinterface_fprintf('stdout, \
                        "resize %s\n", \
                        Registry().name');
            };
        };
    };


object
    ErrorDialog : XmErrorDialog {
        callbacks {
            XmNcancelCallback = procedures {
                xinterface_fprintf('stdout, \
```

```
                "cancel %s\n", \
                Registry().name');
        };
        MrmNcreateCallback = procedures {
            xinterface_fprintf('stdout, \
                "create\n"');
        };
        XmNdestroyCallback = procedures {
            xinterface_destroy_widget('"%s", \
                Registry().name');
        };
        XmNhelpCallback = procedures {
            xinterface_fprintf('stdout, \
                "help %s\n", \
                Registry().name');
        };
        XmNokCallback = procedures {
            xinterface_fprintf('stdout, \
                "ok %s\n", \
                Registry().name');
        };
    };
};


object
    FileSelectionBox : XmFileSelectionBox {
        callbacks {
            XmNapplyCallback = procedures {
                xinterface_fprintf('stdout, \
                    "apply %s\n", \
                    Registry().name');
            };
            XmNcancelCallback = procedures {
                xinterface_fprintf('stdout, \
                    "cancel %s\n", \
                    Registry().name');
            };
            MrmNcreateCallback = procedures {
                xinterface_fprintf('stdout, \
                    "create\n"');
            };
            XmNdestroyCallback = procedures {
                xinterface_destroy_widget('"%s", \
                    Registry().name');
            };
            XmNhelpCallback = procedures {
                xinterface_create_widget('"%s", \
                    Resources().userData');
            };
            XmNmapCallback = procedures {
                xinterface_fprintf('stdout, \
                    "map\n"');
```

```
            };
            XmNnoMatchCallback = procedures {
               xinterface_fprintf('stdout, \
                     "noMatch %s\n", \
                     Registry().name');
            };
            XmNokCallback = procedures {
               xinterface_fprintf('stdout, \
                     "ok %s %s\n", \
                     Registry().name, \
                     XmSelectionBoxCallbackStruct.value');
            };
            XmNunmapCallback = procedures {
               xinterface_fprintf('stdout, \
                     "unmap %s\n", \
                     Registry().name');
            };
         };
      };


   object
      FileSelectionDialog : XmFileSelectionDialog {
         callbacks {
            XmNapplyCallback = procedures {
               xinterface_fprintf('stdout, \
                     "apply %s\n", \
                     Registry().name');
            };
            XmNcancelCallback = procedures {
               xinterface_fprintf('stdout, \
                     "cancel %s\n", \
                     Registry().name');
            };
            MrmNcreateCallback = procedures {
               xinterface_fprintf('stdout, \
                     "create\n"');
            };
            XmNdestroyCallback = procedures {
               xinterface_destroy_widget('"%s", \
                     Registry().name');
            };
            XmNhelpCallback = procedures {
               xinterface_create_widget('"%s", \
                     Resources().userData');
            };
            XmNmapCallback = procedures {
               xinterface_fprintf('stdout, \
                     "map\n"');
            };
            XmNnoMatchCallback = procedures {
               xinterface_fprintf('stdout, \
                     "noMatch %s\n", \
```

```
                Registry().name');
        };
        XmNokCallback = procedures {
          xinterface_fprintf('stdout, \
                "ok %s %s\n", \
                Registry().name, \
                XmSelectionBoxCallbackStruct.value');
        };
        XmNunmapCallback = procedures {
          xinterface_fprintf('stdout, \
                "unmap %s\n", \
                Registry().name');
        };
      };
    };


object
  Form : XmForm {
    callbacks {
      MrmNcreateCallback = procedures {
        xinterface_fprintf('stdout, \
              "create\n"');
      };
      XmNdestroyCallback = procedures {
        xinterface_destroy_widget('"%s", \
              Registry().name');
      };
      XmNfocusCallback = procedures {
        xinterface_fprintf('stdout, \
              "focus %s\n", \
              Registry().name');
      };
      XmNhelpCallback = procedures {
        xinterface_create_widget('"%s", \
              Resources().userData');
      };
      XmNmapCallback = procedures {
        xinterface_fprintf('stdout, \
              "map\n"');
      };
      XmNunmapCallback = procedures {
        xinterface_fprintf('stdout, \
              "unmap %s\n", \
              Registry().name');
      };
    };
  };


object
  FormDialog : XmFormDialog {
```

```
        callbacks {
            MrmNcreateCallback = procedures {
                xinterface_fprintf('stdout, \
                        "create\n"');
            };
            XmNdestroyCallback = procedures {
                xinterface_destroy_widget('"%s", \
                        Registry().name');
            };
            XmNfocusCallback = procedures {
                xinterface_fprintf('stdout, \
                        "focus %s\n", \
                        Registry().name');
            };
            XmNhelpCallback = procedures {
                xinterface_create_widget('"%s", \
                        Resources().userData');
            };
            XmNmapCallback = procedures {
                xinterface_fprintf('stdout, \
                        "map\n"');
            };
            XmNunmapCallback = procedures {
                xinterface_fprintf('stdout, \
                        "unmap %s\n", \
                        Registry().name');
            };
        };
    };


object
    Frame : XmFrame {
        callbacks {
            MrmNcreateCallback = procedures {
                xinterface_fprintf('stdout, \
                        "create\n"');
            };
            XmNdestroyCallback = procedures {
                xinterface_destroy_widget('"%s", \
                        Registry().name');
            };
            XmNhelpCallback = procedures {
                xinterface_create_widget('"%s", \
                        Resources().userData');
            };
        };
    };


object
```

```
    InformationDialog : XmInformationDialog {
        callbacks {
            XmNcancelCallback = procedures {
                xinterface_fprintf('stdout, \
                        "cancel %s\n", \
                        Registry().name');
            };
            MrmNcreateCallback = procedures {
                xinterface_fprintf('stdout, \
                        "create\n"');
            };
            XmNdestroyCallback = procedures {
                xinterface_destroy_widget('"%s", \
                        Registry().name');
            };
            XmNhelpCallback = procedures {
                xinterface_create_widget('"%s", \
                        Resources().userData');
            };
            XmNokCallback = procedures {
                xinterface_fprintf('stdout, \
                        "ok %s\n", \
                        Registry().name');
            };
        };
    };
```

```
object
    Label : XmLabel {
        callbacks {
            MrmNcreateCallback = procedures {
                xinterface_fprintf('stdout, \
                        "create\n"');
            };
            XmNdestroyCallback = procedures {
                xinterface_destroy_widget('"%s", \
                        Registry().name');
            };
            XmNhelpCallback = procedures {
                xinterface_fprintf('stdout, \
                        "help %s\n", \
                        Registry().name');
            };
        };
    };
```

```
object
   List : XmList {
      callbacks {
         XmNbrowseSelectionCallback = procedures {
            xinterface_fprintf('stdout, \
                  "browseSelection %s\n", \
                  Registry().name');
         };
         XmNdefaultActionCallback = procedures {
            xinterface_fprintf('stdout, \
                  "defaultAction %s\n", \
                  Registry().name');
         };
         MrmNcreateCallback = procedures {
            xinterface_fprintf('stdout, \
                  "create\n"');
         };
         XmNdestroyCallback = procedures {
            xinterface_destroy_widget('"%s", \
                  Registry().name');
         };
         XmNextendedSelectionCallback = procedures {
            xinterface_fprintf('stdout, \
                  "extendedSelection %s\n", \
                  Registry().name');
         };
         XmNhelpCallback = procedures {
            xinterface_fprintf('stdout, \
                  "help %s\n", \
                  Registry().name');
         };
         XmNmultipleSelectionCallback = procedures {
            xinterface_fprintf('stdout, \
                  "multipleSelection %s\n", \
                  Registry().name');
         };
         XmNsingleSelectionCallback = procedures {
            xinterface_fprintf('stdout, \
                  "singleSelection %s\n", \
                  Registry().name');
         };
      };
   };


object
   MainWindow : XmMainWindow {
      callbacks {
         MrmNcreateCallback = procedures {
            xinterface_fprintf('stdout, \
                  "create\n"');
         };
         XmNdestroyCallback = procedures {
```

```
            xinterface_destroy_widget('"%s", \
                Registry().name');
        };
        XmNhelpCallback = procedures {
            xinterface_create_widget('"%s", \
                Resources().userData');
        };
    };
};


object
    MenuBar : XmMenuBar {
        callbacks {
            MrmNcreateCallback = procedures {
                xinterface_fprintf('stdout, \
                    "create\n"');
            };
            XmNdestroyCallback = procedures {
                xinterface_destroy_widget('"%s", \
                    Registry().name');
            };
            XmNentryCallback = procedures {
                xinterface_fprintf('stdout, \
                    "entry %s\n", \
                    Registry().name');
            };
            XmNhelpCallback = procedures {
                xinterface_create_widget('"%s", \
                    Resources().userData');
            };
            XmNmapCallback = procedures {
                xinterface_fprintf('stdout, \
                    "map\n"');
            };
            XmNunmapCallback = procedures {
                xinterface_fprintf('stdout, \
                    "unmap %s\n", \
                    Registry().name');
            };
        };
    };


object
    MessageBox : XmMessageBox {
        callbacks {
            XmNcancelCallback = procedures {
                xinterface_fprintf('stdout, \
                    "cancel %s\n", \
                    Registry().name');
            };
```

```
            MrmNcreateCallback = procedures {
               xinterface_fprintf('stdout, \
                     "create\n"');
            };
            XmNdestroyCallback = procedures {
               xinterface_destroy_widget('"%s", \
                     Registry().name');
            };
            XmNhelpCallback = procedures {
               xinterface_fprintf('stdout, \
                     "help %s\n", \
                     Registry().name');
            };
            XmNokCallback = procedures {
               xinterface_fprintf('stdout, \
                     "ok %s\n", \
                     Registry().name');
            };
         };
      };


   object
      MessageDialog : XmMessageDialog {
         callbacks {
            XmNcancelCallback = procedures {
               xinterface_fprintf('stdout, \
                     "cancel %s\n", \
                     Registry().name');
            };
            MrmNcreateCallback = procedures {
               xinterface_fprintf('stdout, \
                     "create\n"');
            };
            XmNdestroyCallback = procedures {
               xinterface_destroy_widget('"%s", \
                     Registry().name');
            };
            XmNhelpCallback = procedures {
               xinterface_fprintf('stdout, \
                     "help %s\n", \
                     Registry().name');
            };
            XmNokCallback = procedures {
               xinterface_fprintf('stdout, \
                     "ok %s\n", \
                     Registry().name');
            };
         };
      };
```

```
object
   OptionMenu : XmOptionMenu {
      callbacks {
         MrmNcreateCallback = procedures {
            xinterface_fprintf('stdout, \
                  "create\n"');
         };
         XmNdestroyCallback = procedures {
            xinterface_destroy_widget('"%s", \
                  Registry().name');
         };
         XmNentryCallback = procedures {
            xinterface_fprintf('stdout, \
                  "entry %s\n", \
                  Registry().name');
         };
         XmNhelpCallback = procedures {
            xinterface_create_widget('"%s", \
                  Resources().userData');
         };
         XmNmapCallback = procedures {
            xinterface_fprintf('stdout, \
                  "map\n"');
         };
         XmNunmapCallback = procedures {
            xinterface_fprintf('stdout, \
                  "unmap %s\n", \
                  Registry().name');
         };
      };
   };


object
   PanedWindow : XmPanedWindow {
      callbacks {
         MrmNcreateCallback = procedures {
            xinterface_fprintf('stdout, \
                  "create\n"');
         };
         XmNdestroyCallback = procedures {
            xinterface_destroy_widget('"%s", \
                  Registry().name');
         };
         XmNhelpCallback = procedures {
            xinterface_create_widget('"%s", \
                  Resources().userData');
         };
      };
   };
```

```
object
   PopupMenu : XmPopupMenu {
      callbacks {
         MrmNcreateCallback = procedures {
            xinterface_fprintf('stdout, \
                  "create\n"');
         };
         XmNdestroyCallback = procedures {
            xinterface_destroy_widget('"%s", \
                  Registry().name');
         };
         XmNentryCallback = procedures {
            xinterface_fprintf('stdout, \
                  "entry %s\n", \
                  Registry().name');
         };
         XmNhelpCallback = procedures {
            xinterface_create_widget('"%s", \
                  Resources().userData');
         };
         XmNmapCallback = procedures {
            xinterface_fprintf('stdout, \
                  "map\n"');
         };
         XmNunmapCallback = procedures {
            xinterface_fprintf('stdout, \
                  "unmap %s\n", \
                  Registry().name');
         };
      };
   };


object
   PromptDialog : XmPromptDialog {
      callbacks {
         XmNcancelCallback = procedures {
            xinterface_fprintf('stdout, \
                  "cancel %s\n", \
                  Registry().name');
         };
         MrmNcreateCallback = procedures {
            xinterface_fprintf('stdout, \
                  "create\n"');
         };
         XmNdestroyCallback = procedures {
            xinterface_destroy_widget('"%s", \
                  Registry().name');
         };
         XmNhelpCallback = procedures {
            xinterface_create_widget('"%s", \
                  Resources().userData');
         };
```

```
            XmNnoMatchCallback = procedures {
               xinterface_fprintf('stdout, \
                     "noMatch %s\n", \
                     Registry().name');
            };
            XmNokCallback = procedures {
               xinterface_fprintf('stdout, \
                     "ok %s %s\n", \
                     Registry().name, \
                     XmSelectionBoxCallbackStruct.value');
            };
         };
      };


   object
      PulldownMenu : XmPulldownMenu {
         callbacks {
            MrmNcreateCallback = procedures {
               xinterface_fprintf('stdout, \
                     "create\n"');
            };
            XmNdestroyCallback = procedures {
               xinterface_destroy_widget('"%s", \
                     Registry().name');
            };
            XmNentryCallback = procedures {
               xinterface_fprintf('stdout, \
                     "entry %s\n", \
                     Registry().name');
            };
            XmNhelpCallback = procedures {
               xinterface_create_widget('"%s", \
                     Resources().userData');
            };
            XmNmapCallback = procedures {
               xinterface_fprintf('stdout, \
                     "map\n"');
            };
            XmNunmapCallback = procedures {
               xinterface_fprintf('stdout, \
                     "unmap %s\n", \
                     Registry().name');
            };
         };
      };


   object
      PushButton : XmPushButton {
         callbacks {
            XmNactivateCallback = procedures {
```

```
            xinterface_fprintf('stdout, \
                    "activate %s\n", \
                    Registry().name');
            };
            XmNarmCallback = procedures {
            xinterface_fprintf('stdout, \
                    "arm %s\n", \
                    Registry().name');
            };
            MrmNcreateCallback = procedures {
            xinterface_fprintf('stdout, \
                    "create\n"');
            };
            XmNdestroyCallback = procedures {
            xinterface_destroy_widget('"%s", \
                    Registry().name');
            };
            XmNdisarmCallback = procedures {
            xinterface_fprintf('stdout, \
                    "disarm %s\n", \
                    Registry().name');
            };
            XmNhelpCallback = procedures {
            xinterface_fprintf('stdout, \
                    "help %s\n", \
                    Registry().name');
            };
        };
    };


object
    QuestionDialog : XmQuestionDialog {
        callbacks {
            XmNcancelCallback = procedures {
            xinterface_fprintf('stdout, \
                    "cancel %s\n", \
                    Registry().name');
            };
            MrmNcreateCallback = procedures {
            xinterface_fprintf('stdout, \
                    "create\n"');
            };
            XmNdestroyCallback = procedures {
            xinterface_destroy_widget('"%s", \
                    Registry().name');
            };
            XmNhelpCallback = procedures {
            xinterface_fprintf('stdout, \
                    "help %s\n", \
                    Registry().name');
            };
            XmNokCallback = procedures {
```

```
            xinterface_fprintf('stdout, \
                "ok %s\n", \
                Registry().name');
        };
    };
};


object
    RadioBox : XmRadioBox {
        callbacks {
            MrmNcreateCallback = procedures {
                xinterface_fprintf('stdout, \
                    "create\n"');
            };
            XmNdestroyCallback = procedures {
                xinterface_destroy_widget('"%s", \
                    Registry().name');
            };
            XmNentryCallback = procedures {
                xinterface_fprintf('stdout, \
                    "entry %s\n", \
                    Registry().name');
            };
            XmNhelpCallback = procedures {
                xinterface_create_widget('"%s", \
                    Resources().userData');
            };
            XmNmapCallback = procedures {
                xinterface_fprintf('stdout, \
                    "map\n"');
            };
            XmNunmapCallback = procedures {
                xinterface_fprintf('stdout, \
                    "unmap %s\n", \
                    Registry().name');
            };
        };
    };


object
    RowColumn : XmRowColumn {
        callbacks {
            MrmNcreateCallback = procedures {
                xinterface_fprintf('stdout, \
                    "create\n"');
            };
            XmNdestroyCallback = procedures {
                xinterface_destroy_widget('"%s", \
                    Registry().name');
```

```
            };
            XmNentryCallback = procedures {
               xinterface_fprintf('stdout, \
                     "entry %s\n", \
                     Registry().name');
            };
            XmNhelpCallback = procedures {
               xinterface_create_widget('"%s", \
                     Resources().userData');
            };
            XmNmapCallback = procedures {
               xinterface_fprintf('stdout, \
                     "map\n"');
            };
            XmNunmapCallback = procedures {
               xinterface_fprintf('stdout, \
                     "unmap %s\n", \
                     Registry().name');
            };
         };
      };


object
   Scale : XmScale {
      callbacks {
         MrmNcreateCallback = procedures {
            xinterface_fprintf('stdout, \
                  "create\n"');
         };
         XmNdestroyCallback = procedures {
            xinterface_destroy_widget('"%s", \
                  Registry().name');
         };
         XmNdragCallback = procedures {
            xinterface_fprintf('stdout, \
                  "drag %s\n", \
                  Registry().name');
         };
         XmNhelpCallback = procedures {
            xinterface_create_widget('"%s", \
                  Resources().userData');
         };
         XmNvalueChangedCallback = procedures {
            xinterface_fprintf('stdout, \
                  "valueChanged %s\n", \
                  Registry().name');
         };
      };
   };
```

```
object
   ScrollBar : XmScrollBar {
      callbacks {
         XmNdecrementCallback = procedures {
            xinterface_fprintf('stdout, \
                  "decrement %s\n", \
                  Registry().name');
         };
         MrmNcreateCallback = procedures {
            xinterface_fprintf('stdout, \
                  "create\n"');
         };
         XmNdestroyCallback = procedures {
            xinterface_destroy_widget('"%s", \
                  Registry().name');
         };
         XmNdragCallback = procedures {
            xinterface_fprintf('stdout, \
                  "drag %s\n", \
                  Registry().name');
         };
         XmNhelpCallback = procedures {
            xinterface_create_widget('"%s", \
                  Resources().userData');
         };
         XmNincrementCallback = procedures {
            xinterface_fprintf('stdout, \
                  "increment %s\n", \
                  Registry().name');
         };
         XmNpageDecrementCallback = procedures {
            xinterface_fprintf('stdout, \
                  "pageDecrement %s\n", \
                  Registry().name');
         };
         XmNpageIncrementCallback = procedures {
            xinterface_fprintf('stdout, \
                  "pageIncrement %s\n", \
                  Registry().name');
         };
         XmNtoBottomCallback = procedures {
            xinterface_fprintf('stdout, \
                  "toBottom %s\n", \
                  Registry().name');
         };
         XmNtoTopCallback = procedures {
            xinterface_fprintf('stdout, \
                  "toTop %s\n", \
                  Registry().name');
         };
         XmNvalueChangedCallback = procedures {
            xinterface_fprintf('stdout, \
                  "valueChanged %s\n", \
```

```
                Registry().name');
        };
    };
};


object
    ScrolledList : XmScrolledList {
        callbacks {
            XmNbrowseSelectionCallback = procedures {
                xinterface_fprintf('stdout, \
                        "browseSelection %s\n", \
                        Registry().name');
            };
            XmNdefaultActionCallback = procedures {
                xinterface_fprintf('stdout, \
                        "defaultAction %s\n", \
                        Registry().name');
            };
            MrmNcreateCallback = procedures {
                xinterface_fprintf('stdout, \
                        "create\n"');
            };
            XmNdestroyCallback = procedures {
                xinterface_destroy_widget('"%s", \
                        Registry().name');
            };
            XmNextendedSelectionCallback = procedures {
                xinterface_fprintf('stdout, \
                        "extendedSelection %s\n", \
                        Registry().name');
            };
            XmNhelpCallback = procedures {
                xinterface_fprintf('stdout, \
                        "help %s\n", \
                        Registry().name');
            };
            XmNmultipleSelectionCallback = procedures {
                xinterface_fprintf('stdout, \
                        "multipleSelection %s\n", \
                        Registry().name');
            };
            XmNsingleSelectionCallback = procedures {
                xinterface_fprintf('stdout, \
                        "singleSelection %s\n", \
                        Registry().name');
            };
        };
    };
```

```
object
   ScrolledText : XmScrolledText {
      callbacks {
         XmNactivateCallback = procedures {
            xinterface_fprintf('stdout, \
                  "activate %s\n", \
                  Registry().name');
         };
         MrmNcreateCallback = procedures {
            xinterface_fprintf('stdout, \
                  "create\n"');
         };
         XmNdestroyCallback = procedures {
            xinterface_destroy_widget('"%s", \
                  Registry().name');
         };
         XmNfocusCallback = procedures {
            xinterface_fprintf('stdout, \
                  "focus %s\n", \
                  Registry().name');
         };
         XmNhelpCallback = procedures {
            xinterface_fprintf('stdout, \
                  "help %s\n", \
                  Registry().name');
         };
         XmNlosingFocusCallback = procedures {
            xinterface_fprintf('stdout, \
                  "losingFocus %s\n", \
                  Registry().name');
         };
         XmNmodifyVerifyCallback = procedures {
            xinterface_fprintf('stdout, \
                  "modifyVerify %s\n", \
                  Registry().name');
         };
         XmNmotionVerifyCallback = procedures {
            xinterface_fprintf('stdout, \
                  "motionVerify %s\n", \
                  Registry().name');
         };
         XmNvalueChangedCallback = procedures {
            xinterface_fprintf('stdout, \
                  "valueChanged %s\n", \
                  Registry().name');
         };
      };
   };


object
   ScrolledWindow : XmScrolledWindow {
```

```
        callbacks {
            MrmNcreateCallback = procedures {
                xinterface_fprintf('stdout, \
                        "create\n"');
            };
            XmNdestroyCallback = procedures {
                xinterface_destroy_widget('"%s", \
                        Registry().name');
            };
            XmNhelpCallback = procedures {
                xinterface_create_widget('"%s", \
                        Resources().userData');
            };
        };
    };


object
    SelectionBox : XmSelectionBox {
        callbacks {
            XmNcancelCallback = procedures {
                xinterface_fprintf('stdout, \
                        "cancel %s\n", \
                        Registry().name');
            };
            MrmNcreateCallback = procedures {
                xinterface_fprintf('stdout, \
                        "create\n"');
            };
            XmNdestroyCallback = procedures {
                xinterface_destroy_widget('"%s", \
                        Registry().name');
            };
            XmNhelpCallback = procedures {
                xinterface_create_widget('"%s", \
                        Resources().userData');
            };
            XmNnoMatchCallback = procedures {
                xinterface_fprintf('stdout, \
                        "noMatch %s\n", \
                        Registry().name');
            };
            XmNokCallback = procedures {
                xinterface_fprintf('stdout, \
                        "ok %s %s\n", \
                        Registry().name, \
                        XmSelectionBoxCallbackStruct.value');
            };
        };
    };
```

```
object
   SelectionDialog : XmSelectionDialog {
      callbacks {
         XmNcancelCallback = procedures {
            xinterface_fprintf('stdout, \
                  "cancel %s\n", \
                  Registry().name');
         };
         MrmNcreateCallback = procedures {
            xinterface_fprintf('stdout, \
                  "create\n"');
         };
         XmNdestroyCallback = procedures {
            xinterface_destroy_widget('"%s", \
                  Registry().name');
         };
         XmNhelpCallback = procedures {
            xinterface_create_widget('"%s", \
                  Resources().userData');
         };
         XmNnoMatchCallback = procedures {
            xinterface_fprintf('stdout, \
                  "noMatch %s\n", \
                  Registry().name');
         };
         XmNokCallback = procedures {
            xinterface_fprintf('stdout, \
                  "ok %s %s\n", \
                  Registry().name, \
                  XmSelectionBoxCallbackStruct.value');
         };
      };
   };


object
   Separator : XmSeparator {
      callbacks {
         MrmNcreateCallback = procedures {
            xinterface_fprintf('stdout, \
                  "create\n"');
         };
         XmNdestroyCallback = procedures {
            xinterface_destroy_widget('"%s", \
                  Registry().name');
         };
         XmNhelpCallback = procedures {
            xinterface_create_widget('"%s", \
                  Resources().userData');
         };
      };
   };
```

```
object
   Text : XmText {
      callbacks {
         XmNactivateCallback = procedures {
            xinterface_fprintf('stdout, \
                  "activate %s\n", \
                  Registry().name');
         };
         MrmNcreateCallback = procedures {
            xinterface_fprintf('stdout, \
                  "create\n"');
         };
         XmNdestroyCallback = procedures {
            xinterface_destroy_widget('"%s", \
                  Registry().name');
         };
         XmNfocusCallback = procedures {
            xinterface_fprintf('stdout, \
                  "focus %s\n", \
                  Registry().name');
         };
         XmNhelpCallback = procedures {
            xinterface_fprintf('stdout, \
                  "help %s\n", \
                  Registry().name');
         };
         XmNlosingFocusCallback = procedures {
            xinterface_fprintf('stdout, \
                  "losingFocus %s\n", \
                  Registry().name');
         };
         XmNmodifyVerifyCallback = procedures {
            xinterface_fprintf('stdout, \
                  "modifyVerify %s\n", \
                  Registry().name');
         };
         XmNmotionVerifyCallback = procedures {
            xinterface_fprintf('stdout, \
                  "motionVerify %s\n", \
                  Registry().name');
         };
         XmNvalueChangedCallback = procedures {
            xinterface_fprintf('stdout, \
                  "valueChanged %s\n", \
                  Registry().name');
         };
      };
   };


object
   ToggleButton : XmToggleButton {
```

```
        callbacks {
            XmNarmCallback = procedures {
                xinterface_fprintf('stdout, \
                        "arm %s\n", \
                        Registry().name');
            };
            MrmNcreateCallback = procedures {
                xinterface_fprintf('stdout, \
                        "create\n"");
            };
            XmNdestroyCallback = procedures {
                xinterface_destroy_widget('"%s", \
                        Registry().name');
            };
            XmNdisarmCallback = procedures {
                xinterface_fprintf('stdout, \
                        "disarm %s\n", \
                        Registry().name');
            };
            XmNhelpCallback = procedures {
                xinterface_create_widget('"%s", \
                        Resources().userData');
            };
            XmNvalueChangedCallback = procedures {
                xinterface_fprintf('stdout, \
                        "valueChanged %s\n", \
                        Registry().name');
            };
        };
    };


object
    WarningDialog : XmWarningDialog {
        callbacks {
            XmNcancelCallback = procedures {
                xinterface_fprintf('stdout, \
                        "cancel %s\n", \
                        Registry().name');
            };
            MrmNcreateCallback = procedures {
                xinterface_fprintf('stdout, \
                        "create\n"");
            };
            XmNdestroyCallback = procedures {
                xinterface_destroy_widget('"%s", \
                        Registry().name');
            };
            XmNhelpCallback = procedures {
                xinterface_fprintf('stdout, \
                        "help %s\n", \
                        Registry().name');
```

```
            };
            XmNokCallback = procedures {
               xinterface_fprintf('stdout, \
                     "ok %s\n", \
                     Registry().name');
            };
         };
      };


   object
      WorkingDialog : XmWorkingDialog {
         callbacks {
            XmNcancelCallback = procedures {
               xinterface_fprintf('stdout, \
                     "cancel %s\n", \
                     Registry().name');
            };
            MrmNcreateCallback = procedures {
               xinterface_fprintf('stdout, \
                     "create\n"');
            };
            XmNdestroyCallback = procedures {
               xinterface_destroy_widget('"%s", \
                     Registry().name');
            };
            XmNhelpCallback = procedures {
               xinterface_fprintf('stdout, \
                     "help %s\n", \
                     Registry().name');
            };
            XmNokCallback = procedures {
               xinterface_fprintf('stdout, \
                     "ok %s\n", \
                     Registry().name');
            };
         };
      };


   end module;
```

FILES

/home/resstaff/bin/X11/Xcanvas /home/resstaff/lib/X11/app-defaults/XInterface

SEE ALSO

X(1), Xinterface(1), Xps(1)

XPS

NAME

Xps - Xinterface postscript display utility.

SYNOPSIS

Xps [Xinterface arguments] [-size <page_size>] [-help]

DESCRIPTION

Xps is an *Xinterface* utility which displays postscript. It reads postscript commands from standard input and displays them on a canvas.

OPTIONS

All arguments are forwarded to *Xinterface* except the following:

–help display the usage of the command.

–size next argument denotes the size of the canvas and must correspond to a postscript definition in either the default library or the *.postscript* file in the users home directory.

X DEFAULTS

Resource values for this program will be read for client name *Xps* and client class *XInterface*. The default values are defined in the file */home/resstafflib/X11/app-defaults/XInterface* but may be redefined in *$HOME/.Xdefaults*.

The current set of defaults is as follows:

```
Xps*.xinterface_main.background: black
Xps*.xinterface_main.foreground: white
Xps*.cancel_button.labelString: CANCEL
Xps*.cancel_button.x: 10
Xps*.cancel_button.y: 10
Xps*.draw_area.background: white
Xps*.draw_area.x: 100
Xps*.draw_area.y: 10
Xps*.draw_area.height: 900
Xps*.draw_area.width: 700
Xps*.output_button.mappedWhenManaged: false
```

UIL FILE

```
! ============================================================
! Interactive Postscript Display UIL file

module xinterface
      names = case_sensitive

include file
            'Xinterface.uil';

object
   xinterface_main : XmBulletinBoard {
         controls {
            XmPushButton    cancel_button;
```

```
            XmPushButton    output_button;
            XmDrawingArea   draw_area;
        };
    };


object
    cancel_button : XmPushButton {
        callbacks {
            MrmNcreateCallback = procedures {
                xinterface_register_widget('"cancel_button"');
            };
            XmNactivateCallback = procedures {
                xinterface_fprintf('stdout, "CANCEL\n"');
                xinterface_exit('"0"');
            };
        };
    };


object
    output_button : XmPushButton {
        callbacks {
            MrmNcreateCallback = procedures {
                xinterface_register_widget('"output_button"');
            };
            XmNactivateCallback = procedures {
                xinterface_fprintf('stdout, "OUTPUT\n"');
            };
        };
    };


object
    draw_area : XmDrawingArea {
        arguments {
            XmNtranslations = translation_table(
                                '<BtnMotion>: Help()');
        };
        callbacks {
            MrmNcreateCallback = procedures {
                xinterface_register_widget('"draw_area"');
            };
            XmNinputCallback = procedures {
                xinterface_fprintf('stdout, \
                        "draw_area %d %d %d %d\n", \
                                        XmAnyCallbackStruct.XEvent.type, \
                                        XmAnyCallbackStruct.XEvent.button, \
                                        XmAnyCallbackStruct.XEvent.x, \
                                        XmAnyCallbackStruct.XEvent.y');
            };
            XmNhelpCallback = procedures {
                xinterface_fprintf('stdout, \
```

```
                        "draw_area %d %d %d\n", \
                                XmAnyCallbackStruct.XEvent.type, \
                                XmAnyCallbackStruct.XEvent.x, \
                                XmAnyCallbackStruct.XEvent.y');
            };
            XmNexposeCallback = procedures {
               xinterface_fprintf('stdout, "EXPOSE\n"');
            };
         };
      };

   end module;
```

FILES

/home/resstaff/lib/postscript/psrc

SEE ALSO

X(1), Xinterface(1), xps(1)

XGRAPHED

NAME

Xgraphed - Xinterface graph layout and interactive editor utility.

SYNOPSIS

Xgraphed [Xinterface arguments] [–options ...]

DESCRIPTION

Xgraphed is an *Xinterface* utility which given a description of a graph in terms of nodes and edges attempts to find a reasonably good two dimensional presentation of the graph. The result is displayed on a letter size canvas. This can then be interactively edited. When the desired graph is achieved it can be output in postscript format.

OPTIONS

All arguments are forwarded to *Xinterface* except the following:

–help display the usage of the command.

–bias *<direction>*
the layout procedure can attempt to lay out the graph in either a *vertical* or a *horizontal* direction.

–fast do not backtrack.

–directed
treat the graph description as directed.

–in the next argument is the name of the graph description file.

–out the next argument is the name of the output file (default stdout).

–scale *<scale_factor>*
the graph may be scaled by *scale_factor*.

X DEFAULTS

Resource values for this program are read for the client name *Xgraphed* and client class *XInterface*. The default values are defined in the file */home/resstaff/lib/X11/app-defaults/XInterface* but may be redefined in *$HOME/.Xdefaults*.

The current set of defaults is as follows:

```
Xgraphed*.xinterface_main.background: black
Xgraphed*.xinterface_main.foreground: white
Xgraphed*.cancel_button.labelString: CANCEL
Xgraphed*.cancel_button.x: 10
Xgraphed*.cancel_button.y: 10
Xgraphed*.output_button.labelString: OUTPUT
Xgraphed*.output_button.x: 10
Xgraphed*.output_button.y: 100
Xgraphed*.draw_area.background: white
Xgraphed*.draw_area.x: 100
Xgraphed*.draw_area.y: 10
Xgraphed*.draw_area.height: 900
Xgraphed*.draw_area.width: 700
```

EXAMPLES

The following is an input file for Xgraphed

```
fred
        jim friend
        steve friend
        eric brother
        bill enemy
        simon enemy
steve
        bill enemy
        simon friend
        eric friend
roger
        simon friend
        jim brother
maurice
        fred friend
        lois spouce
```

UIL FILE

Xgraphed uses the *Xps* UIL specification.

FILES

$HOME/.postscript /home/resstaff/lib/X11/postscript/psrc

BUG REPORTS

John Lewis, SRDG, *<lewis@cpsc.UCalgary.CA>*

SEE ALSO

X(1), Xinterface(1), Xps(1)

XGMACS

NAME

Xgmacs - GNU Emacs with a Motif style interface.

SYNOPSIS

Xgmacs [standard GNU Emacs arguments]

DESCRIPTION

Xgmacs starts up GNU Emacs and via its support for process pipes provides a *Motif* interface. Xgmacs is simply the following shell script:

```
#! /bin/csh
gmacs $argv:q -l /home/resstaff/lib/X11/Xinterface/.xgmacs
```

The file /home/resstaff/lib/X11/Xinterface/.xgmacs is the following GNU Emacs lisp file:

```
(defun xinterface-filter (p s)
  (eval (read s)))

(let ((p (start-process "Xinterface"
                        nil
                        "Xinterface"
                        "-name"
                        "Xgmacs"
                        "-uid"
                        "Xgmacs.uid")))
  (set-process-filter p 'xinterface-filter))
```

OPTIONS

Xgmacs simply passes all arguments to GNU Emacs.

X DEFAULTS

Resource values for this program are read for the client name *Xgmacs* and client class *XInterface*. The default values are defined in the file */home/resstaff/lib/X11/app-defaults/XInterface* but may be redefined in *$HOME/.Xdefaults*.

The current set of defaults is as follows:

```
Xgmacs*background: black
Xgmacs.*.help.x: 0
Xgmacs.*.help.labelString: HELP
Xgmacs.*.help_b1.labelString: help-with-tutorial
Xgmacs.*.help_b1.userData: (help-with-tutorial)
Xgmacs.*.help_b2.labelString: apropos
```

```
Xgmacs.*.help_b2.userData: (call-interactively 'apropos)
Xgmacs.*.help_b3.labelString: describe-function
Xgmacs.*.help_b3.userData: (call-interactively 'describe-function)
Xgmacs.*.help_b4.labelString: describe-bindings
Xgmacs.*.help_b4.userData: (describe-bindings)
Xgmacs.*.help_b5.labelString: describe-key
Xgmacs.*.help_b5.userData: (call-interactively 'describe-key)
Xgmacs.*.help_b6.mappedWhenManaged: false
Xgmacs.*.help_b7.mappedWhenManaged: false
Xgmacs.*.help_b8.mappedWhenManaged: false
Xgmacs.*.help_b9.mappedWhenManaged: false
Xgmacs.*.help_b10.mappedWhenManaged: false
Xgmacs.*.mail.y: 50
Xgmacs.*.mail.labelString: MAIL
Xgmacs.*.mail_b1.labelString: rmail
Xgmacs.*.mail_b1.userData: (rmail)(rmail-summary)
Xgmacs.*.mail_b2.labelString: mail
Xgmacs.*.mail_b2.userData: (mail)
Xgmacs.*.mail_b3.mappedWhenManaged: false
Xgmacs.*.mail_b4.mappedWhenManaged: false
Xgmacs.*.mail_b5.mappedWhenManaged: false
Xgmacs.*.mail_b6.mappedWhenManaged: false
Xgmacs.*.mail_b7.mappedWhenManaged: false
Xgmacs.*.mail_b8.mappedWhenManaged: false
Xgmacs.*.mail_b9.mappedWhenManaged: false
Xgmacs.*.mail_b10.mappedWhenManaged: false
Xgmacs.*.cursor_panel.y: 150
Xgmacs.*.gmacs_cursor_radio_box.x: 20
Xgmacs.*.gmacs_cursor_radio_box.y: 10
Xgmacs.*.gmacs_cursor_radio_box.spacing: 2
Xgmacs.*.gmacs_cursor_radio_box.marginHeight: 0
Xgmacs.*.gmacs_cursor_radio_box.marginWidth: 0
Xgmacs.*.gmacs_cursor_radio_box.isHomogeneous: false
Xgmacs.*.radio_button_1.marginHeight: 0
Xgmacs.*.radio_button_1.marginWidth: 0
Xgmacs.*.radio_button_1.labelString: Character
Xgmacs.*.radio_button_1.set: true
Xgmacs.*.radio_button_2.marginHeight: 0
Xgmacs.*.radio_button_2.marginWidth: 0
Xgmacs.*.radio_button_2.labelString: Page
Xgmacs.*.radio_button_3.marginHeight: 0
Xgmacs.*.radio_button_3.marginWidth: 0
Xgmacs.*.radio_button_3.labelString: Window
Xgmacs.*.cursor_arrows.x: 20
Xgmacs.*.cursor_arrows.y: 70
Xgmacs.*.cursor_up.x: 40
Xgmacs.*.cursor_up.y: 0
Xgmacs.*.cursor_up.arrowDirection: ARROW_UP
Xgmacs.*.cursor_down.x: 40
Xgmacs.*.cursor_down.y: 50
Xgmacs.*.cursor_down.arrowDirection: ARROW_DOWN
Xgmacs.*.cursor_left.x: 20
Xgmacs.*.cursor_left.y: 30
```

```
Xgmacs.*.cursor_left.arrowDirection: ARROW_LEFT
Xgmacs.*.cursor_right.x: 60
Xgmacs.*.cursor_right.y: 30
Xgmacs.*.cursor_right.arrowDirection: ARROW_RIGHT
Xgmacs.*.find_file_button.y: 100
Xgmacs.*.find_file_button.labelString: find-file ...
Xgmacs.*.gmacs_readfile_dialog*XmText.editable: true
Xgmacs.*.undo.x: 40
Xgmacs.*.undo.y: 350
Xgmacs.*.undo.width: 100
Xgmacs.*.undo.height: 50
Xgmacs.*.undo.labelString: UNDO
Xgmacs.*.quit.x: 40
Xgmacs.*.quit.y: 450
Xgmacs.*.quit.width: 100
Xgmacs.*.quit.labelString: EXIT
Xgmacs*.sashHeight: 1
Xgmacs*.sashWidth: 1
```

UIL FILE

This is the UIL module for the GNU Emacs Panel. You are reading the header documentation for the UIL code. This is a quite complex example of how to construct a UIL description for an interface for Xinterface. The GNU Emacs Panel is an interface for the GNU Emacs editor which supplies:

help
– a cascade button containing a pulldown menu with selections for:
help-with-tutorial
apropos
describe-function
describe-bindings
describe-key

mail
– a framed options menu with selections for rmail or smail.

cursor
– a cascade button which pops up a bulletin board on which are posted arrow buttons which control cursor movement. A radio box is used to switch between character and page controls. When the popup is active the cursor button is insensitive.

window
– a cascade button which pops up a bulletin board on which is posted arrow buttons and push buttons which control window operations such as : grow, shrink, split, delete, move cursor to next, and move cursor to previous. When the popup is active the cursor button is insensitive.

find-file
– a cascade button which pops up a high level motif dialog box. This dialog box shows a directory listing and allows the user to select a file. The callback I have associated with the OK button causes GNU Emacs to read the file into a new window. When the popup is active the cursor button is insensitive.

undo
– a push button which requests a GNU Emacs undo.

exit-emacs
– a push button which exits the interface and sends GNU Emacs an exit-emacs command. This interface is not only useful; but also demonstrates the use of many of the motif widgets as well as the syntax of UIL.

```
!============================================================================
! GNU Emacs Control Panel UIL file

module xinterface
      names = case_sensitive

include file
            'Xinterface.uil';

object
   xinterface_main : XmBulletinBoard {
      controls {
            XmOptionMenu    help;
            XmOptionMenu    mail;
            XmBulletinBoard cursor_panel;
            XmPushButton    find_file_button;
            XmPushButton    undo;
            XmPushButton    quit;
         };
   };


object
   userData_button : XmPushButton {
         callbacks {
            XmNactivateCallback = procedures {
               xinterface_fprintf('stdout, "%s\n", \
                     Resources().userData');
            };
         };
   };


object
   help : XmOptionMenu {
         controls {
            XmPulldownMenu {
               callbacks {
                     MrmNcreateCallback = procedures {
                        xinterface_create_widget('"userData_button \
                                                      help_b1"');
                        xinterface_create_widget('"userData_button \
                                                      help_b2"');
                        xinterface_create_widget('"userData_button \
                                                      help_b3"');
                        xinterface_create_widget('"userData_button \
                                                      help_b4"');
                        xinterface_create_widget('"userData_button \
                                                      help_b5"');
                        xinterface_create_widget('"userData_button \
                                                      help_b6"');
                        xinterface_create_widget('"userData_button \
```

```
                                                    help_b7"');
                        xinterface_create_widget('"userData_button \
                                                    help_b8"');
                        xinterface_create_widget('"userData_button \
                                                    help_b9"');
                        xinterface_create_widget('"userData_button \
                                                    help_b10"');
                    };
                };
            };
        };
    };


object
    mail : XmOptionMenu {
        controls {
            XmPulldownMenu {
                callbacks {
                    MrmNcreateCallback = procedures {
                        xinterface_create_widget('"userData_button \
                                                    mail_b1"');
                        xinterface_create_widget('"userData_button \
                                                    mail_b2"');
                        xinterface_create_widget('"userData_button \
                                                    mail_b3"');
                        xinterface_create_widget('"userData_button \
                                                    mail_b4"');
                        xinterface_create_widget('"userData_button \
                                                    mail_b5"');
                        xinterface_create_widget('"userData_button \
                                                    mail_b6"');
                        xinterface_create_widget('"userData_button \
                                                    mail_b7"');
                        xinterface_create_widget('"userData_button \
                                                    mail_b8"');
                        xinterface_create_widget('"userData_button \
                                                    mail_b9"');
                        xinterface_create_widget('"userData_button \
                                                    mail_b10"');
                    };
                };
            };
        };
    };

object
    cursor_panel : XmBulletinBoard {
        controls {
            XmBulletinBoard cursor_arrows;
            XmRadioBox              cursor_radio_box;
        };
    };
```

```
object
   cursor_radio_box : XmRadioBox {
      arguments {
            XmNisHomogeneous = false;
         };
         controls {
            XmToggleButton radio_button_1;
          XmToggleButton radio_button_2;
          XmToggleButton radio_button_3;
      };
   };

object
   radio_button_1 : XmToggleButton {
         callbacks {
            XmNarmCallback = procedures {
                  xinterface_iprintf(\
             "SETARG cursor_up userData (previous-line 1)"');
                  xinterface_iprintf(\
             "SETARG cursor_down userData (next-line 1)"');
                  xinterface_iprintf(\
             "SETARG cursor_left userData (backward-char)"');
                  xinterface_iprintf(\
             "SETARG cursor_right userData (forward-char)"');
            };
            MrmNcreateCallback = procedures {
                  xinterface_register_widget(""radio_button_1"');
                  xinterface_iprintf(\
             "CALLBACK radio_button_1 armCallback"');
            };
         };
   };

object
   radio_button_2 : XmToggleButton {
         callbacks {
            XmNarmCallback = procedures {
                  xinterface_iprintf(\
             "SETARG cursor_up userData (scroll-down)"');
                  xinterface_iprintf(\
             "SETARG cursor_down userData (scroll-up)"');
                  xinterface_iprintf(\
             "SETARG cursor_left userData (beginning-of-line)"');
                  xinterface_iprintf(\
             "SETARG cursor_right userData (end-of-line)"');
            };
         };
   };

object
   radio_button_3 : XmToggleButton {
         callbacks {
            XmNarmCallback = procedures {
```

```
                xinterface_iprintf('\
            "SETARG cursor_up userData (previous-window)"');
                xinterface_iprintf('\
            "SETARG cursor_down userData (other-window 1)"');
                xinterface_iprintf('\
            "SETARG cursor_left userData (shrink-window 1)"');
                xinterface_iprintf('\
            "SETARG cursor_right userData (enlarge-window 1)"');
            };
        };
    };


object
    cursor_arrows : XmBulletinBoard {
        callbacks {
            MrmNcreateCallback = procedures {
                xinterface_create_widget('"cursor_arrow cursor_up"');
                xinterface_create_widget('"cursor_arrow cursor_down"');
                xinterface_create_widget('"cursor_arrow cursor_left"');
                xinterface_create_widget('"cursor_arrow cursor_right"');
            };
        };
    };

object
    cursor_arrow : XmArrowButton {
        callbacks {
            XmNactivateCallback = procedures {
                xinterface_fprintf('stdout, "%s\n", \
                                Resources().userData');
            };
        };
    };


object
    find_file_button : XmPushButton {
        callbacks {
            MrmNcreateCallback = procedures {
                xinterface_register_widget('"find_file_button"');
            };
            XmNactivateCallback = procedures {
                xinterface_create_widget('"gmacs_readfile_dialog \
                        gmacs_readfile_dialog \"');
                xinterface_iprintf('"SETARG find_file_button sensitive false"');
            };
        };
    };
```

```
object
   gmacs_readfile_dialog : XmFileSelectionDialog {
      callbacks {
         XmNcancelCallback = procedures {
            xinterface_destroy_widget('"gmacs_readfile_dialog"');
                  xinterface_iprintf('"SETARG find_file_button sensitive true"');
         };
         XmNokCallback = procedures {
               xinterface_fprintf('stdout, \
                     "(find-file-other-window \\\"%s\\\")\n", \
                                    XmSelectionBoxCallbackStruct.value');
         };
      };
   };


object
   undo : XmPushButton {
         callbacks {
            XmNactivateCallback = procedures {
                  xinterface_fprintf('stdout, "(let* () \
                                          (if (equal last-command \'undo) \
                                             (undo-more 1) \
                                             (undo)) \
                                          (setq last-command \'undo))\n"');
            };
         };
   };



object
   quit : XmPushButton {
         callbacks {
            MrmNcreateCallback = procedures {
               xinterface_register_widget('"quit"');
            };
            XmNactivateCallback = procedures {
               xinterface_fprintf('stdout, "(kill-emacs 0)\n"');
               xinterface_exit('"0"');
            };
         };
   };



end module;
```

FILES

/home/resstaff/bin/X11/Xgmacs
/home/resstaff/lib/X11/app-defaults/XInterface

SEE ALSO

emacs(1), gmacs(1), Xinterface(1), X(1), uil(7)

XDOC

Invoking *Xinterface* without a –uid argument activates an interactive documentation reader.

X DEFAULTS

Resource values for this program will be read for client name *Xinterface* and client class *XInterface*. The default values are defined in the file */home/resstaff/lib/X11/app-defaults/XInterface* but may be redefined in *$HOME/.Xdefaults*.

The current set of defaults is as follows:

```
Xinterface*.background: black
Xinterface*.foreground: white
Xinterface*.xinterface_main.XmBulletinBoard.mappedWhenManaged: false
Xinterface*.xinterface_main.doc_1.mappedWhenManaged: false
Xinterface*.XmBulletinBoard.background: black
Xinterface*.XmBulletinBoard.foreground: white
Xinterface*.XmBulletinBoard.x: 0
Xinterface*.XmBulletinBoard.y: 0
Xinterface*.doc_text.fontList: fixed
Xinterface*.doc_text.rows: 60
Xinterface*.doc_text.columns: 80
Xinterface*.doc_text.background: white
Xinterface*.doc_text.foreground: black
Xinterface*.doc_text.editable: false
Xinterface*.doc_text.editMode: MULTI_LINE_EDIT
Xinterface*.doc_text.height: 600
Xinterface*.cancel_button.x: 460
Xinterface*.cancel_button.y: 660
Xinterface*.cancel_button.labelString: CANCEL
Xinterface*.system_button.x: 100
Xinterface*.system_button.y: 660
Xinterface*.system_button.labelString: DEMO
Xinterface*.backward_button.sensitive: false
Xinterface*.backward_button.x: 300
Xinterface*.backward_button.y: 660
Xinterface*.backward_button.labelString: PREVIOUS
Xinterface*.doc_1_options.mappedWhenManaged: false
Xinterface*.doc_1_options.x: 20
Xinterface*.doc_1_options.y: 660
Xinterface*.doc_1_options.labelString: OPTIONS
Xinterface*.doc_1_1b.labelString: OVERVIEW
Xinterface*.doc_1_1b.userData: doc_1_1
Xinterface*.doc_1_2b.labelString: EXISTING INTERFACES
Xinterface*.doc_1_2b.userData: doc_1_2
Xinterface*.doc_1_3b.labelString: BUILDING INTERFACES
Xinterface*.doc_1_3b.userData: doc_1_3
Xinterface*.doc_1_2_options.mappedWhenManaged: false
```

```
Xinterface*.doc_1_2_options.x: 20
Xinterface*.doc_1_2_options.y: 660
Xinterface*.doc_1_2_options.labelString: OPTIONS
Xinterface*.doc_1_2_Xdialog.labelString: Xdialog
Xinterface*.Xdialog_demo.labelString: Xdialog DEMO
Xinterface*.doc_1_2_Xmemo.labelString: Xmemo
Xinterface*.Xmemo_demo.labelString: Xmemo DEMO
Xinterface*.doc_1_2_Xbiff.labelString: Xbiff
Xinterface*.doc_1_2_Xchez.labelString: Xchez
Xinterface*.doc_1_2_Xcalc.labelString: Xcalc
Xinterface*.Xcalc_demo.labelString: Xcalc DEMO
Xinterface*.doc_1_2_Xcanvas.labelString: Xcanvas
Xinterface*.Xcanvas_demo.labelString: Xcanvas DEMO
Xinterface*.doc_1_2_Xps.labelString: Xps
Xinterface*.Xps_demo.labelString: Xps DEMO
Xinterface*.doc_1_2_Xgraphed.labelString: Xgraphed
Xinterface*.Xgraphed_demo.labelString: Xgraphed DEMO
Xinterface*.doc_1_2_Xgmacs.labelString: Xgmacs
Xinterface*.Xgmacs_demo.labelString: Xgmacs DEMO
Xinterface*.doc_1_2_Xdoc.labelString: Xdoc
Xinterface*.doc_1_3_options.mappedWhenManaged: false
Xinterface*.doc_1_3_options.x: 20
Xinterface*.doc_1_3_options.y: 660
Xinterface*.doc_1_3_options.labelString: OPTIONS
Xinterface*.doc_1_3_b1.labelString: UIL AND XINTERFACE
Xinterface*.doc_1_3_b1.userData: doc_1_3_1
Xinterface*.doc_1_3_b2.labelString: XINTERFACE CALLBACKS
Xinterface*.doc_1_3_b2.userData: doc_1_3_2
Xinterface*.doc_1_3_b3.labelString: COMMUNICATING WITH XINTERFACE
Xinterface*.doc_1_3_b3.userData: doc_1_3_3
Xinterface*.doc_1_3_2_options.mappedWhenManaged: false
Xinterface*.doc_1_3_2_options.x: 20
Xinterface*.doc_1_3_2_options.y: 660
Xinterface*.doc_1_3_2_options.labelString: OPTIONS
Xinterface*.doc_1_3_2_1.labelString: xinterface_create_widget
Xinterface*.doc_1_3_2_2.labelString: xinterface_destroy_widget
Xinterface*.doc_1_3_2_3.labelString: xinterface_manage_widget
Xinterface*.doc_1_3_2_4.labelString: xinterface_unmanage_widget
Xinterface*.doc_1_3_2_5.labelString: xinterface_register_widget
Xinterface*.doc_1_3_2_6.labelString: xinterface_deregister_widget
Xinterface*.doc_1_3_2_7.labelString: xinterface_iprintf
Xinterface*.doc_1_3_2_8.labelString: xinterface_fprintf
Xinterface*.doc_1_3_2_9.labelString: xinterface_args
Xinterface*.doc_1_3_2_10.labelString: xinterface_system
Xinterface*.doc_1_3_2_11.labelString: xinterface_TextGetString
Xinterface*.doc_1_3_2_12.labelString: xinterface_TextSetString
Xinterface*.doc_1_3_2_13.labelString: xinterface_exit
Xinterface*.doc_1_3_3_options.mappedWhenManaged: false
Xinterface*.doc_1_3_3_options.x: 20
Xinterface*.doc_1_3_3_options.y: 660
Xinterface*.doc_1_3_3_options.labelString: OPTIONS
Xinterface*.doc_1_3_3_1.labelString: Communicating TO Xinterface
Xinterface*.doc_1_3_3_2.labelString: Communicating FROM Xinterface
```

```
Xinterface*.doc_1_3_3_3.labelString: Communicating IN Xinterface
Xinterface*.doc_1_3_3_4.labelString: Command Line Arguments
Xinterface*.doc_1.userData: 0
Xinterface*.doc_1_1.userData: 10500
Xinterface*.doc_1_2.userData: 25500
Xinterface*.doc_1_2_Xdialog.userData: Xdialog
Xinterface*.doc_1_2_Xmemo.userData: Xmemo
Xinterface*.doc_1_2_Xbiff.userData: Xbiff
Xinterface*.doc_1_2_Xchez.userData: Xchez
Xinterface*.doc_1_2_Xcalc.userData: Xcalc
Xinterface*.doc_1_2_Xcanvas.userData: Xcanvas
Xinterface*.doc_1_2_Xps.userData: Xps
Xinterface*.doc_1_2_Xgraphed.userData: Xgraphed
Xinterface*.doc_1_2_Xgmacs.userData: Xgmacs
Xinterface*.doc_1_2_Xdoc.userData: Xdoc
Xinterface*.Xdialog.userData: 25500
Xinterface*.Xmemo.userData: 40500
Xinterface*.Xbiff.userData: 60500
Xinterface*.Xchez.userData: 70500
Xinterface*.Xcalc.userData: 90500
Xinterface*.Xcanvas.userData: 110500
Xinterface*.Xps.userData: 180500
Xinterface*.Xgraphed.userData: 200500
Xinterface*.Xgmacs.userData: 220500
Xinterface*.Xdoc.userData: 270500
Xinterface*.doc_1_3.userData: 320500
Xinterface*.doc_1_3_1.userData: 320500
Xinterface*.doc_1_3_2.userData: 340500
Xinterface*.doc_1_3_2_1.userData: 340500
Xinterface*.doc_1_3_2_2.userData: 341360
Xinterface*.doc_1_3_2_3.userData: 341700
Xinterface*.doc_1_3_2_4.userData: 341960
Xinterface*.doc_1_3_2_5.userData: 342160
Xinterface*.doc_1_3_2_6.userData: 343280
Xinterface*.doc_1_3_2_7.userData: 343510
Xinterface*.doc_1_3_2_8.userData: 345110
Xinterface*.doc_1_3_2_9.userData: 346060
Xinterface*.doc_1_3_2_10.userData: 346760
Xinterface*.doc_1_3_2_11.userData: 347060
Xinterface*.doc_1_3_2_12.userData: 347310
Xinterface*.doc_1_3_2_13.userData: 347810
Xinterface*.doc_1_3_3.userData: 360500
Xinterface*.doc_1_3_3_1.userData: 360500
Xinterface*.doc_1_3_3_2.userData: 370500
Xinterface*.doc_1_3_3_3.userData: 380500
```

UIL FILE

```
!=============================================================
! Interactive Documentation UIL file

module xinterface
   names = case_sensitive
```

```
include file
        'Xinterface.uil';


object
   xinterface_main : XmBulletinBoard {
        callbacks {
          MrmNcreateCallback = procedures {
            xinterface_create_widget('"doc_text doc_text"');
            xinterface_create_widget('"cancel_button cancel_button"');
            xinterface_create_widget('"backward_button backward_button"');
            xinterface_create_widget('"system_button system_button"');

            xinterface_create_widget('"doc_1_options doc_1_options"');
            xinterface_create_widget('"bulletin_board doc_1"');
                xinterface_iprintf('"SETVAR doc_1 option_menu doc_1_options"');
                xinterface_iprintf('"SETVAR doc_1 option_menu_mapped true"');
                xinterface_iprintf('"SETVAR doc_1 previous doc_1"');
                xinterface_iprintf('"SETVAR doc_1 backward_sensitive false"');
                xinterface_iprintf('"SETVAR doc_1 system_mapped false"');
                xinterface_iprintf('"SETVAR doc_1 system_userData nil"');

            xinterface_create_widget('"bulletin_board doc_1_1"');
                xinterface_iprintf('"SETVAR doc_1_1 option_menu doc_1_options"');
                xinterface_iprintf('"SETVAR doc_1_1 option_menu_mapped false"');
                xinterface_iprintf('"SETVAR doc_1_1 previous doc_1"');
                xinterface_iprintf('"SETVAR doc_1_1 backward_sensitive true"');
                xinterface_iprintf('"SETVAR doc_1_1 system_mapped false"');
                xinterface_iprintf('"SETVAR doc_1_1 system_userData nil"');

            xinterface_create_widget('"doc_1_2_options doc_1_2_options"');
            xinterface_create_widget('"bulletin_board doc_1_2"');
                xinterface_iprintf('"SETVAR doc_1_2 option_menu doc_1_2_options"');
                xinterface_iprintf('"SETVAR doc_1_2 option_menu_mapped true"');
                xinterface_iprintf('"SETVAR doc_1_2 previous doc_1"');
                xinterface_iprintf('"SETVAR doc_1_2 backward_sensitive true"');
                xinterface_iprintf('"SETVAR doc_1_2 system_mapped false"');
                xinterface_iprintf('"SETVAR doc_1_2 system_userData nil"');

            xinterface_create_widget('"bulletin_board Xdialog"');
                xinterface_iprintf('"SETVAR Xdialog option_menu doc_1_2_options"');
                xinterface_iprintf('"SETVAR Xdialog option_menu_mapped false"');
                xinterface_iprintf('"SETVAR Xdialog previous doc_1_2"');
                xinterface_iprintf('"SETVAR Xdialog backward_sensitive true"');
                xinterface_iprintf('"SETVAR Xdialog system_mapped true"');
                xinterface_iprintf('"SETVAR Xdialog system_userData Xdialog &"');

            xinterface_create_widget('"bulletin_board Xmemo"');
                xinterface_iprintf('"SETVAR Xmemo option_menu doc_1_2_options"');
                xinterface_iprintf('"SETVAR Xmemo option_menu_mapped false"');
                xinterface_iprintf('"SETVAR Xmemo previous doc_1_2"');
                xinterface_iprintf('"SETVAR Xmemo backward_sensitive true"');
```

```
                    xinterface_iprintf("'SETVAR Xmemo system_mapped true"');
                    xinterface_iprintf("'SETVAR Xmemo system_userData Xmemo &"');

                xinterface_create_widget("'bulletin_board Xbiff"');
                    xinterface_iprintf("'SETVAR Xbiff option_menu doc_1_2_options"');
                    xinterface_iprintf("'SETVAR Xbiff option_menu_mapped false"');
                    xinterface_iprintf("'SETVAR Xbiff previous doc_1_2"');
                    xinterface_iprintf("'SETVAR Xbiff backward_sensitive true"');
                    xinterface_iprintf("'SETVAR Xbiff system_mapped false"');
                    xinterface_iprintf("'SETVAR Xbiff system_userData nil"');

                xinterface_create_widget("'bulletin_board Xchez"');
                    xinterface_iprintf("'SETVAR Xchez option_menu doc_1_2_options"');
                    xinterface_iprintf("'SETVAR Xchez option_menu_mapped false"');
                    xinterface_iprintf("'SETVAR Xchez previous doc_1_2"');
                    xinterface_iprintf("'SETVAR Xchez backward_sensitive true"');
                    xinterface_iprintf("'SETVAR Xchez system_mapped false"');
                    xinterface_iprintf("'SETVAR Xchez system_userData nil"');

                xinterface_create_widget("'bulletin_board Xcalc"');
                    xinterface_iprintf("'SETVAR Xcalc option_menu doc_1_2_options"');
                    xinterface_iprintf("'SETVAR Xcalc option_menu_mapped false"');
                    xinterface_iprintf("'SETVAR Xcalc previous doc_1_2"');
                    xinterface_iprintf("'SETVAR Xcalc backward_sensitive true"');
                    xinterface_iprintf("'SETVAR Xcalc system_mapped true"');
                    xinterface_iprintf("'SETVAR Xcalc system_userData Xcalc &"');

                xinterface_create_widget("'bulletin_board Xcanvas"');
                    xinterface_iprintf("'SETVAR Xcanvas option_menu doc_1_2_options"');
                    xinterface_iprintf("'SETVAR Xcanvas option_menu_mapped false"');
                    xinterface_iprintf("'SETVAR Xcanvas previous doc_1_2"');
                    xinterface_iprintf("'SETVAR Xcanvas backward_sensitive true"');
                    xinterface_iprintf("'SETVAR Xcanvas system_mapped true"');
                    xinterface_iprintf("'SETVAR Xcanvas system_userData \
xterm -e csh -c \'Xcanvas\' &"');

                xinterface_create_widget("'bulletin_board Xps"');
                    xinterface_iprintf("'SETVAR Xps option_menu doc_1_2_options"');
                    xinterface_iprintf("'SETVAR Xps option_menu_mapped false"');
                    xinterface_iprintf("'SETVAR Xps previous doc_1_2"');
                    xinterface_iprintf("'SETVAR Xps backward_sensitive true"');
                    xinterface_iprintf("'SETVAR Xps system_mapped true"');
                    xinterface_iprintf("'SETVAR Xps system_userData \
xterm -e csh -c \'Xps\' &"');

                xinterface_create_widget("'bulletin_board Xgraphed"');
                    xinterface_iprintf("'SETVAR Xgraphed option_menu doc_1_2_options"');
                    xinterface_iprintf("'SETVAR Xgraphed option_menu_mapped false"');
                    xinterface_iprintf("'SETVAR Xgraphed previous doc_1_2"');
                    xinterface_iprintf("'SETVAR Xgraphed backward_sensitive true"');
                    xinterface_iprintf("'SETVAR Xgraphed system_mapped true"');
                    xinterface_iprintf("'SETVAR Xgraphed system_userData \
xterm -e csh -c \
```

```
\'(echo \\\"fred\\\"; \
echo \\\"   jim friend\\\"; \
echo \\\"   steve friend\\\"; \
echo \\\"   eric brother\\\"; \
echo \\\"   bill enemy\\\"; \
echo \\\"   simon enemy\\\"; \
echo \\\"steve\\\"; \
echo \\\"   bill enemy\\\"; \
echo \\\"   simon friend\\\"; \
echo \\\"   eric friend\\\"; \
echo \\\"roger\\\"; \
echo \\\"   simon friend\\\"; \
echo \\\"   jim brother\\\"; \
echo \\\"maurice\\\"; \
echo \\\"   fred friend\\\"; \
echo \\\"   lois spouce\\\") | Xgraphed -scale 3.0\' &"');

        xinterface_create_widget("'bulletin_board Xgmacs"');
            xinterface_iprintf("'SETVAR Xgmacs option_menu doc_1_2_options"');
            xinterface_iprintf("'SETVAR Xgmacs option_menu_mapped false"');
            xinterface_iprintf("'SETVAR Xgmacs previous doc_1_2"');
            xinterface_iprintf("'SETVAR Xgmacs backward_sensitive true"');
            xinterface_iprintf("'SETVAR Xgmacs system_mapped true"');
            xinterface_iprintf("'SETVAR Xgmacs system_userData Xgmacs &"');

        xinterface_create_widget("'bulletin_board Xdoc"');
            xinterface_iprintf("'SETVAR Xdoc option_menu doc_1_2_options"');
            xinterface_iprintf("'SETVAR Xdoc option_menu_mapped false"');
            xinterface_iprintf("'SETVAR Xdoc previous doc_1_2"');
            xinterface_iprintf("'SETVAR Xdoc backward_sensitive true"');
            xinterface_iprintf("'SETVAR Xdoc system_mapped false"');
            xinterface_iprintf("'SETVAR Xdoc system_userData nil"');

        xinterface_create_widget("'doc_1_3_options doc_1_3_options"');
        xinterface_create_widget("'bulletin_board doc_1_3"');
            xinterface_iprintf("'SETVAR doc_1_3 option_menu doc_1_3_options"');
            xinterface_iprintf("'SETVAR doc_1_3 option_menu_mapped true"');
            xinterface_iprintf("'SETVAR doc_1_3 previous doc_1"');
            xinterface_iprintf("'SETVAR doc_1_3 backward_sensitive true"');
            xinterface_iprintf("'SETVAR doc_1_3 system_mapped false"');
            xinterface_iprintf("'SETVAR doc_1_3 system_userData nil"');

        xinterface_create_widget("'bulletin_board doc_1_3_1"');
            xinterface_iprintf("'SETVAR doc_1_3_1 option_menu doc_1_3_options"');
            xinterface_iprintf("'SETVAR doc_1_3_1 option_menu_mapped false"');
            xinterface_iprintf("'SETVAR doc_1_3_1 previous doc_1_3"');
            xinterface_iprintf("'SETVAR doc_1_3_1 backward_sensitive true"');
            xinterface_iprintf("'SETVAR doc_1_3_1 system_mapped false"');
            xinterface_iprintf("'SETVAR doc_1_3_1 system_userData nil"');

        xinterface_create_widget("'doc_1_3_2_options doc_1_3_2_options"');
        xinterface_create_widget("'bulletin_board doc_1_3_2"');
            xinterface_iprintf("'SETVAR doc_1_3_2 option_menu doc_1_3_2_options"');
```

```
                    xinterface_iprintf("'SETVAR doc_1_3_2 option_menu_mapped true"');
                    xinterface_iprintf("'SETVAR doc_1_3_2 previous doc_1_3"');
                    xinterface_iprintf("'SETVAR doc_1_3_2 backward_sensitive true"');
                    xinterface_iprintf("'SETVAR doc_1_3_2 system_mapped false"');
                    xinterface_iprintf("'SETVAR doc_1_3_2 system_userData nil"');

                xinterface_create_widget("'doc_1_3_3_options doc_1_3_3_options"');
                xinterface_create_widget("'bulletin_board doc_1_3_3"');
                    xinterface_iprintf("'SETVAR doc_1_3_3 option_menu doc_1_3_3_options"');
                    xinterface_iprintf("'SETVAR doc_1_3_3 option_menu_mapped true"');
                    xinterface_iprintf("'SETVAR doc_1_3_3 previous doc_1_3"');
                    xinterface_iprintf("'SETVAR doc_1_3_3 backward_sensitive true"');
                    xinterface_iprintf("'SETVAR doc_1_3_3 system_mapped false"');
                    xinterface_iprintf("'SETVAR doc_1_3_3 system_userData nil"');

                    xinterface_iprintf("'CALLBACK doc_1 mapCallback"');
                };
            };
    };

object
    doc_text : XmScrolledText {
        callbacks {
            MrmNcreateCallback = procedures {
                xinterface_TextSetString("'Xinterface.txt"');
            };
        };
    };

object
    cancel_button : XmPushButton {
            callbacks {
                XmNactivateCallback = procedures {
                    xinterface_exit("'0"');
                };
        };
    };

object
    backward_button : XmPushButton {
            callbacks {
                XmNactivateCallback = procedures {
                    xinterface_iprintf("'CALLBACK %s unmapCallback", \
                                                    Registry(current).name');
                    xinterface_iprintf("'CALLBACK %s mapCallback", \
                                                    Registry().previous');
                };
            };
    };


object
    forward_button : XmPushButton {
```

```
        callbacks {
            XmNactivateCallback = procedures {
                xinterface_iprintf('"CALLBACK %s unmapCallback", \
                                                    Registry(current).name');
                xinterface_iprintf('"CALLBACK %s mapCallback", \
                                                    Resources().userData');
            };
        };
    };


object
    cursorPosition_button : XmPushButton {
        callbacks {
            XmNactivateCallback = procedures {
                xinterface_iprintf('"SETARG doc_text cursorPosition %s", \
                                                    Resources().userData');
            };
        };
    };


object
    system_button : XmPushButton {
        callbacks {
            XmNactivateCallback = procedures {
                xinterface_system('"%s", \
                                                    Resources().userData');
            };
        };
    };


object
    bulletin_board : XmBulletinBoard {
        callbacks {
            XmNmapCallback = procedures {
                xinterface_iprintf('"SETARG doc_text cursorPosition %s", \
                                    Resources().userData');
                xinterface_iprintf('"SETARG %s mappedWhenManaged %s", \
                                    Registry().option_menu, \
                                    Registry().option_menu_mapped');
                xinterface_register_widget('"%s current", \
                                    Registry().name');
                xinterface_iprintf('"SETVAR backward_button previous %s", \
                                    Registry().previous');
                xinterface_iprintf('"SETARG backward_button sensitive %s", \
                                    Registry().backward_sensitive');
            xinterface_iprintf('"SETARG system_button userData %s", \
                            Registry().system_userData');
            xinterface_iprintf('"SETARG system_button mappedWhenManaged %s", \
```

```
                                Registry().system_mapped');
        };
            XmNunmapCallback = procedures {
                    xinterface_deregister_widget("'current"');
                    xinterface_iprintf("'SETARG %s mappedWhenManaged false", \
                                        Registry().option_menu');
        };
         };
    };


object
    doc_1_options : XmOptionMenu {
            controls {
               XmPulldownMenu {
                    callbacks {
                       MrmNcreateCallback = procedures {
                          xinterface_create_widget("'forward_button doc_1_1b"');
                          xinterface_create_widget("'forward_button doc_1_2b"');
                          xinterface_create_widget("'forward_button doc_1_3b"');
                       };
                    };
               };
            };
    };


object
    doc_1_2_options : XmOptionMenu {
            controls {
               XmPulldownMenu {
                    callbacks {
                       MrmNcreateCallback = procedures {
                          xinterface_create_widget("'forward_button \
                                    doc_1_2_Xdialog"');
                          xinterface_create_widget("'forward_button \
                                    doc_1_2_Xmemo"');
                          xinterface_create_widget("'forward_button \
                                    doc_1_2_Xbiff"');
                          xinterface_create_widget("'forward_button \
                                    doc_1_2_Xchez"');
                          xinterface_create_widget("'forward_button \
                                    doc_1_2_Xcalc"');
                          xinterface_create_widget("'forward_button \
                                    doc_1_2_Xcanvas"');
                          xinterface_create_widget("'forward_button \
                                    doc_1_2_Xps"');
                          xinterface_create_widget("'forward_button \
                                    doc_1_2_Xgraphed"');
                          xinterface_create_widget("'forward_button \
                                    doc_1_2_Xgmacs"');
                          xinterface_create_widget("'forward_button \
                                    doc_1_2_Xdoc"');
```

```
                    };
                };
            };
        };

object
    doc_1_3_options : XmOptionMenu {
        controls {
            XmPulldownMenu {
                callbacks {
                    MrmNcreateCallback = procedures {
                        xinterface_create_widget("'forward_button \
                                doc_1_3_b1"');
                        xinterface_create_widget("'forward_button \
                                doc_1_3_b2"');
                        xinterface_create_widget("'forward_button \
                                doc_1_3_b3"');
                    };
                };
            };
        };
    };


object
    doc_1_3_2_options : XmOptionMenu {
        controls {
            XmPulldownMenu {
                callbacks {
                    MrmNcreateCallback = procedures {
                    xinterface_create_widget("'cursorPosition_button \
                                doc_1_3_2_1"');
                        xinterface_create_widget("'cursorPosition_button \
                                doc_1_3_2_2"');
                        xinterface_create_widget("'cursorPosition_button \
                                doc_1_3_2_3"');
                        xinterface_create_widget("'cursorPosition_button \
                                doc_1_3_2_4"');
                        xinterface_create_widget("'cursorPosition_button \
                                doc_1_3_2_5"');
                        xinterface_create_widget("'cursorPosition_button \
                                doc_1_3_2_6"');
                        xinterface_create_widget("'cursorPosition_button \
                                doc_1_3_2_7"');
                        xinterface_create_widget("'cursorPosition_button \
                                doc_1_3_2_8"');
                        xinterface_create_widget("'cursorPosition_button \
                                doc_1_3_2_9"');
                        xinterface_create_widget("'cursorPosition_button \
                                doc_1_3_2_10"');
                        xinterface_create_widget("'cursorPosition_button \
                                doc_1_3_2_11"');
```

```
                                xinterface_create_widget("'cursorPosition_button \
                                        doc_1_3_2_12"');
                                xinterface_create_widget("'cursorPosition_button \
                                        doc_1_3_2_13"');
                            };
                        };
                    };
                };
        };

    object
        doc_1_3_3_options : XmOptionMenu {
                controls {
                    XmPulldownMenu {
                            callbacks {
                                MrmNcreateCallback = procedures {
                                xinterface_create_widget("'cursorPosition_button \
                                        doc_1_3_3_1"');
                                xinterface_create_widget("'cursorPosition_button \
                                        doc_1_3_3_2"');
                                xinterface_create_widget("'cursorPosition_button \
                                        doc_1_3_3_3"');
                            };
                        };
                    };
                };
        };

    end module;
```

FILES

/home/resstaff/lib/X11/app-defaults/XInterface

SEE ALSO

X(1), Xinterface(1), uil(7)

BUILDING NEW INTERFACES

UIL and Xinterface

Users/programmers who wish to develop their own interfaces may use Xinterface as a high level tool which provides access to the X Window System facilities and the Motif Widget Set.

Xinterface provides the mechanism to support the UIL specification. It uses the Motif Resource Manager to access the UID files and install the description defined therein. The UIL description may access the Motif widget set and so may quickly and easily define a quite complicated interface.

The UID files are accessible to Xinterface by either specifying their name including path to the "-uid" argument or by setting the XINTERFACE_PATHS environment variable to include the head of the path and specifying the tail of the path to the "-uid" argument.

In addition to supporting the UIL description Xinterface also supplies the following set of callback routines:

xinterface_create_widget
– create and register a UIL object.

xinterface_destroy_widget
– destroy a specified UIL object.

xinterface_manage_widget
– manage an existing widget.

xinterface_unmanage_widget
– unmanage an existing widget.

xinterface_register_widget
– register an object with Xinterface.

xinterface_deregister_widget
– remove a UIL object from the name register.

xinterface_iprintf
– initiate inter–object communication with another object.

xinterface_fprintf
– print a string to a stream.

xinterface_args
– load and execute command line arguments.

xinterface_system
– execute a system command.

xinterface_TextGetString
– write the value of a Text object to a file.

xinterface_TextSetString
– set the value of a Text object from a file.

xinterface_exit
– exit Xinterface.

Each of these is discussed in more detail in the section entitled 'Xinterface Callbacks'.

These routines allow an interface to take advantage of the following Xinterface Features:

- user controlled dynamic modification through command line arguments.
- external dynamic modification by the calling process.
- internal modification via inter-object communication.
- event communication with the calling process.
- independant control.
- user configurability via the resource manager.

The X window system has attempted to "provide mechanism not policy". To this end many facilities have been provided for allowing the user to specify their preferences as to the "look and feel" of an interface.

For example, by placing the following line:

Xdialog*.dialog_quit_button.mappedWhenManaged: true

in your .Xdefaults file you can cause Xdialog to map (make visible) the QUIT button.

When constructing a UIL interface description special attention needs to be paid to this issue. The arguments section of each UIL object allows the defaults for each field to be overridden. THESE ARGUMENT VALUES OVERRIDE ALL ELSE. Where it is reasonable to respect user preferences the argument values should be specified in the resource files.

By specifying the "-name interface_name" option to Xinterface your interface becomes known as interface_name. Then the resource files can unambiguously set properties of your interface by specifying properties as follows:

interface_name[{./*}path]{./*}property_name: value

The file /usr/X11/lib/app-defaults/XInterface specifies all argument value pairs for interfaces using Xinterface. Programmer defaults should be added there. Doing this allows user preferences specified in the users .Xdefaults file or passed in as a command line argument to override the programmer preferences.

The specification for the interface provided to Xinterface must be in the form of one or more User Interface Description (UID) files. This is a compiled User Interface Language (UIL) description.

When writing a UIL description it should be noted that Xinterface expects an object named "xinterface_main" to be the root of the interface hierarchy.

Xinterface provides a set of callback routines which may be specified in the UIL description. The file "Xinterface.uil" contains the declaration of these procedures. When writing a UIL module for Xinterface this file should be included as follows:

```
module <module_name>
    [clause section]

include file 'Xinterface.uil';

        .
        .
        .

end module;
```

To compile the UIL description you have written simply run the uil compiler with your file as an argument. The output file can be provided as a –uid argument to Xinterface.

For further help with constructing interface specifications for Xinterface see the UIL Programmers Guide.

Xinterface Callback Functions

xinterface_create_widget

procedure
xinterface_create_widget('format, [args ...]':string);

The format argument is evaluated as a C sprintf statement and arguments are sought when required. The resulting string must be of the following form:

"object_name [registered_name]"

The object known as *object_name in the UIL file will registered name* argument is provided the widget will be entered in the registry under that name. If no registered_name is supplied then the object will not be registered.

The registered name is used in resource retrieval for the new instance of the object. This allows a template of an object to be specified and multiple instances to be created and defined in resource files.

Another of the major benefits of using this technique is that it allows major portions of the interface to be loaded when needed. This reduces run-time size and startup time.

xinterface_destroy_widget

procedure
xinterface_destroy_widget('format, [args ...]':string);

The format argument is evaluated as a C sprintf statement and arguments are sought when required. The resulting string must be of the following form:

"object_name"

This removes the entry for the object known as *object_name* in the widget registry from the widget registry and destroys the associated widget.

xinterface_manage_widget

procedure
xinterface_manage_widget('format, [args ...]':string);

The format argument is evaluated as a C sprintf statement and arguments are sought when required. The resulting string must be of the following form:

"object_name"

XtManageChild will be invoked with the object whose registered name is *object_name* as argument. The object must already have been created and must be registered as *object_name*.

This provides a quick mechanism for modifying the appearance of the interface.

xinterface_unmanage_widget

procedure
xinterface_unmanage_widget('format, [args ...]':string);

The format argument is evaluated as a C sprintf statement and arguments are sought when required. The resulting string must be of the following form:

"object_name"

XtUnmanageChild will be invoked with the object whose registered name is *object_name* as argument. The object must already have been created and must be registered as *object_name*.

This provides a quick mechanism for modifying the appearance of the interface.

xinterface_register_widget

procedure
xinterface_register_widget('format, [args ...]':string);

The format argument is evaluated as a C sprintf statement and arguments are sought when required. The resulting string must be of the following form:

"object_name [alias]"

If the *alias* argument is present then the *object_name* entry in the registry is augmented with the given alias.

If the alias tag is missing then the active object will be registered under the name *object_name* in the widget registry. An object may register itself under multiple names or may put multiple entries under the same name. Objects are responsible for deleting their entries when they are destroyed.

xinterface_deregister_widget

procedure
xinterface_deregister_widget('format, [args ...]':string);

The format argument is evaluated as a C sprintf statement and arguments are sought when required. The resulting string must be of the following form:

"object_name"

This removes the first entry for the object known as *object_name* in the widget registry from the widget registry.

xinterface_iprintf

procedure
xinterface_iprintf('format, args':string);

Calling xinterface_iprintf('format, args') causes inter-object communication to occur. The C printf syntax is used. After the message string has been built it is sent to the communication module which expects a string in the standard communication format:

The message is sent to the object registered as *object_name*. *See Xinterface_fprintf* for details of the valid arguments.

xinterface_fprintf

procedure
xinterface_fprintf('stream, format, args':string);

Calling xinterface_fprintf('stream, format, args') causes the string defined by *format* with arguments *args* to be printed over stream *stream*.

xinterface_args

procedure
xinterface_args('format, [args ...]':string);

The format argument is evaluated as a C sprintf statement and arguments are sought when required. The resulting string must be of the following form:

"object_name"

This installs the command line arguments for the object whose is registered as object_name. If no object is registered as object_name the call does nothing.

Command line arguments which are proceeded by a "-arg" and are of the standard communication form.

When "xinterface_args(object_name)" is invoked each argument on the command line which is proceeded by a "-arg" is examined and if the object names match the communication is sent.

Command line args are reusable.

xinterface_system

procedure
xinterface_system('format, [args ...]':string);

The format argument is evaluated as a C sprintf statement and arguments are sought when required. The resulting string is used as the argument to the C system call.

xinterface_TextGetString

procedure
xinterface_TextGetString('format, [args ...]':string);

The format argument is evaluated as a C sprintf statement and arguments are sought when required. The resulting string is taken as the filename to write the contents of the current TEXT object to. If the filename is the NULL string ('') then the file is stdout.

The active object must be of type XmText or XmScrolledText.

xinterface_TextSetString

procedure
xinterface_TextSetString('format, [args ...]':string);

The format argument is evaluated as a C sprintf statement and arguments are sought when required. The resulting string is taken as the filename to read and set the value of the current TEXT object to be set to the contents of the file.

The active object must be of type XmText or XmScrolledText.

xinterface_exit

procedure
xinterface_exit(('format, [args ...]':string);

The format argument is evaluated as a C sprintf statement and arguments are sought when required. The resulting string must be the ascii representation of an integer. This integer is extracted and Xinterface terminates with that exit status.

Arguments to Callback Functions

All of Xinterface's callback functions accept a format string and arguments in C-like syntax.

A callback function can access the following information:
- The callback structure.
- Resources for the active object.
- Resources for a registered object.
- The name of a registered object.
- The variables for a registered object.

The callback structure contains, at least, the:
- reason identifier.
- the XEvent structure.

Both of these are accessible. The type of event structure depends on the type of event which caused the callback method to be triggered. All fields of all types of XEvent structures are accessible. The following prints the button number field in an XButtonPress event structure:

```
xinterface_fprintf('stdout, "%d\n", XmAnyCallbackStruct.XEvent.button');
```

For a description of the various XEvent structures see the Xlib Reference Manual Vol. 2. A good example of this is given in the Xps UIL file in the demos.

Resource values for the active object may be referenced by using the Resources function. The syntax is: Resources().resource_name. The following prints the userData resource field for the active object.

```
xinterface_fprintf('stdout, "%s\n", Resources().userData');
```

The Resources function also has a more general form. The general syntax is: Resources(object).resource_name. The object parameter specifies a name or alias of an object registered in the object registry. Xinterface's object registry allows an object to register itself so that other objects can access it. This allows an arbitrary object who wishes to access the resources or registry of another object to refer to it by name. The object registry also allows alternate names for an object. I use this facility for registering the active bulletin board as "current" in the Xdoc.uil file. This allows an arbitrary object who wishes to access resources belonging to the "current" widget to do so. The following prints the userData field of the object registered as current.

```
xinterface_fprintf('stdout, "%s\n", Resources(current).userData');
```

Resources for a given object depend on the type of object. See appendix B of the UIL Programmers Guide for a list of resources for each object.

Once an object is registered it has at least a name. In addition it may have:

- alternate names.
- variables.

These attributes can be accessed by the object itself or by any object knowing its name or an alias (alternate name) for the object. The following prints the name field of the active object if it has been registered.

```
xinterface_fprintf('stdout, "%s\n", Registry().name');
```

The object registry also contains the list of variables associated with an object. Variables are simply name-value pairs where both name and value are strings. Setting a variable is accomplished with the SETVAR command which works much like the Unix set command (if the variable exists its value is updated; else it is created). The following sets the value of the variable named foo_bar of the object named or aliased object_name to hello world and then prints it.

```
xinterface_iprintf('"SETVAR object_name foo_bar hello world\n"');
xinterface_fprintf('stdout, "%s\n", Registry(object_name).foo_bar');
```

A good example of this is given in the Xdoc UIL file in the demos.

Communicating With XInterface

Communicating TO XInterface

There are two forms of communication to Xinterface. First Xinterface provides a method by which the calling process can dynamically modify the interface. Communication is through UNIX pipes. Xinterface receives messages through standard input and sends messages through standard output. Communication with the caller can be enabled by specifying the –input argument to Xinterface.

Specifically, starting Xinterface with the –input argument causes Xinterface to add stdin as a possible source of input events. Messages can then be passed to Xinterface by writing over this pipe. This mechanism allows the calling process to set the resource values of objects, draw postscript on objects, or activate callback methods belonging to objects in the interface.

Mesages can only be sent to objects who have registered themselves. This provides a public/private feature for objects.

A message takes the following form:

```
communication ::=
        communication_type_1
      | communication_type_2
      | communication_type_3
      | communication_type_4

communication_type_1 ::=
      CALLBACK  object_name  callback_argument

communication_type_2 ::=
      SETARG  object_name  argument_name  argument_value

communication_type_3 ::=
```

SETVAR object_name variable_name variable_value

communication_type_4 ::=
POSTSCRIPT object_name postscript_code

object_name ::=
{the registered name of the object}

callback_argument ::=
{the name of the callback reason for the object}

argument_name ::=
{the name of the argument to set in the object}

argument_value ::=
{the new value for argument_name}

variable_name ::=
{the name of the variable to set in the object}

variable_value ::=
{the new value for variable_name}

postscript_code ::=
{the postscript code to process}

In all cases a communication is passed as a single string up to the end of line character with the arguments delimited by white space. Xinterface parses the string into its components.

The CALLBACK message invokes the callback specified by callback_argument belonging to the object registered as object_name. The argument value is the name of the callback minus the XmN prefix. Thus to invoke the callback identified by 'XmNactivateCallback' for the object registered as 'fooeyonu' the message would take the following form:

CALLBACK fooeyonu activateCallback

The SETARG message causes the named argument of object_name to take on the value argument_value. Again the XmN prefix is dropped from the argument name. Thus, if an object of type XmLabel is registered as 'fooeyonu' then the string displayed in it can be modified to "hello world" by sending the following message:

SETARG fooeyonu labelString hello world

The SETVAR message causes the variable associated with the object registered as object_name to take on the value var_value. If an object is registered as 'fooeyonu' then any of the variables can be changed. For example, the value of the variable named "var1" can be set to "hello world" by sending the following message:

SETVAR fooeyonu var1 hello world

The POSTSCRIPT message causes the given postscript code to be sent to the interpreter associated with the object registered as *object_name*.

A communications pipe remains open until an end of file is detected; it is then closed and disregarded as a source of input.

Xinterface also provides dynamic command line argument access. That is, an object may request its arguments at any time during execution. This allows the interface to dynamically configure itself to the users specifications.

Command line arguments which are proceeded by a "-arg" flag are interpreted as communications of the same form as external or internal communications.

In all cases a communication is passed as a single string with the arguments delimited by white space. Xinterface parses the string into its components.

Command line arguments are handled almost identically to inter-object communications or caller->Xinterface communications. The exception being that the arguments:

- must be requested (via xinterface_args).
- are still available for later access.

The communication protocol is otherwise identical. Thus anything that can be communicated to an object can be given as an argument to it.

This not only provides a mechanism for dynamically modifying the "look and feel" of the interface; but also the functionality. Since communications may be used to trigger callbacks the action performed by an activated object may be determined by the command line arguments. That is, an object may request its command line arguments when it is activated and in doing so may trigger other callback functions. For an example of this see Xdialog (under Existing Xinterfaces).

Since objects requests command line arguments there need not be a correspondence between the actual or registered object name and the name used in the command line arguments. The object whose name corresponds with the name used for retrieval will receive the arguments. Thus one object may cause another to receive its command line arguments.

To enable its arguments to be used an object must register itself with xinterface_register_widget(object_name) and then any object may call xinterface_args(object_name) to retrieve all arguments for object_name.

Communicating FROM Xinterface

Communication from Xinterface is simply the result of a call to *xinterface_fprintf*. This callback routine prints a message to a stream. It must be called as one of the routines in a callback method and so is simply notifying the caller that the method has been activated.

Communicating IN Xinterface

Xinterface provides a method by which an active object can dynamically modify or activate another unrelated object. This "inter-object communication" can occur between any two objects provided that the receiver has registered itself in the widget registry.

An example of the use of this facility is provided by Xdialog. When the OK button is pressed the OK buttons activateCallback must send an activate message to the EDITOR. This is necessary since only the editor can access its contents.

A message can be sent to any registered widget (to its registered address) and is of the same form as that of external communication.

Inter-object communication is performed through the callback function *xinterface_iprintf* so the sender must be active in a callback method.

The receiver must be registered in the widget registry.

X DEFAULTS

This program understands all of the core UIL object class names and resources as well as the names of any objects specified in the **UIL** files. The Class name *XInterface* may be used to specify the resource values for all clients provided by this program. The *-name* argument specified to this program establishes the base name for resource management which is by default *Xinterface*.

FILES

/home/resstaff/include/X11/Xinterface/Xinterface.uil, /home/resstaff/lib/X11/Xinterface/psrc, /home/resstaff/lib/X11/app-defaults/XInterface,

SEE ALSO

uil(1), uil(7), Xdialog(1), Xmemo(1), Xchez(1), Xcalc(1), Xcanvas(1), Xps(1), Xgraphlayout(1), xps(1)

BUGS

- Some nested UIL objects access their resources improperly.

- The -rv flag does not affect the display.

AUTHOR

John D. Lewis
Software Research and Development Group
University of Calgary
Calgary T2N 1N4, Canada
Phone (403) 220-6780
email: lewis@cpsc.UCalgary.CA