The Helios X Window System Manual

PERIHELION SOFTWARE LTD

May 1991

Copyright

This manual refers to the X Window System V11 release 4. The information within this manual was partially based on material in the following two documents: *The XLib C Language X Interface*, by Jim Gettys, Ron Newman, and Robert Scheifler, and *The X Window System Protocol, Version 11*, by Robert Scheifler and Ron Newman. These two documents are copyright (c) 1985, 1986, 1987, 1988, 1989 the Massachusetts Institute of Technology, Cambridge, Massachusetts, The United States of America, and Digital Equipment Corporation, Maynard, Massachusetts, The United States of America. The material was used according to the terms of the copyright, which grants free use, subject only to the following conditions:

> 'Permission to use, copy, modify and distribute this documentation (*that is, the original MIT or DEC documents*) for any purpose and without fee is hereby granted, provided that the above copyright notice appears in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of MIT or Digital not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. MIT and Digital make no representations about the suitability of the software described herein for any purpose. It is provided "as is" without expressed or implied warranty.'

As all X Window System manuals are ultimately based upon the original MIT work, a rough similarity between this manual and others describing X is clearly unavoidable. Nevertheless, some care has been taken to give the description of the material a new slant whilst still describing the same software.

The X Window System software copyright is held by the members of the X Consortium, formed by MIT and others in 1986. Permission for the use of X is granted to any party interested in implementing it.

This manual was written by Nick Clifton, with the editorial assistance of Ian Davies.

Part number DM5049

Contents

1

Introduction

1.1 What is provided ?

This product is a complete implementation of the X11 protocol at both the client and server ends. It is based on the X11 release 4 distribution tapes (patched to level 18) from MIT and includes all of the features of that server (including the *SHAPE*, *MULTI_BUFFERING* and *MITMISC* extensions), and most of the clients. This product includes all of the libraries and header files necessary to compile X clients, including those that use the *X toolkit* and *Athena Widget* libraries. The product makes use of the new features available in *Helios 1.2.1* including sockets, BSD support, pseudo-terminals, and ethernet support.

1.2 What is missing ?

This product does not include the files needed for compiling *Motif* or *OpenLook* X clients, these files are scheduled for inclusion in future products. The following standard X clients are not provided:

xditview	Helios does not currently support DVI format files
xdm	Too UNIX specific
xinit	Unnecessary, startx shell script or initrc work fine
xman	Unix style manual pages not supported by Helios
xmh	Unix mailer not yet ported to Helios

In addition a large number of the clients contributed to X11R4 tapes have not yet been ported. All of these clients are available in the public domain, along with numerous other X programs, games, etc.

1.3 What is required ?

In order to use this package you will need the following :

- A transputer network of one or more T425s, T800s and / or T805s. **Note,** T200s and T400s are *not* supported.
- *Helios 1.2.1.*
- Lots of disc space (preferably $\geq$ 30 megabytes).
- Lots of RAM on the transputers (preferably a minimum of 2 megabytes per transputer involved with X).

Optionally you may also want :

- A transputer with graphics hardware and VRAM and a mouse attached to your host machine, (in order to run the X server).
- An ethernet connection supported by the *Helios Ethernet Package* (in order to run X clients connected over the ethernet).
- More RAM (lots more RAM . . .).
- More transputers.

If you want to build your own X clients, or you have a graphics board not currently supported by this package you will also need :

- The *Helios C Compiler* (version 2.0 or later).
- The *Helios Assembler.*
- The *Helios Compiler Driver* (version 1.46 or later).
- The *Helios Assembler Macro Pre-Processor* (for making device drivers).

1.4 What is on the discs ?

The discs or tape that comes with this package contain the following directories. These directories are intended to be installed into your existing Helios world, starting from the */helios* directory, (file names appear in parentheses) :

	Resource files for X clients (**Xauthority, Xdefaults, twmrc, uwmrc**).	
	example X world start up shell script (**startx**).	
bin	The X server (**Xhelios.**)	
bin/X11	*The X clients . . .*	
	Demos	**(aquarium, clover, frac, ico, maze, muncher, plaid, psycho, puzzle, rose, worm, xeyes, xfade, xlock, xmandel)**
	Info	**(appres, listres, showcols, showkeys, showsnf, xdpyinfo, xlsatoms, xlsclients, xlsfonts, xlswins, xprop, xversion, xwininfo)**
	Bitmap editor	**(atobm, bitmap, bmtoa)**
	Font compiler	**(bdftosnf, mkfontdir)**
	Benchmarks	**(gbench, x11perf, xbench, xgc)**
	Logos	**(hroot, perilogo, xlogo)**
	Customisers	**(setkeys, xauth, xhost, xmodmap, xrdb, xset, xsetroot, xstdcmap)**
	Window managers	**(twm, uwm)**
	Utilities	**(xbiff, xcalc, xclipboard, xcutsel, xrefresh)**
	Clocks	**(oclock, xclock)**
	X debugging tools	**(xev, xmag, xscope)**
	Text editor	**(xedit)**
	Terminal emulator	**(xterm)**
	Font tools	**(xfd, xfontsel)**
	Load monitors	**(xlights, xload)**
	Window dump tools	**(xpr, xwd, xwud)**
	Games	**(xtrek)**

bin/X11/pond	Bitmaps used by the aquarium demo.	
etc	Resource files for the X server.	**(newXrc, newXrc.ATW, newXrc.GDS)**
	Example initrc file.	**(initrc.x)**
examples	*Sources of example X clients . . .*	
	Taking over control of the X server.	**(grab.c)**
	Modifying the keyboard map.	**(setkeys.c)**
	Athena Widget demo.	**(hello.c)**
	Load displayer.	**(xlights.c)**
include	Extensions to the Helios **ioevents.h** header file **(xioevents.h)**.	
include/X11	Header files for clients using the (C language) X library and X toolkit library.	
include/X11/Xaw	Header files for clients using the (C language) X Athena Widget library.	
include/X11/Xmu	Header files for clients using the (C language) X Miscellaneous Utilities library.	
include/X11/bitmaps	A collection of bitmaps ready for use.	
include/X11/extensions	Header files for clients using the X server extensions library.	

lib

The X libraries ...
The resident X library (**X, Xlib.def**)
The scanned X library (**libX11.a**)
X miscellaneous utilities (**libXmu.a**)
X extensions (**libXext.a**)
X Display Manager Control Protocol (**libXdmcp.a**)
X Toolkit (**libXt.a**)
X Athena Widget (**libXaw.a**)
Unix emulation (**unix.lib**)

The device drivers ...
Keyboard drivers (**keyboard.d, keyboard.no keyboard.yes**)
Mouse driver (**mouse.d**)
Graphics hardware drivers (**newG300.d, newATW.d, newGDS.d, newsplash.d, Paratech.d**)

Miscellaneous ...
The X/Helios Window Server (**window**)
The Helios pty server (**ttyserv.bak, tpseudo.dbk**)
The Xtrek daemon (**xtrekd**)

lib/X11

The databases ...
The X Error Message Database (**XErrorDB**)
The X Toolkit Error Message Database (**XtErrorDB**)
The colour name database (**rgb.dir, rgb.pag, rgb.txt**)

lib/X11/app-defaults Application specific resource files.

lib/X11/fonts/100dpi Common variable width fonts for 100 dots-per-inch displays.

lib/X11/fonts/75dpi Common variable width fonts for 75 dots-per-inch displays.

lib/X11/fonts/misc Miscellaneous and fixed-width fonts.

lib/X11/fonts/r2 Fonts originally supplied for the X11 release 2 server.

lib/X11/twm Default resource file for the twm window manager.

local/xsrc	The Imake utility (**imake, Imake.rules, Imake.tmpl, Library.Tmpl, Project.tmpl, Server.tmpl, helios.cf, site.def**). Header files used by device drivers (**hardware.h, md.h, mdkeyboard.h, mdmouse.h, mdutils.h**).
local/xsrc/devices	*Source code for device drivers . . .* Keyboard device driver (**keyboard.c**) Mouse device driver (**mouse.c**) Generic graphics hardware device driver (**generic.c**) Example G300 device driver (**newG300.c**) The device driver module header (**device.a, device.o**) *Source code for the device driver test harness . . .* The harness program (**harness, harness.c**) The screen handling routines (**screen.c screen.h**) A device driver reset utility (**reset.c**)

1.5 What are the differences between X for Helios and X for Unix ?

There are several important differences between X running under Helios and X running under Unix. If you are used to X for Unix, or you are reading any of the books on X normally available, you should bear the following points in mind :

- There are two X libraries. The first (just called *X*) is a resident library, containing the most frequently used X library calls. The second (called *libX11.a*) is a scanned library containing all the remaining X library calls. All X clients must be linked with the first X library, (with the option *-lX* to the *Helios Compiler Driver*), but only some will need to be linked with the second, (using the option *-lX11*). If you find that the Helios linker is complaining about missing X library functions, then make sure that you are linking with the second X library as well as the first. (A resident library is shared by all programs running on the same processor, a scanned library is linked in with the program using it, but only those functions actually used by the program are extracted from the scanned library.) The makefile in the examples directory demonstrates how to compile an X client.

- The names of some of the standard X11 header files have had to be changed because they clash with the names of other header files when the MS-DOS or TOS filing systems are being used. The files that have changed are :

  in *include/X11*:

Old Name	*New Name*
CompositeI.h	Composite.Ih
CompositeP.h	Composite.Ph
ConstrainP.h	Constrain.Ph
IntrinsicI.h	Intrinsic.Ih
IntrinsicP.h	Intrinsic.Ph
ResourceI.h	Resource.Ih

  in *include/X11/Xaw*:

Old Name	*New Name*
AsciiSinkP.h	AsciiSink.Ph
AsciiSrcP.h	AsciiSrc.Ph
AsciiTextP.h	AsciiText.Ph
MenuButtonP.h	MenuButton.Ph
ScrollbarP.h	ScrollBar.Ph
SimpleMenuP.H	SimpleMenu.Ph
TemplateP.h	Template.Ph
TextSinkP.h	TextSink.Ph
ViewportP.h	Viewport.Ph

- Similarly the names of some of the X resource files have been changed :

Old Name	*New Name*
.Xdefaults	Xdefaults
.Xdefaults-<hostname>	Xdef-<hostname>
.Xauthority	Xauthority

- Another consequence of these restricted filing systems is that commands whose names are longer than 8 characters (for example **`mkfontdir`**) can only be executed from the command line if their name is truncated to 8 characters (**`mkfontdi`**) or if the full path name is given, (**`/helios/bin/X11/mkfontdir`**), or if your current directory is the directory containing the command, (**`cd /helios/bin/X11 ; mkfontdir`**).

- A bug in the current version(s) of the *Helios C Compiler* means that the *XtOffsetOf()* macro (defined in the header file *Intrinsic.h*) does not work with sub-structures. That is you cannot do *XtOffsetOf(structfred,field.sub_field)*. In order to get around this problem, four new macros are provided, (also defined in *Intrinsic.h*), called :

  XtOffsetOfMul(),
  XtOffsetOfMulMul(),
  XtOffsetOfMulMulMul(),
  XtOffsetOfMulArray()

  which cope with one, two, and three levels of sub-fields and a sub-field with an array as a sub-sub-field, respectively. These macros should be used wherever the *XtOffsetOf()* macro fails to compile, which will most commonly be in static declarations of the *XtResource* structure.

- There is a known bug that drawing wide line ellipses can cause the X server to go into a infinite loop, effectively locking up the transputer upon which it is running.

- The root of the filing system has changed. Whenever you see */usr* mentioned by an X client you should interpret this as */helios*. So for example, application resources normally reside in the directory */usr/lib/X11/app-defaults* on Unix systems, whereas under Helios, they reside in */helios/lib/X11/app-defaults*.

- The Helios X server can support a wide range of different display types, (depending upon the settings in its resource files, and upon which device drivers it loads). Conventional X servers are built to support only a small number of displays, (that is those supplied by the manufacturer of the computer upon which the X server is running). Thus the Helios X server can often report different statistics about the display which it is managing, depending upon the installation in which it is running.

1.6 What are the differences between X for Helios release 4 and release 2 ?

If you have used the previous X product from Perihelion Software (which implemented release 2 of the X11 server and clients), you should be aware of the following changes that have been made for release 4 :

- The X server resource file is now called *newXrc*. (The previous release used a resource file called *Xrc*). The format of this file has changed considerably from the release 2 version. In particular,
  - keywords no longer have to be in upper case, (in fact keywords are now case insensitive).
  - Comments start with a hash character (#) and extend to the end of the line.
  - The first (uncommented) occurrence of a keyword is the one that is used. Any subsequent occurrences are ignored, (and no warning is produced).
  - Variables take the form *<keyword>* = *<value>* where *<keyword>* and *<value>* are sequences of alpha-numerics, underscores, periods and forward slashes.
  - If other characters are required in either field they should be enclosed between double quotes (").
  - The sequence \ *<char>* is treated as *<char>* except that *<char>* is treated as if it were an alpha-numeric. (Note the sequence \# is treated as an alpha-numeric, not the start of a comment). A backslash can be inserted by the sequence \\ .
  - White space between *<keyword>* and the equality sign, and white space between the equality sign and *<value>* is ignored.
  - An alternative form of just *<keyword>* with no equality sign is also accepted. This form specifies a boolean variable (switched on).
  - The form *!<keyword>* specifies a boolean variable (switched off).
- The X toolkit and Athena Widget libraries now work, and do not leak memory.
- This release is much bigger, both in terms of disc space used, and in terms of memory requirements. The X server cannot be practically run in less than 2 megabytes of RAM, and most X clients will need at least a megabyte of RAM in order to run.
- The release 2 fonts are provided (in the directory */helios/lib/X11/fonts/r2*), but they are not included in the X server's default font path. They can be added by the command:

```
xset fp+ /helios/lib/X11/fonts/r2
```

- Graphics hardware device drivers are now also responsible for putting entries into the colour lookup table(s) on the hardware board. This means that (in theory) any kind of CLUT can be supported, and the hardware driver can do things like synchronising installing CLUT entries with the frame flyback events.

- The directory structure has been reorganised to more closely resemble the directory structure used by Unix implementations of X. In particular the X clients directory used to be called */helios/xclients* whereas it is now called */helios/bin/X11* and the X fonts directory used to be called */helios/fonts* whereas it is is now called */helios/lib/X11/fonts/r2*.

- The X server no longer supports the Helios GSP model. It is no longer possible to use the `ls /X` command to see which clients have contacted the X server, and clients cannot talk directly to the server via the *Read()* and *Write()* system calls. This also means that the X library macro called *ConnectionStream()* is now called *ConnectionNumber* and that it returns a POSIX level file index, rather than a Helios level Stream pointer.

2

Installation and Configuration

This chapter covers all that you need to know about installing and configuring your X world. The chapter is best used by reading it through completely first, and then following the instructions on Installation in Section 2.1.

2.1 Installation

In order to install X you should follow these steps:

1. Read Chapter 1 then this chapter.
2. Make sure you have sufficient resources available. For example, do you have enough disc space, the right version of Helios, a graphics TRAM, a mouse, etc ?
3. Make a backup of the discs or tapes (just in case...).
4. Start Helios. Run *loadpac* and select option 4 (install software) then option 11 (install the X Window System). You will be prompted to insert the installation discs (in order) and messages will explain exactly what is going on. The entire process will take about one hour, so be prepared for a long wait.
5. If you have not done so already, reboot Helios.

6. Follow the notes on Configuration in Section 2.3. In particular make sure you have edited your *host.con* file (section 2.3.3 and your *newXrc* file (section 2.3.1.

7. If your graphics hardware is not one of those supported by the Helios X package you will need to make a graphics hardware device driver. See Section 2.7.

8. Try starting the X server, for example :

```
remote -d <graphics transputer>
/helios/bin/Xhelios
```

If it works, the screen on the monitor should be cleared to a grey cross-hatch pattern and a cursor shaped like a black X should be in the top left hand quarter of the screen. The X server should have printed out the following messages (or something very similar):

```
X: loading resource file: /helios/etc/newXrc...loaded
X: loading frame buffer driver: newG300.d...loaded
X: loading keyboard     driver: keyboard.d...loaded
X: loading mouse        driver: mouse.d...loaded
X: reading font directory  /helios/lib/X11/fonts/misc/fonts.dir...succeeded

X: reading font alias file /helios/lib/X11/fonts/misc/fonts.alias...succeeded

X: reading font directory  /helios/lib/X11/fonts/75dpi/fonts.dir...succeeded

X: reading font alias file /helios/lib/X11/fonts/75dpi/fonts.alias...succeeded

X: reading font directory  /helios/lib/X11/fonts/100dpi/fonts.dir...succeeded

X: reading font alias file /helios/lib/X11/fonts/100dpi/fonts.alias...succeeded

X: Server Ready
```

9. Try starting an X client, for example :

```
/helios/bin/X11/ico -display helios:0
```

(This should create a black window with a white, wire-frame icosahedron bouncing about inside it.)

10. Reboot Helios and then follow the notes on starting X in Section 2.4.

11. If you wish to run the game *xtrek* you will need to have installed the *Helios Ethernet Package* and you will need to add the line :

```
xtrek stream tcp wait guest /helios/lib/xtrekd
xtrekd
```

to the file */helios/lib/inetd.conf*.

You should now have successfully installed X.

2.2 Installing X in a restricted space

If you have a limited amount of disc space in which to install X you could try some of the options mentioned below. It is usually best to install all of the X release first, and then remove files as necessary, rather than performing only a partial installation, which could leave some critical files uninstalled.

FONTS *(/helios/lib/X11/fonts)*

The X server comes supplied with a large number of fonts, but it does not actually need all of them in order to work. The *misc* directory should not be deleted, but the other three (*r2*, *100dpi*, *75dpi*) can be removed (probably in that order). If you remove some files from any of these directories, but you do not delete it entirely then you should run the command *mkfontdir*. If you delete any of the directories *100dpi*, *75dpi*, or *misc* then you must create a dummy directory to replace them. To do this type :

```
cd /helios/lib/X11/fonts
mkdir <name-of-deleted-font-directory>
cd <name-of-deleted-font-directory>
cat > fonts.dir
0 <return> <CTRL-D>
touch fonts.alias
```

The font directories are :

Name	*Size*
r2:	2.3 megabytes
100dpi:	3 megabytes
75dpi:	2.5 megabytes
misc:	.7 megabytes

COMMANDS *(/helios/bin/X11)*

A large number of commands are provided with the X package, however few of them are really essential. The bigger commands tend to be the ones that use the X toolkit and Athena Widget libraries. You will probably not want to delete the commands *xterm* (terminal emulator), *twm* (window manager), and *xset* (display configurer).

Name	*Size*
appres:	190 kilobytes (simple resource displayer)
aquarium:	207 kilobytes (simple fishy demo)
clover:	208 kilobytes (simple and buggy colour demo)
listres:	365 kilobytes (simple resource displayer)
oclock:	211 kilobytes (nice circular clock)
xbiff:	218 kilobytes (simple file monitor)
xcalc:	250 kilobytes (simple desktop calculator)
xclipboard:	301 kilobytes (simple cut and paste tool)
xclock:	217 kilobytes (simple clock)
xcutsel:	238 kilobytes (simple cut and paste tool)
xedit:	313 kilobytes (basic editor program)
xeyes:	213 kilobytes (nice moving eyes demo)
xfd:	258 kilobytes (useful font displayer)
xfontsel:	287 kilobytes (not quite as useful font displayer)
xgc:	369 kilobytes (benchmarking program)
xload:	233 kilobytes (unimpressive load monitor)
xmandel:	254 kilobytes (standard mandelbrot demo)
xscope	112 kilobytes (X server debugging tool)

LIBRARIES *(/helios/lib)*

If you do not intend to compile any X clients under Helios, then you can remove the header files and libraries that they would need. (Do not delete the resident X library */helios/lib/X*, as this is needed by compiled X clients.)

Name	*Size*
libX11.a:	294 kilobytes (scanned X library)
libXaw.a:	305 kilobytes (Athena Widget library)
libXt.a:	390 kilobytes (X toolkit library)
libX*.a:	184 kilobytes (remaining X libraries)

HEADER FILES *(/helios/include/X11)*

Again if you do not intend to compile any X clients under Helios then you can delete the header files that they would use.

Name	*Size*
*.h:	498 kilobytes
/.h:	304 kilobytes

DEVICE DRIVERS *(/helios/local/xsrc)*

If you have a device driver that works with your display hardware then you will not need to build your own, and you can delete the directory containing the sources of the example drivers.

Name	*Size*
xsrc:	273 kilobytes

X SERVER *(/helios/bin/Xhelios)*

If you do not have any graphics hardware in your processor network, then you may not want to run the X server and instead just have your X clients connect to an X server somewhere else on the ethernet. In this case you can delete the X server object code and its device drivers.

Name	*Size*
Xhelios:	482 kilobytes

2.3 Configuring the X server

This section describes how the X server can be configured to suit your environment. This is basically done by editing various text files that control the behaviour of the X server.

2.3.1 The X server resource file

Once started up the X server will attempt to read in a resource file called *newXrc*. This file specifies many of the configurable options in the X server. The file itself is fully commented, explaining what the options are and what values they can have. A list of those options which apply specifically to the X server, (rather than to any of the device drivers loaded by the X server), is set out below :

auto_repeat_delay

This is the time (in milliseconds) before a key which is being held down will start to generate repeat key press events.

auto_repeat_interval

This is the time (in milliseconds) that will elapse between key press events generated by the auto-repeat feature.

auto_repeat_filter

This is a list of ASCII codes for keys which should not be allowed to auto-repeat.

hardware_device

This is the file name of the hardware device driver that the X server should use to control the display. Currently device drivers are provided for graphics cards based on an INMOS G300 TRAM (*newG300.d*) and the SPLASH card from Tektite (*newsplash.d*). In addition *untested* drivers are supplied for the Atari Transputer Workstation (*newATW.d*), the ParsyTec GDS graphics board (*newGDS.d*) and the Paratech GM8103/PSG8103 graphics card (Paratech.d).

keyboard_device

This is the file name of keyboard device driver that the X server should load.

mouse_device

This is the file name of mouse device driver that the X server should load.

There are many more parameters in the *newXrc* file, but these are used by the various device drivers loaded by the X server, and not by the X server itself.

The X server will look in the following places (in order) when trying to load a *newXrc* file :

1. The file specified by the X server's command line option *-newXrc*.
2. The file specified in the environment variable *NEWXRC*.
3. The X server's current directory.
4. The user's home directory (as specified by the environment variable *HOME*).
5. The file */helios/etc/newXrc*.

2.3.2 Command line options

The X server understands a large selection of options on its command line. The full list can be found by running the X server with the *-help* option, or by reading the X server's manual page. This section lists a few of the more important options, including a couple which do not appear in the standard Unix implementation.

bc

Enable bug compatibility mode (for buggy release 2 and 3 X clients).

-bs

Disable backing store support (saves memory).

-fp <string>

Specifies the default font path.

-fn <string>

Specifies the default font (normally *fixed*).

-su

Disable save under support (saves memory).

:<number>

Specifies the screen number of the server.

notcp

Do not try to create a socket in the Internet domain. (This option is useful if you do not have the Helios Ethernet product.) [Helios Specific]

-newXrc <path>

Specifies the path name of the X server's resource file (see Section 2.3.1). [Helios Specific]

2.3.3 The Helios I/O server resource file

The Helios I/O server resource file (normally called */helios/host.con*) contains various parameters that control how Helios interacts with its host computer, (usually a PC or clone). If this host machine is being used to provide the mouse and keyboard input to the X server, (the usual situation), then certain parameters in the *host.con* file can affect how the X server behaves. These parameters are :

Xsupport

This *must* be enabled (that is there must be no hash (#) signs between this parameter and the start of the line). This parameter tells the Helios IO server that the X server will be requiring the keyboard and mouse services.

mouse_resolution

This specifies how far the mouse can be moved (on your desk top) before it will send a mouse motion event to the X server. A smaller number means finer grain mouse movements, but lots more messages will be produced, possibly slowing down disc access.

mouse_divisor

This specifies by how much the distance moved by the mouse on the desk top should be reduced when reporting a mouse motion event to the X server. A smaller number means that slight movements of the mouse will cause the cursor on the screen to move further.

It is recommended that you set the *mouse_divisor* to the value 1 (one) and the *mouse_resolution* to 4 (four) for the best results. In addition, if you are using the Microsoft Mouse Driver, then it is recommended that you turn off the acceleration provided by that driver. This can usually be done by changing the command :

```
REM C:\MOUSE1\SETSPEED /P2 /FC:\MOUSE1\mousepro.fil
```

in the *autoexec.bat* file to the line :

```
C:\MOUSE1\SETSPEED /P4 /FC:\MOUSE1\mousepro.fil
```

In addition if the X/Helios Window Server is being used, (see the Section 2.4) then the following parameter *must* be commented out of the *host.con* file :

Server_windows

This parameters tells the Helios I/O server to implement its own */window* service, rather than allowing the X/Helios Window service to create windows.

2.3.4 Environment variables

The X server uses two environment variables, *HOME* and *NEWXRC*, both of which are covered in Section 2.3.1 describing the *newXrc* file.

2.3.5 Keyboard device driver

The X server comes supplied with two versions of the keyboard device driver, (called *keyboard.no* and *keyboard.yes*) in the directory */helios/lib*. The file *keyboard.yes* is a full implementation of an X keyboard device driver. If it is used the X server will take over complete control of the keyboard, and X clients will be able to receive keyboard events whenever any key is pressed. The file *keyboard.no*, however, is a fake. It causes the X server to receive its information about the keyboard by reading from its standard input, (just like any ordinary program). This means, for example, that X clients will never see function keys, cursor keys, control characters, etc. Instead they will just receive the normal alphabetic, numeric and punctuation keys. The purpose of this fake keyboard driver is to allow the X server to be run from a normal Helios session, (that is one that was started without starting up X), and to allow the Helios session to retain control of the keyboard, so for example ALT-F1 can still be used to switch between windows, whilst also allowing X clients to receive keyboard input whenever the user types at the X server. The X server actually loads a file called */helios/lib/keyboard.d* for its device driver, unless overridden by the X server resource file, so you should check which of the two keyboard drivers has been copied onto this file before starting the X server.

2.4 Starting X

There are two basic ways of starting an X world under Helios. The first is have the X server start up automatically when Helios is booted. The second method is to start up Helios first (without starting X) and then to run a shell script to start up the X world. The former method is much easier from a user's point of view, whereas the latter makes debugging much easier if things go wrong.

A word of warning about two of the X clients. The X terminal emulator *xterm* and the Tab Window Manager *twm* both produce status information whilst they are starting up. This is because both of these clients have long initialisation sequences, and the messages are there to confirm that they are still working. The messages are produced on *stderr* and can be removed by redirecting the output of the command to */null*, for example :

```
xterm >& /null
```

2.4.1 Starting X automatically (the X/Helios window server)

In order to do this the file *host.con* (normally found in */helios*) must be edited to comment out the line starting *Server_windows*. This will allow the X/Helios Window Server (the file */helios/lib/window*) to be run. This program will read in

the X server resource file */helios/etc/newXrc*. (Note that this is the only version that will be read, no command line or environment variables exist to change this.)

This file is examined to find the parameters *processor*, (which tells the window server upon which transputer it should run the X server), and *progname*, (which tells the window server the name and location of the object code for the X server). The window server will then start an X server, wait for it to start up, and then create a window on the X server, inside of which the normal Helios login program will be run. Users can then log in to Helios as normal.

In order for this start up method to work, the full keyboard device driver must be installed. To make sure, type:

```
cd /helios/lib
cp keyboard.yes keyboard.d
```

Note that the terminal emulator used by the window server is not Xterm, but its own built in emulator (which is much smaller!) You can still run Xterm, however, simply by invoking it from the command line once you have logged in.

2.4.2 Starting X manually (the startx script)

For more control over what is started and when, you may prefer to start up Helios first and then start up the X world by hand. The shell script */helios/startx* is provided as an example of how to do this. The script needs editing before it can be used. You should read the script, make any changes necessary and then try running it. If things do not work, try running the commands individually.

This method is amenable to running with either form of keyboard driver, although if the fake keyboard driver is used then once the X server has started you will still be able to type commands to the normal Helios environment.

2.4.3 Starting multiple X servers

You can run multiple X servers in the same Helios network. In order to do this you will need multiple graphics cards, (that is you cannot run two X servers on the same graphics card). Each X server is started up with a different display number, and then this number is used by X clients to distinguish which display(s) they wish to contact. For example if you have a Helios network with two graphics cards called `fred`, and `jim` then you could start up two X servers with the commands :

```
remote -d fred /helios/bin/Xhelios :0
remote -d jim  /helios/bin/Xhelios :1
```

These servers can then be contacted by X clients with display names of **Helios:0** (for the X server running on the processor called **fred**) and **Helios:1** (for the X server running on the processor called **jim**).

2.4.4 Not starting any X servers

If you have the Helios Ethernet package installed and your Helios network is connected up to the ethernet, then you can run X clients on the Helios network, displaying their output on X servers connected elsewhere on the ethernet. Thus you may not want to run an X server under Helios at all. Note that the reverse is also true. If you have one (or more) X servers running on the Helios network, you can run X clients elsewhere on the ethernet displaying their output on the Helios X servers. In fact you can have all possible combinations going on simultaneously!

2.5 Configuring X clients

X clients also have a range of ways in which they can be configured. This section describes what they are and how to use them.

2.5.1 Shell parameters

In order to be able to execute X clients you must have either set your current directory to the directory containing the clients, (using the **cd** command), or (more commonly) you must set your shell's search path to include the directory containing the X clients. The latter method can be done by editing the file *~/.cshrc*. If this file has a command along the lines of: **set path = (. /helios/local/bin /helios/bin)** then edit it to read: **set path = (. /helios/local/bin /helios/bin /helios/bin/X11)**

If the *~/.cshrc* file does not contain a line setting up the *path* variable then you should add a line that looks like the second version of the **set path** command above.

Whilst you are editing the file *~/.cshrc* you should also add the line :

```
setenv DISPLAY helios:0
```

This line tells X clients which X server they should contact, (unless the client has an option on its command line to override this).

Once you have finished making changes to the *~/.cshrc* file you should save it and then type :

```
source ~/.cshrc
```

2.5.2 The Xdefaults file

Most X clients can be configured in two ways. Firstly by options specified on their command lines, and secondly through a resource mechanism provided by the X library. This second method takes the form of a file called *Xdefaults* that should be located in the user's home directory. (Note that the following description is an abbreviated explanation of how the *Xdefaults* mechanism works. For a complete description you should consult one of the various books available on the X library. See Appendix A for a bibliography.)

The *Xdefaults* file contains lines in the following format :

```
<program name>*<program resource>:<value>
```

So, for example to set the foreground colour of the Helios terminal emulator, (the terminal emulator used by the X/Helios Window Server), the line should read :

```
term*foreground:  black
```

Spaces between the various components are optional, and are ignored when parsing the file. If no program name is given then the resource applies to all programs, hence :

```
*foreground:  blue
```

Sets the foreground colour of every X client (that actually uses a foreground resource) to be blue. If there are clashes, then the more specific case will prevail. Thus if both of the above lines were in the *Xdefaults* file, then terminal emulator would still have a black foreground, whilst everyone else would have a blue foreground. The order that the lines appear in the *Xdefaults* file is not important.

2.5.3 Host specific defaults

The file *Xdefaults* is read by the X library, not the X server. This means that the file must be accessible to the X client, wherever that may be running. In a networked environment the X client could well be running on a different processor, (and under a different filing system), from the X server. The X library thus provides a mechanism for specifying defaults that are specific to an X server, rather than an X client. What happens is that after opening the file *Xdefaults* the X library will try to open a file called *Xdef-<name-of-X-server's-host-machine>*. This file must also be in the home directory of the X client (not the X server.) Thus if the X client is running on a machine called *fred* and the X server is running on a machine called *jim*, then the X library will try to open the files *$HOME/Xdefaults*

and *$HOME/Xdef-jim*, both in *fred's* file space. If there are any clashes between the two default files then the host specific version will take priority.

Note Unix versions of the X library call these files *.Xdefaults* and *.Xdefaults-<host>* respectively. The names have been changed by the Helios version of the X library, to avoid problems when the MS-DOS filing system is being used.

2.5.4 Application specific defaults

X clients that have been built using the X toolkit, have a third potential source of default values. These are the files kept in the directory */helios/lib/X11/app-defaults*. These files provide resource values, (in exactly the same format as the *Xdefaults* file) on a per-application basis. Each file is named after its application, (with the first two letters capitalised), and contains resources specifically for use by that application. Thus if you have an application called *fred*, which uses the X toolkit, then during *fred's* start up, the X toolkit will attempt to load a file called */helios/lib/X11/app-defaults/FRed*. Entries in this file will take precedence over entries in either of the *Xdefaults* files. The point of this mechanism is that parsing the various defaults files is a slow and compute intensive operation, so reducing the amount of information that has to be parsed on a per-client basis is a definite win.

2.5.5 Window manager resource files

The X window managers avoid the use of the Xdefaults mechanism altogether by having their own resource files. These files have quite a different format, and are parsed by the window managers themselves. These files are fully documented in the manual pages of the window managers, so only a brief summary follows here. The files (called *uwmrc* for the *uwm* window manager, and *twmrc* for the *twm* window manager) basically work by defining what should happen when a mouse button is pressed somewhere on the screen. The resource files distinguish between which mouse button was pressed, which modifier keys on the keyboard were pressed at the time, (control, shift, etc), and where the mouse was on the screen, (over the background, over a window, over an icon, etc). The window managers provide a range of built in functions, (such as resizing a window, turning a window into an icon, etc), and they also provide the possibility of defining menus that should appear when button is pressed. These menus can then run further window manager commands, or they can provide strings to be executed by the Helios shell. The files are always kept in the user's home directory, and they are entirely text based, so that they can be easily edited.

2.5.6 Environment variables

The following environment variables affect clients using the X library :

DISPLAY

This is the (default) name and screen number of the X server to be contacted by the X client.

HOME

This is used to locate the user's home directory (when searching for the Xdefaults file).

USER

If *$HOME* does not exist then the X library tries to use *$USER* instead.

XENVIRONMENT

If this variable exists it is used as the path name of the host specific defaults file, rather than *$HOME/Xdef-<host>*.

RESOURCE_NAME

If this variable exists, it is used instead of the client's name when parsing the defaults databases.

XAUTHORITY

If this variable exists then it is used as the path name for the X Authorisation Database.

In addition if the client uses the X toolkit, the following extra environment variables may be used :

XUSERFILESEARCHPATH

If this variable exists then it specifies the path name of the application specific defaults database.

XAPPLRESDIR

If $XUSERFILESEARCHPATH does not exist, and this variable does exist, then it is used as the pathname of the directory to be searched for the application specific defaults.

LANG

If it exists this variable is used to set up the language field of X toolkit's *XtPerDisplay* structure.

XFILESEARCHPATH

If it exists, then this variable is used as a prefix to all the filenames located by the X toolkit.

2.6 Problems starting X

If you have problems starting or configuring your X world then this section may help. The section tries to identify common problems and their solutions, but if you are still stuck then contact your distributor. If you have a support contract with DSL, you can also contact them for help.

- *Exec Format Error*

  This usually means one of three things. Either the command you just tried to run is corrupt, (maybe because you ran out of disc space when copying it onto your hard disc, or because it was copied in ASCII mode and not BINARY mode), in which case you should check the size of the program against the size of the original and if necessary copy it again. Alternatively there may not be enough memory left on the processor upon which you want to run the command. (You can check the amount of memory left using the *map* or *free* commands.) Try running the program on a different processor, or removing a few programs that are already running on that processor. Another case when this error message can occur is when the program needs a resident library in order to run, and that library is not present in */helios/lib*. You can check which libraries the program is trying to load by using the CTRL_SHIFT-L key sequence and watching the messages appearing on the Helios Server window.

- *X Server hangs up loading mouse driver*

  This usually occurs when *Xsupport* has not been enabled in the *host.con* file. Locate this file (it is usually in the directory */helios*) and edit it. Make sure that the line containing the word *Xsupport* does not have any hash signs (#) at the start.

- *Command not found*

  Make sure you have the directory containing the command in your search path. All the X clients are located in the directory */helios/bin/X11*. You can add this directory to your search path with the commands:

```
set path = ($path /helios/bin/X11)
rehash
```

(You can put these commands in your *cshrc* file so that they are executed every time you start up a Helios shell.)

Another possible problem is that if the command has more than eight characters in its name then the Helios shell may not be able to find it. (See the note in Section 1.5.)

- *Xterm does not start*

  The X terminal emulator program (*xterm*) needs the Helios pseudo terminal server in order for it to work. This server is normally supplied with the Helios Ethernet package, but if you have not purchased this you can use the backup versions supplied with the X package. First check to see if you have the files */helios/lib/ttyserv* and */helios/lib/tpseudo.d*. If they are missing then type:

```
cd /helios/lib
cp ttyserv.bak ttyserv
cp tpseudo.dbk tpseudo.d
```

  If the Xterm program is working it should display the following messages whilst starting up :

```
Xterm: starting
Xterm: initialising X toolkit
Xterm: initialised X toolkit
Xterm: getting resources
Xterm: got resources
Xterm: getting a terminal
Xterm: spawning a child
Xterm: child spawned
Xterm: Ready
```

- *Lots of carriage return characters* (CTRL-M) *in a text file*

  This can happen as a consequence of the way MS-DOS stores text files. The Helios utility *xlatecr* can be used to remove these unwanted characters. Simply type :

```
xlatecr <name-of-bad-file>
```

2.7 Making your own hardware device driver

If you have a graphics card that is not currently supported by the Helios X package you will need to create your own device driver for it. This is not a difficult task.

In order to complete the task you will need :

- The *Helios C compiler, Helios Assembler, Helios Compiler driver,* and possibly the *Helios Assembler Macro Pre-Processor.*
- Knowledge of how to drive the graphics card, (preferably in C).
- The example device driver sources (supplied).

The directory */helios/local/xsrc/devices* contains a skeleton form of an X graphics hardware device driver (called *generic.c*) and an example of a working device driver for the INMOS G300 graphics chip (called *newG300.c*). What you need to do is as follows :

1. cd to */helios/local/xsrc/devices*
2. Copy *generic.c* or *newG300.c* to *<your-device-name>.c*
3. Edit *<your-device-name>.c* to implement the three functions required, (see below).
4. Edit *makefile* and change the variable *PROGRAM* to match the name of your device driver.
5. Compile your device driver object code, (type **make <your-device-name>.o**) and fix any simple bugs, typos, etc. Note the header files used by the code are in the parent of the current directory.
6. Link your device driver, (type **make**). If the linker complains about

   ```
   Symbol '.<xyz>' undefined - set to zero
   ```

   where <xyz> is a system function call, it means that <xyz> is missing from the file *device.a*. Edit *device.a* and create a stub for <xyz> (in exactly the same as there is already a stub for *Malloc* and *Open*). Re-make *device.o* (for which you will need the *Helios Assembler Macro Pre-Processor*) and then relink your device driver.
7. Copy *newXrc* to *newXrc.G300*
8. Edit *newXrc* and replace all of the G300 specific parameters with comments and default values for the parameters used by your device driver. Also change the value of the *hardware_device* parameter to match the name of your device driver.

9. Run the program *harness* (located in the current directory) which will attempt to load your device driver and use it to initialise the graphics hardware. The program will try to identify any device specific parameters and allow you to change them dynamically. When it is working the program fills the screen with vertical columns of solid colour (ranging from black to through to bright red, then black through to bright green, then black through to blue, and then repeating) with a solid white border (4 pixels wide) all the way around the outside.

10. When everything is working correctly copy the file *newXrc* to */helios/etc* or wherever it ought to be installed. Copy your device driver to */helios/lib* and try starting the X server.

11. If your keyboard and / or mouse are not attached to the Helios I/O server then you will need some way of attaching them to the X server. You can either attempt to implement the */keyboard* and */mouse* services as provided by the IO server, or you can create your own keyboard and mouse device drivers based on the files *keyboard.c* and *mouse.c*. If you do this be sure to edit the *newXrc* file to set the parameters *keyboard_device* and *mouse_device*.

The graphics hardware device driver provides three functions to the X server, all of which are documented here :

DevClose

This function is called by the X server when it wishes the graphics hardware to be shutdown. Currently this function will not be called, but it should be supported to enable compatibility with future versions of the X server.

DevOpen

This function is called when the X server wishes the graphics hardware to be initialised. The function is passed two parameters, *pdev* (which can be ignored) and *pinfo*. The second parameter can be used to access the contents of the X server's resource file in order to obtain the values of parameters used by your device driver. This function must perform two actions. It must initialise the graphics hardware, (apart from the colour lookup table(s)), taking note of any parameters set by the user in the *newXrc* buffer, and it must allocate and fill out a *Hardware_Device* structure. This structure is documented in the file */helios/local/xsrc/hardware.h* and contains fields that tell the X server about hardware dependent information. The function must return a pointer to this filled out structure.

DevOperate

This function is called whenever the X server wants to install values into the colour lookup table(s) provided by the graphics hardware. (The X server will never want to read these values.) The X server will try to batch up multiple requests to change entries in the CLUT, so that they can all be performed together. The format of the request is defined in the file */helios/local/xsrc/hardware.h*.

Note, all device drivers must be compiled with a special option to the *Helios C Compiler* (*-m*) which means that **static variables are not allowed.** If you need to pass information between the various functions you must use the *pprivate* field of the *Hardware_Device* structure.

3

Programming With X Under Helios

This chapter describes how to compile X clients under Helios. It covers the problems you are most likely to encountered, and describes one of the example programs provided.

3.1 How to compile X clients

X clients should be compiled by using the *Helios Compiler Driver*. All of the programs should be linked with the resident X library, and if necessary the scanned X library. A typical command line will look like this :

```
c client.c -O -lX11 -lX -o client
```

If the client uses the X toolkit and / or the Athena Widget library, then these libraries should be linked in. The command line for a typical large X client will look like :

```
c *.o -O -lXaw -lXt -lXext -lXmu -lX11 -lX -o client
```

The *Helios Compiler Driver* has a large range of options, and is fully documented in the *Helios Encyclopaedia*. The program emulates most of the features of the Unix *cc* command, (including the features specified in the POSIX standard). X clients, can, if you wish, be compiled by hand, (that is directly invoking the *Helios C Compiler* and *Helios Assembler*), but this is a tedious and error prone operation.

3.2 Compiling Unix based X clients

Helios, with its support of the POSIX standard, and its BSD emulation library, is now very close in its programming interface, to Unix. The major problems that you are likely to come across whilst porting X clients to Helios are as follows :

Earlier X releases

This package implements X11 release 4. X clients written for earlier releases, or for earlier versions of the X protocol, may encounter some problems. In particular, release 4 is more strict about adherence to the X11 protocol, and clients which assume, for example, that all the bits in a pixel can be specified as `'~0'` will not necessarily work. For these clients, either fixing the incorrect assumptions, or enabling the bug compatibility mode of the X server (by typing **`xset bc`**) is the best answer.

ANSI C

The *Helios C Compiler* is a strict ANSI C compiler. It does not have a K&R option, and it tends to produce lots of warnings even from the simplest program. A lot of these warnings can be suppressed by the *-wA* to the *Helios Compiler Driver*, and others can be safely ignored. Error messages starting with the words *Serious Error*, however, must not be ignored as they will prevent the code from compiling.

Stack

Helios does not operate in a virtual memory environment, and so the stack of executing programs cannot be extended whilst the program is running. The *Helios Compiler Driver* automatically gives X clients a stack size of 10 kilobytes, but for very big X clients, (or highly recursive ones), this may not be enough. The stack size can be changed at compile time by the *-s <number>* option to the *Helios Compiler Driver*, or by using the

```
objed -i -s <number> <program-name>
```

command after the program has been built. If the program does run out of stack, then either it will simply crash, (having walked over the memory used by its vector stack), or if you are lucky, it will give you a message about stack overflow and then terminate.

Memory

Another consequence of the lack of virtual memory is that programs have complete access to the entire address space, and can easily corrupt the operating system or any other running program. Programs should be very

careful to avoid writing through NULL pointers, using memory after it has been freed, for example :

```
free( node ); node = node->next;
```

or freeing static data :

```
free( "Hello World" );
```

fork()

Helios does not support the *fork()* system call. It does support *vfork()* (with the provision that the *vfork()*, and subsequent *exec()* calls both take place inside the same function). If your program cannot exist without the *fork()* you should investigate the *Fork()* system call, but be prepared for a long hard battle.

BSD Emulation

If you wish to use the BSD emulation features of Helios then your C programs must be compiled with the *-D_BSD* set and linked with the BSD emulation library, (*-lbsd*).

Variable Arguments

The way variable arguments are handled in ANSI C differs slightly from K&R C. The follow shows an example of a function written in both K&R C and ANSI C :

```
/* K&R style */

#include <varargs.h>

void
debug( format, va_alist )
  const char * format;
  va_dcl
{
  va_list args;


  va_start( args );

  vfprintf( stderr, format, args );
```

```
/* ANSI style */

#include <stdarg.h>

void
debug( const char * format,


{
  va_list args;


  va_start( args, format );

  vfprintf( stderr, format, ar
```

```
        fputc( '\n', stderr );
        va_end( args );
        return;
    } /* debug */
```

```
        fputc( '\n', stderr );
        va_end( args );
        return;
    } /* debug */
```

Shared Memory

Helios does not have a shared memory model, but since all clients executing on the same processor have full access to the all the memory on that processor, a shared memory scheme between cooperating clients can easily be implemented.

3.3 Compiling Helios based X clients

There are several features of the Helios environment that can be used advantageously by X clients, this section mentions a few.

Fast RAM

Although the X server takes over all of the available Fast RAM on the processor upon which it is running, if the X client is running on another processor, then it could have some Fast RAM available to it. The Helios functions *Accelerate()* and *AccelerateCode()*, (both of which are documented in the *Helios Encyclopaedia*) can be used to access this resource.

Inline Code

The *Helios C Compiler* supports inline transputer assembler via the *_operate()* macro. This is used extensively by the X server, especially when it performs 2D block copies. The header file */helios/include/bytblt.h* demonstrates exactly how this macro works.

Lightweight Threads

Helios supports lightweight (that is low administration overheads) threads via the *Fork()* system call. Helios also has full support for semaphores, linked lists, and inter-process communication.

Distributed Processing

Helios is a distributed operating system, with full support for distributed and parallel processing. The CDL language can be used to specify the connectivity of tasks, and the tasks can communicate via ordinary reads and writes on their streams.

3.4 Example program

The directory */helios/examples* has four example programs in it. Three of them, *hello.c*, *setkeys.c*, and *xlights.c* are (relatively) simple examples of X clients. The fourth, *grab.c*, discussed here, is an example of how a non-X client could take over control of the screen, do something (for example display images loaded off disc), and then return control to the X server. The code can be compiled with the command :

```
c grab.c -DEXAMPLE_CODE -FA -O -lX11 -lX -o grab
```

and then run, (on the processor with the graphics hardware) with the command :

```
grab
```

Note, if your graphics card does not have a G300 chip displaying on a monitor that is 1024 pixels wide by 768 pixels tall, then you will need to edit *grab.c* before compiling it, and change the *#define* constants near line 225.

The program, when run, demonstrates taking over the X server, displaying a vaguely interesting pattern by writing directly into the video RAM, and then returning control to the X server, (whereupon the screen is repainted).

4

Xlib Summary

This chapter provides a quick reminder of the functions and macros in the X Windows System Library, *Xlib*. It does not attempt to give a full description of Xlib. If you require further information, you should consult one of the X reference books listed in the bibliography at the end of this manual (Appendix A).

To help you find and use the most suitable function or macro for a particular purpose, the information described in this chapter has been arranged in several sections. The first section lists each function or macro by subject; this is useful for determining which functions are available for achieving the desired effect. Subsequent sections give an alphabetical list of macros and functions, and provide information about their purpose and the calling syntax. Notice that a quick and easy way to check whether a particular name refers to a function or a macro is to look at the first letter of the name: Xlib functions start with an X, whereas macros do not. Most macros also have function equivalents; if you prefer to use a function instead of the macro, prefix the macro name by *X*.

As in the rest of this manual, the names of the functions and macros have not been changed from the originals; the spelling used in the descriptions follows British rather than American English (that is, *color* is spelt as *colour*).

4.1 Listing by Subject

4.1.1 Buffers

XStoreBuffer

XStoreBytes
XFetchBuffer
XFetchBytes
XRotateBuffers

4.1.2 Client Connections

XKillClient
XSetCloseDownMode

4.1.3 Colorcells

BlackPixel
WhitePixel
XAllocColor
XAllocColorCells
XAllocColorPlanes
XAllocNamedColor
XFreeColors
XLookupColor
XParseColor
XQueryColor
XQueryColors
XStoreColor
XStoreColors
XStoreNamedColor

4.1.4 Colormaps

DefaultColormap
DefaultColormapOfScreenp
DisplayCells
XCopyColormapAndFree
XCreateColormap
XFreeColormap
XGetStandardColormap
XInstallColormap
XListInstalledColormaps
XSetStandardColormap
XSetWindowColormap

XUninstallColormap

4.1.5 Context Manager

XDeleteContext
XFindContext
XSaveContext
XUniqueContext

4.1.6 Cursors

XCreateFontCursor
XCreateGlyphCursor
XCreatePixmapCursor
XDefineCursor
XFreeCursor
XRecolorCursor
XQueryBestCursor
XQueryBestSize
XUndefineCursor

4.1.7 Display Specifications

DefaultColormap
DefaultDepth
DefaultGC
DefaultScreenDefault
VisualDisplayCells
DisplayHeightMM
DisplayPlanes
DisplayString
DisplayWidth
DisplayWidthMM
RootWindow
ScreenCount

4.1.8 Drawing Primitives

XCopyArea
XCopyPlane

XClearArea
XClearWindow
XDraw
XDrawArc
XDrawArcs
XDrawFilled
XDrawLine
XDrawLines
XDrawPoint
XDrawPoints
XDrawRectangle
XDrawRectangles
XDrawSegments
XFillArc
XFillArcs
XFillPolygon
XFillRectangle
XFillRectangles

4.1.9 Errors

XDisplayName
XGetErrorDatabaseText
XGetErrorText
XSetAfterFunction
XSetErrorHandler
XSetIOErrorHandler
XSynchronize

4.1.10 Events

QLength
XAllowEvents
XCheckIfEvent
XCheckMaskEvent
XCheckTypedEvent
XCheckTypedWindowEvent
XCheckWindowEvent
XEventsQueued
XGetInputFocus

XGetMotionEvents
XIfEvent
XMaskEvent
XNextEvent
XPeekEvent
XPeekIfEvent
XPending
XPutBackEvent
XSelectInput
XSendEvent
XSetInputFocus
XSynchronize
XWindowEvent

4.1.11 Extensions

XFreeExtensionList
XListExtensions
XQueryExtension

4.1.12 Fonts

XCreateFontCursor
XFreeFont
XFreeFontInfo
XFreeFontNames
XFreeFontPath
XGetFontPath
XGetFontProperty
XListFonts
XListFontsWithInfo
XLoadFont
XQueryFont
XSetFont
XSetFontPath
XUnloadPath

4.1.13 Grabbing

XChangeActivePointerGrab

XGrabButton
XGrabKey
XGrabKeyboard
XGrabPointer
XGrabServer
XUngrabButton
XUngrabKey
XUngrabKeyoard
XUngrabPointer
XUngrabServer

4.1.14 Graphics Context (GC)

DefaultGC
XChangeGC
XCopyGC
XCreateGC
XFreeGC
XGCContextFromGC
XSetArcMode
XSetBackground
XSetClipMask
XSetClipOrigin
XSetClipRectangles
XSetDashes
XSetFillRule
XSetFillStyle
XSetForeground
XSetFunction
XSetGraphicsExposures
XSetLineAttributes
XSetPlaneMask
XSetRegion
XSetState
XSetStipple
XSetSubwindowMode
XSetTile
XSetTSOrigin

4.1.15 Host Access

XAddHost
XAddHosts
XDisableAccessControl
XEnableAccessControl
XListHosts
XRemoveHost
XRemoveHosts
XSetAccessControl

4.1.16 Housekeeping

DefaultScreen
XCloseDisplay
XFree
XOpenDisplay
XNoOp

4.1.17 Images

ImageByteOrder
XAddPixel
XCreateImage
XDestroyImage
XGetImage
XGetPixel
XGetSubImage
XPutImage
XPutPixel
XSubImage

4.1.18 Keyboard

XChangeKeyboardMapping
XDeleteModifiermapEntry
XFreeModifiermap
XGetKeyboardMapping
XGetModifierMapping
XInsertModifiermapEntry
XKeycodeToKeysym

XKeysymToKeycode
XKeysymToString
XLookupKeysym
XLookupString
XNewModifiermap
XQueryKeymap
XRebindKeysym
XRefreshKeyboardMapping
XSetModifierMapping
XStringToKeysym

4.1.19 Macros (Display)

AllPlanes
BlackPixel
BlackPixelOfScreen
CellsOfScreen
ConnectionNumber
DefaultColormap
DefaultColormapOfScreen
DefaultDepth
DefaultDepthOfScreen
DefaultGC
DefaultGCOfScreen
DefaultRootWindow
DefaultScreen
DefaultScreenOfDisplay
DefaultVisual
DefaultVisualOfScreen
DisplayCells
DisplayHeightMM
DisplayOfScreen
DisplayPlanes
DisplayString
DisplayWidth
DisplayWidthMM
DoesBackingStore
DoesSaveUnders
EventMaskOfScreen
HeightOfScreen

HeightMMOfScreen
LastKnownRequestProcessed
MaxCmapsOfScreen
MinCmapsOfScreen
NextRequest
PlanesOfScreen
ProtocolRevision
ProtocolVersion
QLength
RootWindow
RootWindowOfScreen
ScreenCount
ScreenOfDisplay
ServerVendor
VendorRelease
WhitePixel
WhitePixelOfScreen
WidthOfScreen
WidthMMOfScreen

4.1.20 Macros (Image Format)

BitmapBitOrder
BitmapPad
BitmapUnit
ImageByteOrder

4.1.21 Mapping

XChangeKeyboardMapping
XDeleteModifiermapEntry
XFreeModifiermap
XGetKeyboardMapping
XGetModifierMapping
XGetPointerMapping
XInsertModifiermapEntry
XMapRaised
XMapSubwindows
XMapWindow
XNewModifiermapXQueryKeymap

XRefreshKeyboardMapping
XSetModifierMapping
XSetPointerMapping
XUnmapSubwindows
XUnmapWindow

4.1.22 Output Buffer

XFlush
XSync

4.1.23 Pointers

XChangeActivePointerGrab
XChangePointerControl
XGetPointerControl
XGetPointerMapping
XGrabPointer
XQueryPointer
XSetPointerMapping
XUngrabPointer
XWarpPointer

4.1.24 Properties

XChangeProperty
XDeleteProperty
XGetAtomName
XGetFontProperty
XGetWindowProperty
XInternAtom
XListProperties
XRotateWindowProperties
XSetStandardProperties

4.1.25 Regions

XClipBox
XCreateRegion
XDestroyRegion

XEmptyRegion
XEqualRegion
XIntersectRegion
XOffsetRegion
XPointInRegion
XPolygonRegion
XRectInRegion
XSetRegion
XShrinkRegion
XSubtractRegion
XUnionRectWithRegion
XUnionRegion
XXorRegion

4.1.26 Resource manager and database

Xpermalloc
XrmGetFileDatabase
XrmInitialize
XrmGetResource
XrmGetStringDatabase
XrmMergeDatabases
XrmParseCommand
XrmPutFileDatabase
XrmPutLineResource
XrmPutResource
XrmPutStringResource
XrmQGetResource
XrmQGetSearchList
XrmQGetSearchResource
XrmQPutResource
XrmQPutStringResource
XrmQuarkToString
XrmStringToBindingQuarkList
XrmStringToQuarkList
XrmStringToQuark
XrmUniqueQuark

4.1.27 Save set

XAddToSaveSet
XChangeSaveSet
XRemoveFromSaveSet

4.1.28 Screen saver

XActiveScreenSaver
XForceScreenSaver
XGetScreenSaver
XResetScreenSaver
XSetScreenSaver

4.1.29 Selections

XConvertSelection
XGetSelectionOwner
XSetSelectionOwner

4.1.30 Standard geometry

XGeometry
XParseGeometry
XTranslateCoordinates

4.1.31 Text

XDrawImageString
XDrawImageString16
XDrawString
XDrawString16
XDrawText
XDrawText16
XQueryTextExtents
XQueryTextExtents16
XTextExtents
XTextExtents16
XTextWidth
XTextWidth16

4.1.32 Tile, pixmap, stipple and bitmap

XCreateBitmapFromData
XCreatePixmap
XCreatePixmapFromBitmapData
XFreePixmap
XQueryBestSize
XQueryBestStipple
XQueryBestTile
XReadBitmapFile
XSetTile
XSetWindowBorderPixmap
XSetWindowBackgroundPixmap
XWriteBitmapFile

4.1.33 User preferences

XAutoRepeatOff
XAutoRepeatOn
XBell
XChangeKeyboardControl
XGetDefault
XGetKeyboardControl
XGetPointerControl

4.1.34 Visuals

DefaultVisual
XGetVisualInfo
XMatchVisualInfo

4.1.35 Window attributes

XChangeWindowAttributes
XDefineCursor
XGetGeometry
XGetWindowAttributes
XSetWindowBackground
XSetWindowBackgroundPixmap
XSetWindowBorder
XSetWindowBorderPixmap

XSetWindowColormap
XSelectInput

4.1.36 Window configuration

XConfigureWindow
XGetGeometry
XMoveResizeWindow
XMoveWindow
XResizeWindow
XRestackWindow
XSetWindowBorderWidth

4.1.37 Window existence

XCreateSimpleWindow
XCreateWindow
XDestroySubwindows
XDestroyWindow

4.1.38 Window manager hints

XFetchName
XGetClassHint
XGetIconName
XGetIconSizes
XGetNormalHints
XGetSizeHints
XGetTransientForHint
XGetWMHints
XGetZoomHints
XSetClassHint
XSetCommand
XSetIconName
XSetIconSizes
XSetNormalHints
XSetSizeHints
XSetTransientForHint
XSetWMHints
XSetZoomHints

XStoreName

4.1.39 Window manipulation

XCirculateSubwindows
XCirculateSubwindowsDown
XCirculateSubwindowsUp
XConfigureWindow
XLowerWindow
XMoveWindow
XMoveResizeWindow
XQueryTree
XRaiseWindow
XReparentWindow
XResizeWindow
XRestackWindows
XSetWindowBorderWidth

4.1.40 Window mapping

XMapRaised
XMapSubwindows
XMapWindow
XUnmapSubwindows
XUnmapWindow

4.2 Macros

All of these macros are also available as functions, with an **X** prepended to their name. Thus the function version of the *RootWindow()* macro is called *XRootWindow()*.

AllPlanes
Returns a value, with all bits set, which is suitable for use as a 'plane' argument to a procedure. The returned value is compatible with the C type unsigned long.

unsigned long
AllPlanes

BitmapBitOrder
Return value is an integer; equivalent to the macro's MSBFirst or LSBFirst.

int
*BitmapBitOrder(Display * display)*

BitmapPad
Each scanline must be padded to a multiple of bits returned by this function.

int
*BitmapPad(Display * display)*

BitmapUnit
Returns the size of the bitmap's scanline unit in bits. The scanline is calculated in multiples of this value; it is always less than the bitmap scanline pad.

int
*BitmapUnit(Display * display)*

BlackPixel
Returns the black pixel value for the specified screen.

unsigned long
*BlackPixel(Display * display, int screen_number)*

BlackPixelOfScreen
Returns the black pixel value for the specified screen.

unsigned long
*BlackPixelOfScreen(Screen * screen)*

CellsOfScreen
Returns the number of colour map cells of the specified screen.

int
*CellsOfScreen(Screen * screen)*

ConnectionNumber
Returns a file descriptor for the display.

int
*ConnectionNumber(Display * display)*

DefaultColormap
Returns the default colour map ID for allocation on the screen; most routine allocations of colour should be made out of this colour map.

Colormap
*DefaultColormap(Display * display, Screen * screen)*

DefaultColormapOfScreen
Returns the default colour map of the specified screen.

Colormap
*DefaultColormapOfScreen(Screen * screen)*

DefaultDepth
Returns the default depth (planes) of the default root window, for the specified screen.

int
*DefaultDepth(Display * display, int screen_number)*

DefaultDepthOfScreen
Returns default depth of specified screen.

int
*DefaultDepthOfScreen(Screen * screen)*

DefaultGC
Returns the default graphics context for the root window of the specified screen. This GC is created for the convenience of simple applications and contains the default GC components with the foreground and background pixel values initialised to black and white.

GC
*DefaultGC(Display * display, int screen_number)*

DefaultGCOfScreen
Returns the default graphics context of the specified screen.

GC
*DefaultGCOfScreen(Screen * screen)*

DefaultRootWindow
Returns the root window for the specified screen.

Window
*DefaultRootWindow(Screen * screen)*

DefaultScreen

Returns the default screen number referenced in XOpenDisplay(). This macro should be used to retrieve the screen number in applications that will use only a single screen.

int
*DefaultScreen(Display * display)*

DefaultScreenOfDisplay

Returns the default screen of the specified display.

*Screen **
*DefaultScreenOfDisplay(Display * display)*

DefaultVisual

Returns the default visual type for the specified screen.

*Visual **
*DefaultVisual(Display * display, int screen_number)*

DefaultVisualOfScreen

Returns the default visual of the specified screen.

*Visual **
*DefaultVisualOfScreen(Screen * screen)*

DisplayCells

Returns the number of entries in the default colour map.

int
*DisplayCells(Display * display, int screen_number)*

DisplayHeightMM

Returns the height of the specified screen, in millimeters.

int
*DisplayHeightMM(Display * display, int screen_number)*

DisplayOfScreen

Returns the display of the specified screen.

*Display **
*DisplayOfScreen(Screen * screen)*

DisplayPlanes
Returns the depth of the root window in the specified screen.

int
*DisplayPlanes(Display * display, int screen_number)*

DisplayString
Returns the string that was passed to **XOpenDisplay()** when the current display was opened. If the display was opened with a NULL string, this macro returns the value of the environment variable, DISPLAY. This macro is useful when a child process is created, and wants to open a new connection to the same display as the parent process.

*char **
*DisplayString(Display * display)*

DisplayWidth
Returns the width of the screen in pixels.

int
*DisplayWidth(Display * display, int screen_number)*

DisplayWidthMM
Returns the width of the specified screen in millimeters.

int
*DisplayWidthMM(Display * display, int screen_number)*

DoesBackingStore
Returns a value which indicates if the screen supports backing stores; the returned value may be WhenMapped, NotUseful, or Always.

int
*DoesBackingStore(Screen * screen)*

DoesSaveUnders
Returns a boolean flag indicating whether the screen supports save unders; TRUE indicates that is does.

Bool
*DoesSaveUnders(Screen * screen)*

EventMaskOfScreen
Returns the initial root event mask for the specified screen.

long
*EventMaskOfScreen(Screen * screen)*

HeightMMOfScreen
Returns the height of the specified screen in millimeters.

int
*HeightMMOfScreen(Screen * screen)*

HeightOfScreen
Returns the height of the screen (in pixels).

int
*HeightOfScreen(Screen * screen)*

ImageByteOrder
Specifies the required byte order for images for each scanline unit in XYFormat or for each pixel value in ZFormat. This macro returns LSBFirst or MSBFirst.

int
*ImageByteOrder(Display * display)*

LastKnownRequestProcessed
Returns the serial number of the request which was known by Xlib to have been processed by the X Server. This number is automatically set by Xlib when replies, events, and errors are received.

int
*LastKnownRequestProcessed(Display * display)*

MaxCmapsOfScreen
Returns the maximum number of colour maps supported by the screen.

int
*MaxCmapsOfScreen(Screen * screen)*

MinCmapsOfScreen
Returns the minimum number of colour maps supported by the screen.

int
*MinCmapsOfScreen(Screen * screen)*

NextRequest
Returns the full serial number that is to be used for the next request issued to the X Server. Serial numbers are maintained separately for each display connection.

int
*NextRequest(Display * display)*

PlanesOfScreen
Returns the number of planes in the specified screen.

int
*PlanesOfScreen(Screen * screen)*

ProtocolRevision
Returns the minor protocol revision number of the X Server (4).

int
*ProtocolRevision(Display * display)*

ProtocolVersion
Returns the major version number (11) of the X protocol which is associated with the connected display.

int
*ProtocolVersion(Display * display)*

QLength
Returns the length of the event queue for the connected display. Note that there may be events that have not been read into the queue yet.

int
*QLength(Display * display)*

RootWindow
Returns the root window; this macro is used with functions that take a parent window as an argument.

Window
*RootWindow(Display * display, int screen_number)*

RootWindowOfScreen
Returns the root window of the specified screen.

Window
*RootWindowOfScreen(Screen * screen)*

ScreenCount
Returns the number of available screens.

int
*ScreenCount(Display * display)*

ScreenOfDisplay
Returns a pointer to the screen of the specified display.

*Screen **
*ScreenOfDisplay(Display * display, int screen_number)*

ServerVendor
Returns a pointer to a null terminated string which identifies the owner of this X server implementation.

*char **
*ServerVendor(Display * display)*

VendorRelease
Returns the vendor's release number for the X server.

int
*VendorRelease(Display * display)*

WhitePixel
Returns the white pixel value for the specified screen.

unsigned long
*WhitePixel(Display * display, int screen_number)*

WhitePixelOfScreen
Returns the white pixel value of the specified screen.

unsigned long
*WhitePixelOfScreen(Screen * screen)*

WidthMMOfScreen
Returns the width of the specified screen (in millimeters).

int
*WidthMMOfScreen(Screen * screen)*

WidthOfScreen
Returns the width of the screen (in pixels).

int
*WidthOfScreen(Screen * screen)*

4.3 Functions

XActivateScreenSaver

Turns on the screen saver to avoid burning out phosphors when the display is left on but unused.

void
*XActivateScreenSaver(Display * display)*

XAddExtension

Obtains an extension code for use with an extension to the X library.

*ExtCodes **
*XAddExtension(Display * display)*

XAddHost

Adds the specified host to the access control list.

void
*XAddHost(Display * display, XHostAddress * host)*

XAddHosts

Adds a number of specified hosts to the access control list.

void
*XAddHosts(Display * display, XHostAddress * host, int num_hosts)*

XAddPixel

Adds a constant value to each pixel value in an image.

int
*XAddPixel(XImage * ximage, unsigned long value)*

XAddToExtensionList

Adds an extension data structure to the extsntion data structure list.

void
*XAddToExtensionList(struct _XExtData ** structure, XExtData * ext_data)*

XAddToSaveSet

Adds the children of the specified window to the client's save set.

void
*XAddToSaveSet(Display * display, Window w)*

XAllocColor

Allocates the read-only colormap cell with the closest RGB values available and then returns the actual RGB values used.

Status
*XAllocColor(Display * display, Colormap cmap, XColor * colorcell_def)*

XAllocColorCells

Allocates read/write colormap cells in a read/write colormap.

Status
*XAllocColorCells(Display * display, Colormap cmap, Bool contig,*
*unsigned long * plane_masks_return, unsigned int nplanes,*
*unsigned long * pixels_return, unsigned int npixels)*

XAllocColorPlanes

Allocates read/write colour planes.

Status
*XAllocColorPlanes(Display * display, Colormap cmap, Bool contig,*
*unsigned long * pixels_return, int ncolors, int nreds, int ngreens,*
*int nblues, unsigned long * rmask_return,*
*unsigned long * gmask_return, unsigned long * bmask_return)*

XAllocNamedColor

Finds the RGB values for *colorname* from the colour database and then allocates a read-only colorcell with the closest colour available (see **AllocColor**).

Status
*XAllocNamedColor(Display * display, Colormap cmap,*
*const char * colourname, XColor * colorcell_def,*
*XColor * rgb_db_def)*

XAllocClassHint

Returns allocated space suitable for use as an XClassHint structure.

*XClassHint **
XAllocClassHint(void)

XAllocIconSize

Returns allocated space suitable for use as an XIconSize structure.

*XIconSize **
XAllocIconSize(void)

XAllocSizeHints
Returns allocated space suitable for use as an XSizeHints structure.

*XSizeHints **
XAllocWMHints(void)

XAllocStandardColormap
Returns allocated space suitable for use as an XStandardColormap structure.

*XStandardColormap **
XAllocStandardColormap(void)

XAllocWMHints
Returns allocated space suitable for use as a XWMHints structure.

*XWMHints **
XAllocWMHints(void)

XAllowEvents
Controls keyboard and pointer events when they are grabbed.

void
*XAllowEvents(Display * display, int event_mode, Time time)*

XAutoRepeatOff
Stops keyboard auto-repeat so that multiple press/release events will not occur when a key is held down.

void
*XAutoRepeatOff(Display * display)*

XAutoRepeatOn
Restarts keyboard auto-repeat (opposite of **XAutoRepeatOff**).

void
*XAutoRepeatOn(Display * display)*

XBell
Rings the keyboard bell at the specified volume.

void
*XBell(Display * display, int percent)*

XChangeActivePointerGrab
Alters the specified dynamic parameters of an active pointer grab.

void
*XChangeActivePointerGrab(Display * display, unsigned int event_mask,*
Cursor cursor, Time time)

XChangeGC

Changes some or all of the components of the specified graphics context.

void
*XChangeGC(Display * display GC gc, unsigned long valuemask,*
*XGCvalues * values)*

XChangeKeyboardControl

Sets the user keyboard preferences.

void
*XChangeKeyboardControl(Display * display, unsigned long value_mask,*
*XKeyboardControl * values)*

XChangeKeyboardMapping

Defines the key symbols allocated for the specified keycodes.

void
*XChangeKeyboardMapping(Display * display, int first_keycode,*
*int keysyms_per_keycode, KeySym * keysyms, int num_keycodes)*

XChangePointerControl

Sets the pointer preferences to define how the pointing device is to move.

void
*XChangePointerControl(Display * display, Bool do_accel,*
Bool do_threshold, int accel_numerator, int accel_denominator,
int threshold)

XChangeProperty

Changes the specified property associated with the given window.

void
*XChangeProperty(Display * display, Window w, Atom property, Atom type,*
*int format, int mode, const unsigned char * data, int nelements)*

XChangeSaveSet

Adds or removes a given window's children from the client's save set.

void
*XChangeSaveSet(Display * display, Window w, int change_mode)*

XChangeWindowAttributes
Changes some or all of the attributes of a given window.

void
*XChangeWindowAttributes(Display * display, Window w,*
*unsigned long valuemask, XSetWindowAttributes * attributes)*

XCheckIfEvent
Looks in the event queue for a matching event.

void
*XCheckIfEvent(Display * display, XEvent * event,*
Bool (predicate)(Display *, XEvent *, char *), char * args)*

XCheckMaskEvent
Looks in the event queue and removes the next event to match the given event mask.

void
*XCheckMaskEvent(Display * display, unsigned long mask_event,*
*XEvent * event)*

XCheckTypedEvent
Looks in the event queue and returns the next event to match the given event type.

void
*XCheckTypedEvent(Display * display, int event_type, XEvent * report)*

XCheckTypedWindowEvent
Looks in the event queue and returns the next event that matches the given window and event type.

void
*XCheckTypedWindowEvent(Display * display, Window w, int event_type,*
*XEvent * report)*

XCheckWindowEvent
Looks in the event queue and removes the next event to match the given window and event type.

void
*XCheckWindowEvent(Display * display, Window w, int event_mask,*
*XEvent * event)*

XCirculateSubwindows

Cycles the given window's children (subwindows) up or down in the stacking order.

void
*XCirculateSubwindows(Display * display, Window w, int direction)*

XCirculateSubwindowsDown

Lowers the highest obscured subwindow of the given window in the stacking order and circulates the lowest subwindow to the top.

void
*XCirculateSubwindowsDown(Display * display, Window w)*

XCirculateSubwindowsUp

Raises the lowest obscured subwindow of the given window in the stacking order and circulates the top subwindow to the bottom.

void
*XCirculateSubwindowsUp(Display * display, Window w)*

XClearArea

Clears the specified rectangular area in the given window.

void
*XClearArea(Display * display, Window w, int x, int y, unsigned int width,*
unsigned int height, Bool exposures)

XClearWindow

Clears the given window.

void
*XClearWindow(Display * display, Window w)*

XClipBox

Returns the smallest rectangle that encloses the given region.

void
*XClipBox(Region r, XRectangle * rect_return)*

XCloseDisplay

Closes the connection between the current client program and the specified X server and display.

void
*XCloseDisplay(Display * display)*

XConfigureWindow
Changes the given window's configuration (that is, its position, size, stacking order, and so on).

void
*XConfigureWindow(Display * display, Window w, unsigned int value_mask, XWindowChanges * values)*

XConvertSelection
Uses the value of a selection.

void
*XConvertSelection(Display * display, Atom selection, Atom target, Atom propriety, Window requestor, Time time)*

XCopyArea
Copies the specified drawable area: combines its source and destination rectangles.

void
*XCopyArea(Display * display, Drawable src, Drawable dest, GC gc, int src_x, int src_y, unsigned int width, unsigned int height, int dest_x, int dest_y)*

XCopyColormapAndFree
Copies the given colormap, returning the new ID; used to obtain a new virtual colormap.

Colormap
*XCopyColormapAndFree(Display * display, Colormap cmap)*

XCopyGC
Copies some or all of the components of one graphics context to another one.

void
*XCopyGC(Display * display, GC src, unsigned long valuemask, GC dest)*

XCopyPlane
Copies a single plane from the given source drawable into the entire depth of the destination drawable.

void
*XCopyPlane(Display * display, Drawable src, Drawable dest, GC gc,*
int src_x, int src_y, unsigned int width, unsigned int height, int dest_x,
int dest_y, unsigned long plane)

XCreateBitmapFromData
Creates a single-plane pixmap from the given X11-format bitmap data.

Pixmap
*XCreateBitmapFromData(Display * display, Drawable drawable,*
*char * data, unsigned int width, unsigned int height)*

XCreateColormap
Creates a colormap of the specified *visual* class.

Colormap
*XCreateColormap(Display * display, Window w, Visual * visual, int alloc)*

XCreateFontCursor
Creates a cursor using the standard X11 cursor font, the set of standard cursor shapes or characters.

Cursor
*XCreateFontCursor(Display * display, unsigned int shape)*

XCreateGC
Creates a new graphics context according to the given specifications.

GC
*XCreateGC(Display * display, Drawable drawable,*
*unsigned long valuemask, XGCValues * values)*

XCreateGlyphCursor
Creates a cursor from the indicated fonts.

Cursor
*XCreateGlyphCursor(Display * display, Font source_font, Font mask_font,*
unsigned int source_char, unsigned int mask_char,
*XColor * foreground_color, XColor * background_color)*

XCreateImage
Creates an image, allocating it sufficient memory.

*XImage **
*XCreateImage(Display * display, Visual * visual, unsigned int depth,*
*int format, int offset, char * data, unsigned int width,*
unsigned int height, int bitmap_pad, int bytes_per_line)

XCreateFontCursor
Creates a cursor using the cursor font.

Cursor
*XCreateFontCursor(Display * display, unsigned int shape)*

XCreatePixmap
Creates a pixmap.

Pixmap
*XCreatePixmap(Display * display, Drawable * drawable,*
unsigned int width, unsigned int height, unsigned int depth)

XCreatePixmapCursor
Creates a cursor.

Cursor
*XCreatePixmapCursor(Display * display, Pixmap source, Pixmap mask,*
*XColor * foreground_color, XColor * background_color,*
unsigned int x_hot, unsigned int y_hot)

XCreatePixmapFromBitmapData
Creates a pixmap of the specified depth using bitmap information.

Pixmap
*XCreatePixmapFromBitmapData(Display * display, Drawable * drawable,*
*char * data, unsigned int width, unsigned int height, unsigned long fg,*
unsigned long bg, unsigned int depth)

XCreateRegion
Creates a new empty region of undefined size (see **XPolygonRegion**).

Region
XCreateRegion(void)

XCreateSimpleWindow
Creates a simple unmapped I/O subwindow of the given parent window according to the specified size, background and border.

Window
*XCreateSimpleWindow(Display * display, Window parent, int x, int y,*
unsigned int width, unsigned int height, unsigned int border_width,
unsigned long border, unsigned long background)

XCreateWindow

Creates an unmapped subwindow of the given I/O window, *parent*, and sets the specified attributes.

void
*XCreateWindow(Display * display, Window parent, int x, int y,*
unsigned int width, unsigned int height, unsigned int border_width,
*int depth, unsigned int class, Visual * visual, unsigned long valuemask,*
*XSetWindowAttributes * attributes)*

XDefineCursor

Assigns the specified cursor to a window.

void
*XDefineCursor(Display * display, Window w, Cursor * cursor)*

XDeleteContext

Deletes the context entry for a specified window and type.

int
*XDeleteContext(Display * display, Window w, XContext context)*

XDeleteModifiermapEntry

Deletes a modifier keymap entry.

*XModifierKeymap **
*XDeleteModifiermapEntry(XModifierKeymap * modmap,*
KeyCode keysym_entry, int modifier)

XDeleteProperty

Deletes a window property.

void
*XDeleteProperty(Display * display, Window w, Atom property)*

XDestroyImage

Removes the memory allocation that is currently associated with the given image.

int
*XDestroyImage(XImage * ximage)*

XDestroyRegion
Removes the memory store allocation that is currently associated with the given region.

void
XDestroyRegion(Region r)

XDestroySubwindows
Removes all the children of the given window, starting with the lowest subwindow in the stack and ending with the top one.

void
*XDestroySubwindows(Display * display, Window w)*

XDestroyWindow
Unmaps the given window and destroys all its children.

void
*XDestroyWindow(Display * display, Window window)*

XDisableAccessControl
Overrides the host access list and enables the clients of any host to access the server.

void
*XDisableAccessControl(Display * display)*

XDisplayMotionBufferSize
Returns the size of the motion buffer on the given display.

unsigned long
*XDisplayMotionBufferSize(Display * display)*

XDisplayName
Returns the name of the display that will be contacted by a call to **XOpenDisplay** if passed the given string.

*char **
*XDisplayName(const char * string)*

XDrawArc
Draws a circular or elliptical arc to fit within a defined rectangle.

void
*XDrawArc(Display * display, Drawable drawable, GC gc, int x, int y,*
unsigned int width, unsigned int height, int angle1, int angle)

XDrawArcs
Draws more than one arc; multiple version of **XDrawArc.**

void
*XDrawArcs(Display * display, Drawable drawable, GC gc, XArc * arcs,*
int narcs)

XDrawImageString
Draws a string of 8-bit image text characters, including the foreground and background of these characters, according to the given graphics context.

void
*XDrawImageString(Display * display, Drawable drawable, GC gc, int x,*
*int y, const char * string, int length)*

XDrawImageString16
As **DrawImageString,** except that it draws 16-bit image text characters.

void
*XDrawImageString16(Display * display, Drawable drawable, GC gc, int x,*
*int y, const XChar2b * string, int length)*

XDrawLine
Draws a line between two given points, using the components of the graphics context.

void
*XDrawLine(Display * display, Drawable drawable, GC gc, int x1, int y1,*
int x2, int y2)

XDrawLines
Draws connected lines between given points, using the components of the graphics context.

void
*XDrawLines(Display * display, Drawable drawable, GC gc,*
*XPoint * points, int npoints, int mode)*

XDrawPoint
Draws a single point within the specified drawable, using the foreground pixel component of the given graphics context.

void
*XDrawPoint(Display * display, Drawable drawable, GC gc, int x, int y)*

XDrawPoints

Draws many points within the specified drawable, using the foreground pixel component of the given graphics context.

void
*XDrawPoints(Display * display, Drawable drawable, GC gc,*
*XPoint * points, int npoints, int mode)*

XDrawRectangle

Draws the outline of a rectangle, according to the given dimensions and graphics context.

void
*XDrawRectangle(Display * display, Drawable drawable, GC gc, int x,*
int y, unsigned int width, unsigned int height)

XDrawRectangles

Draws the outline of many rectangles.

void
*XDrawRectangles(Display * display, Drawable drawable, GC gc,*
*XRectangle * rectangles, int nrectangles)*

XDrawSegments

Draws many line segments which may be connected or unconnected (see **XDrawLines**).

void
*XDrawSegments(Display * display, Drawable drawable, GC gc,*
*XSegment * segments, int nsegments)*

XDrawString

Draws the foreground of a 8-bit text string starting at a defined point within the given drawable.

void
*XDrawString(Display * display, Drawable drawable, GC gc, int x, int y,*
*const char * string, int length)*

XDrawString16

Draws the foreground of a 2-byte text string starting at a defined point within the given drawable.

void
*XDrawString16(Display * display, Drawable drawable, GC gc, int x, int y,*
*const XChar2b * string, int length)*

XDrawText

Draws multiple 8-bit strings, which may contain different fonts, with the first string starting at a defined point within the drawable.

void
*XDrawText(Display * display, Drawable drawable, GC gc, int x, int y,*
*XTextItem * items, int nitems)*

XDrawText16

As **XDrawText**, only using 16-bit text strings.

void
*XDrawText16(Display * display, Drawable drawable, GC gc, int x, int y,*
*XTextItem16 * items, int nitems)*

XEHeadOfExtensionList

Returns a pointer to the head of the list of extsion data structures.

*ExtData **
XEHeadOfExtensionList(XEDataObject)

XEmptyRegion

Checks to see if a given region is empty or not.

int
XEmptyRegion(Region r)

XEnableAccessControl

Prevents the clients of any host accessing the server by using the host access control list to limit connection.

void
*XEnableAccessControl(Display * display)*

XEqualRegion

Compares two regions to see if they are the same in size, shape, and relative location.

int
XEqualRegion(Region r1, Region r2)

XEventsQueued
Looks at the event queue to see if there are any events queued.

```
int
XEventsQueued( Display * display, int mode )
```

XFetchBuffer
Gets the data from a given communications cut buffer.

```
char *
XFetchBuffer( Display * display, int * nbytes_return, int buffer )
```

XFetchBytes
Gets the data from cut buffer zero.

```
char *
XFetchBytes( Display * display, int * nbytes )
```

XFetchName
Returns the value of the name property of the given window.

```
Status
XFetchName( Display * display, Window w, char ** window_name_return )
```

XFillArc
Fills the specified arc according to the components of the graphics context.

```
void
XFillArc( Display * display, Drawable drawable, GC gc, int x, int y,
    unsigned int width, unsigned int height, int angle1, int angle2 )
```

XFillArcs
Fills multiple arcs according to the components of the graphics context.

```
void
XFillArcs( Display * display, Drawable drawable, GC gc, XArc * arcs,
    int narcs )
```

XFillPolygon
Fills the defined polygon using the components of the graphics context.

```
void
XFillPolygon( Display * display, Drawable drawable, GC gc,
    XPoint * points, int npoints, int shape, int mode )
```

XFillRectangle

Fills a rectangular area within the specified drawable using the components of the graphics context.

```
void
XFillRectangle( Display * display, Drawable drawable, GC gc, int x, int y,
    unsigned int width, unsigned int height )
```

XFillRectangles

Fills multiple rectangular areas within the specified drawable using the components of the graphics context.

```
void
XFillRectangles( Display * display, Drawable drawable, GC gc,
    XRectangle * rectangles, int nrectangles )
```

XFindContext

Gets data associated with the context manager, defined by the given context ID, that is assigned to the specified window.

```
void
XFindContext( Display * display, Window w, XContext context,
    caddr_t * data )
```

XFindOnExtensionList

Returns the extension data structure for the extension given.

```
ExtData *
XFindOnExtensionList( XExtData ** structure, int number )
```

XFlush

Flushes any queued output requests from buffer to display.

```
void
XFlush( Display * display )
```

XForceScreenSaver

Turns the screen saver on or off according to the specified mode (see **XActivateScreenSaver**).

```
void
XForceScreenSaver( Display * display, int mode )
```

XFree

Frees data in memory allocated by Xlib call.

void
*XFree(char * data)*

XFreeColormap
Discards the specified colormap and installs the default colormap.

void
*XFreeColormap(Display * display, Colormap cmap)*

XFreeColors
Frees the colormap cells or planes.

void
*XFreeColors(Display * display, Colormap cmap, unsigned long * pixels,*
int npixels, unsigned long planes)

XFreeCursor
Disassociates the cursor ID from the specified cursor, destroying the cursor.

void
*XFreeCursor(Display * display, Cursor cursor)*

XFreeExtensionList
Frees the memory associated with the X extension list.

void
*XFreeExtensionList(char ** list)*

XFreeFont
Frees any memory allocated to the specified font information structure and unloads the font.

void
*XFreeFont(Display * display, XFontStruct * font_struct)*

XFreeFontInfo
Frees the specified Font information structures without unloading the fonts.

void
*XFreeFontInfo(char ** names, XFontStruct * info, int actual_count)*

XFreeFontNames
Frees the font name strings in the specified array.

void
*XFreeFontNames(char ** list)*

XFreeFontPath
Frees the data and memory allocation associated with the specified array (see **XGetFontPath**).

void
*XFreeFontPath(char ** list)*

XFreeGC
Frees the memory associated with the specified graphics context, removing the GC from the server and display.

void
*XFreeGC(Display * display, GC gc)*

XFreeModifiermap
Frees the specified modifier keymap structure and destroys it.

void
*XFreeModifiermap(XModifierKeymap * modmap)*

XFreePixmap
Disassociates the ID from the specified pixmap.

void
*XFreePixmap(Display * display, Pixmap pixmap)*

XFreeStringList
Releases the storage used by a string array

void
*XFreeStringList(char ** list)*

XGCContextFromGC
Gets the resource ID from the specified graphics context structure.

GContext
XGContextFromGC(GC gc)

XGeometry
Calculates and returns the window geometry given the user geometry and default geometry.

int
*XGeometry(Display * display, int screen, const char * user_geom,*
*const char * default_geom, unsigned int bwidth, unsigned int fwidth,*
*unsigned int fheight, int xadder, int yadder, int * x_return,*
*int * y_return, int * width_return, int * height_return)*

XGetAtomName
Returns the name string for the given atom.

*char **
*XGetAtomName(Display * display, Atom atom)*

XGetClassHint
Returns the class property for the given window.

Status
*XGetClassHint(Display * display, Window w, XClassHint * class_hints)*

XGetCommand
Returns the WM_COMMAND property for the given window.

Status
*XGetCommand(Display * display, Window w, char *** argv_return,*
*int * argc_return)*

XGetDefault
Returns a character string containing the user's preferences (defaults) for the given program and option.

*char **
*XGetDefault(Display * display, const char * program, const char * option)*

XGetErrorDatabaseText
Looks up and returns error messages from the error database.

void
*XGetErrorDatabaseText(Display * display, const char * name,*
*const char * message, const char * default_string, char * buffer,*
int length)

XGetErrorText
Gets a character string that describes the error associated with the given error code.

void
*XGetErrorText(Display * display, int code, char * buffer, int length)*

XGetFontPath
Gets the search path string for the current font.

*char **
*XGetFontPath(Display * display, int * npaths_return)*

XGetFontProperty
Gets the value of a font property given its identifying atom.

Bool
*XGetFontProperty(XFontStruct * font_struct, Atom atom,*
*unsigned long * value_return)*

XGetGCValues
Returns the indicated values for the specified GC.

Bool
*XGetGCValues(Display * display, GC gc, unsigned long valuemask,*
*XCGValues * values_return)*

XGetGeometry
Gets the geometry of the specified drawableand any information about the root window.

Status
*XGetGeometry(Display * display, Drawable drawable,*
*Window * root_return, int * x_return, int * y_return,*
*unsigned int * width_return, unsigned int * height_return,*
*unsigned int * border_width_return, unsigned int * depth_return)*

XGetIconName
Gets the icon name property of the specified window.

Status
*XGetIconName(Display * display, Window w, char ** icon_name)*

XGetIconSizes
Gets the size property set for the specified window.

void
*XGetIconSizes(Display * display, Window w, XIconSize ** size_list,*
*int * count)*

XGetImage
Returns the contents of the specified rectangle as an X image structure.

XImage
*XGetImage(Display * display, Drawable drawable, int x, int y,*
unsigned int width, unsigned int height, unsigned long plane_mask,
int format)

XGetInputFocus
Gets the existing input focus window and the one it would revert to on becoming invisible.

void
*XGetInputFocus(Display * display, Window * focus, int * revert_to_return)*

XGetKeyboardControl
Gets a list of the current keyboard control values.

void
*XGetKeyboardControl(Display * display, XKeyboardState * values_return)*

XGetKeyboardMapping
Obtains the key symbols that are currently mapped to the given keycodes.

*KeySym **
*XGetKeyboardMapping(Display * display, KeyCode first_keycode,*
*int keycode_count, int * keysyms_per_keycode_return)*

XGetModifierMapping
Obtains the keycodes for the modifier keys (SHIFT, ALT, and so forth).

*XModifierKeymap **
*XGetModifierMapping(Display * display)*

XGetMotionEvents
Returns any pointer motion events for the specified window that occurred between the given start and stop times.

*XTimeCoord **
*XGetMotionEvents(Display * display, Window w, Time start, Time stop,*
*int * nevents_return)*

XGetNormalHints
Returns the size hints property for the specified window in its normal state (that is, not zoomed or iconified).

Status
*XGetNormalHints(Display * display, Window w, XSizeHints * hints)*

XGetPixel
Returns a single pixel value from an X image.

void
*XGetPixel(XImage * ximage, int x, int y)*

XGetPointerControl
Obtains the current pointer acceleration parameters.

void
*XGetPointerControl(Display * display, int * accel_numerator_return,*
*int * accel_denominator_return, int * threshold_return)*

XGetPointerMapping
Obtains the current mappings for the pointer buttons.

int
*XGetPointerMapping(Display * display, unsigned char * map_return,*
int nmap)

XGetRGBColormaps
Sets or resets a standard colourmap structure for a given window.

Status
*XGetRGBColormaps(Display * display, Window w,*
*XStandardColormap ** std_colourmap_return, int *count_return,*
Atom property)

XGetScreenSaver
Obtains the current screen saver parameters (see **XSetScreenSaver**).

void
*XGetScreenSaver(Display * display, int * timeout_return,*
*int * interval_return, int * prefer_blanking_return,*
*int * allow_exposures_return)*

XGetSelectionOwner
Gets the window ID of the current owner of the specified selection.

Window
*XGetSelectionOwner(Display * display, Atom selection)*

XGetSizeHints
Gets the size hints property for the specified window.

Status
*XGetSizeHints(Display * display, Window w, XSizeHints * hints,*
Atom property)

XGetStandardColormap
Obtains the standard colormap property for the specified window.

Status
*XGetStandardColormap(Display * display, Window w,*
*XStandardColormap * colourmap_return, Atom property)*

XGetSubImage
Gets a subimage from a defined area within the given drawable and copies it to a pre-existing destination X image (see **XGetImage, XSubImage**).

*XImage **
*XGetSubImage(Display * display, Drawable drawable, int x, int y,*
unsigned int width, unsigned int height, unsigned long plane_mask,
*int format, XImage * dest_image, int dest_x, int dest_y)*

XGetTextProperty
Gets the indicated property (of type TEXT) from the indicated window.

Status
*XGetTextProperty(Display * display, Window w,*
*XTextProperty * text_prop_return, Atom property)*

XGetTransientForHint
Gets the transient property for the specified window.

Status
*XGetTransientForHint(Display * display, Window w,*
*Window * prop_window_return)*

XGetVisualInfo
Returns a visual information structure that fits the attributes defined by the specified template.

*XVisualInfo **
*XGetVisualInfo(Display * display, long vinfo_mask,*
*XVisualInfo * vinfo_template_return, int * nitems_return)*

XGetWindowAttributes
Gets the current attributes for the specified window.

Status
*XGetWindowAttributes(Display * display, Window w,*
*XWindowAttributes * window_attributes_return)*

XGetWindowProperty
Gets the value of the given property if it matches the specified atom type.

int
*XGetWindowProperty(Display * display, Window w, Atom property,*
long long_offset, long long_length, Bool delete, Atom req_type,
*Atom * actual_type_return, int * actual_format_return,*
*unsigned long * nitems_return, unsigned long * bytes_after_return,*
*unsigned char ** prop_return)*

XGetWMClientMachine
Returns the WM_CLIENT_COMMAND property for the given window.

Status
*XGetWMClientMachine(Display * display, Window w,*
*XTextProperty * text_prop_return)*

XGetWMColormapWindows
Returns the WM_COLORMAP property for the given window.

Status
*XGetWMColormapWindows(Display * display, Window w,*
*Window ** colourmaps_windows_return, int * count_return)*

XGetWMHints
Gets the window manager hints property that is set for the given window.

*XWMHints **
*XGetWMHints(Display * display, Window w)*

XGetWMIconName
Returns the window's WM_ICON_NAME property.

Status
*XGetWMIconName(Display * display, Window w,*
*XTextProperty * text_prop_return)*

XGetWMName
Returns the window's WM_NAME property.

Status
*XGetWMName(Display * display, Window w,*
*XTextProperty * text_prop_return)*

XGetWMNormalHints
Returns the WM_NORMAL_HINTS property of the given window.

Status
*XGetWMNormalHints(Display * display, Window w,*
*XSizeHints * hints_return, long * supplied_return)*

XGetWMSizeHints
Returns the WM_SIZE_HINTS property of the given property on the given window.

Status
*XGetWMSizeHints(Display * display, Window w,*
*XSizeHints * hints_return, long * supplied_return, Atom property)*

XGetWMProtocols
Returns a list of protocols supported by the current window manager on the given top-level window

Status
*XGetWMProtocols(Display * display, Window w,*
*Atom ** protocols_return, int * count_return)*

Status XGetZoomHints
Gets the size hints property for the given zoomed window.

XWMHints
*XGetZoomHints(Display * display, Window w, XSizeHints * zhints_return)*

XGrabButton
Establishes a passive grab, allowing the calling client to grab control of the pointer when the given button is pressed.

void
*XGrabButton(Display * display, unsigned int button,*
unsigned int modifiers, Window grab_window, Bool owner_events,
unsigned int event_mask, int pointer_mode, int keyboard_mode,
Window confine_to, Cursor cursor)

XGrabKey

Establishes a passive grab, allowing the calling client to grab control of the keyboard when the given key is pressed.

void
*XGrabKey(Display * display, int keycode, unsigned int modifiers,*
Window grab_window, Bool owner_events, int pointer_mode,
int keyboard_mode)

XGrabKeyboard

Makes an active grab of the keyboard.

int
*XGrabKeyboard(Display * display, Window grab_window,*
Bool owner_events, int pointer_mode, int keyboard_mode, Time time)

XGrabPointer

Makes an active grab of the pointer.

int
*XGrabPointer(Display * display, Window grab_window,*
Bool owner_events, unsigned int event_mask, int pointer_mode,
int keyboard_mode, Window confine_to, Cursor cursor, Time time)

XGrabServer

Grabs the X server for the exclusive use of the calling client and queues requests from other clients.

void
*XGrabServer(Display * display)*

XIconifyWindow

Sends a request that a top-level window be iconified.

Status
*XIconifyWindow(Display * display, Window w, int screen_number)*

XIfEvent

Looks for events on the queue that match the given predicate.

void
*XIfEvent(Display * display, XEvent * event_return, Bool (* predicate)(*
*Display *, XEvent *, char *), char * args)*

XInitExtension
Initialises an extension to the X server

*XExtCodes **
*XInitExtension(Display * display, const char * extension_name)*

XInsertModifiermapEntry
Inserts a new entry in the given modifier map.

XModifierKeymap
*XInsertModifiermapEntry(XModifierKeymap * modmap,*
KeyCode keysym_entry, int modifier)

XInstallColormap
Sets up the given colormap.

void
*XInstallColormap(Display * display, Colormap cmap)*

XInternAtom
Returns the atom ID for the named property, if it exists.

Atom
*XInternAtom(Display * display, const char * property_name,*
Bool only_if_exists)

XIntersectRegion
Calculates and returns the intersection of the two specified regions.

void
XIntersectRegion(Region sra, Region srb, Region dr_return)

XKeycodeToKeysym
Returns the corresponding key symbol for the given keycode.

KeySym
*XKeycodeToKeysym(Display * display, KeyCode keycode, int index)*

XKeysymToKeycode
Returns the corresponding keycode for the given keysym.

KeyCode
*XKeysymToKeycode(Display * display, Keysym keysym)*

XKeysymToString
Returns a character string version of the given keysym.

*char **
XKeysymToString(Keysym keysym_str)

XKillClient
Closes the client that created the given resource, if already terminated, destroys the remaining resource(s).

void
*XKillClient(Display * display, XID resource)*

XListDepths
Returns an array of depths that are supported on the specified screen.

*int **
*XListDepths(Display * display, int screen_number, int * count_return)*

XListExtensions
Lists the current X extensions supported by the server.

*char ***
*XListExtensions(Display * display, int * nextensions_return)*

XListFonts
Lists the names of any available fonts that agree with the given pattern.

*char ***
*XListFonts(Display * display, const char * pattern, int maxnames,*
*int * actual_count_return)*

XListFontsWithInfo
As **XListFonts**, but also lists any associated information.

*char ***
*XListFontsWithInfo(Display * display, const char * pattern, int maxnames,*
*int * count_return, XFontStruct ** info_return)*

XListHosts
Lists all the hosts that can access the display (the current control access list).

*XHostAddress **
*XListHosts(Display * display, int * nhosts_return, Bool * state_return)*

XListInstalledColormaps
Lists all the colormaps that are currently installed for the screen associated with the given window.

*Colormap **
*XListInstalledColormaps(Display * display, Window w, int * num_return)*

XListPixmapFormats
Lists all of the pixmap formats supported by a given display.

*XPixmapFormatValues **
*XListPixmapFormats(Display * display, int * count_return)*

XListProperties
Lists the properties that are set for the specified window.

*Atom **
*XListProperties(Display * display, Window w, int * num_prop_return)*

XLoadFont
Loads the named font; if already loaded, returns its font ID.

Font
*XLoadFont(Display * display, const char * name)*

XLoadQueryFont
Loads the named font and returns the associated font information (see **XLoadFont, XQueryFont**).

XFontStruct
*XLoadQueryFont(Display * display, const char * name)*

XLookupColor
Finds and returns the RGB database value for the colorname and its closest hardware equivalent.

Status
*XLookupColor(Display * display, Colormap cmap, const char * colorname,*
*XColor * rgb_db_def, XColor * hardware_def)*

XLookupKeysym
Returns a keysym from a list associated with the keycode for the given keyboard event.

Keysym
*XLookupKeysym(XKeyEvent * event, int index)*

XLookupString
Looks for and returns the string and keysym associated with the given event.

int
*XLookupString(XKeyEvent * event, char * buffer_return, int num_bytes,*
*KeySym * keysym_return, XComposeStatus * status)*

XLowerWindow
Lowers the given window's position in the stacking order in relation to its siblings.

void
*XLowerWindow(Display * display, Window w)*

XMapRaised
Maps the given Window and raises it in the stacking order.

void
*XMapRaised(Display * display, Window w)*

XMapSubwindows
Maps the subwindows of the given window.

void
*XMapSubwindows(Display * display, Window w)*

XMapWindow
Maps the given window, allowing it to be displayed.

void
*XMapWindow(Display * display, Window w)*

XMaskEvent
Removes the next event in the input queue that matches the given event mask.

void
*XMaskEvent(Display * display, unsigned long event_mask,*
*XEvent * event_return)*

XMatchVisualInfo
Gets the visual information that matches the given screen, depth and class.

void
*XMatchVisualInfo(Display * display, int screen, int depth, int class,*
*XVisualInfo * vinfo)*

XMaxRequestSize
Returns the maximum size of a single request supported by the X server.

long
*XMaxRequestSize(Display * display)*

XMoveResizeWindow
Changes the size and position of the given window.

void
*XMoveResizeWindow(Display * display, Window w, int x, int y,*
unsigned int width, unsigned int height)

XMoveWindow
Changes the position of the given window.

void
*XMoveWindow(Display * display, Window w, int x, int y)*

XNewModifiermap
Makes and returns a new modifier keymap structure.

*XModifierKeymap **
XNewModifiermap(int max_keys_per_mod)

XNextEvent
Gets the next input event in the queue.

void
*XNextEvent(Display * display, XEvent * event_return)*

XNoOp
Sends a No Operation request to the X server.

void
*XNoOp(Display * display)*

XOffsetRegion
Alters the offset of the given region by the specified amount.

void
XOffsetRegion(Region r, int dx, int dy)

XOpenDisplay
Opens the named display and enables the client program to communicate with the server.

*Display **
*XOpenDisplay(const char * display_name)*

XParseColor
Returns the RGB values for the named colour or hexadecimal value.

Status
*XParseColor(Display * display, Colormap colormap, const char * spec,*
*XColor * rgb_db_def_return)*

XParseGeometry
Calculates window size and position from the standard string.

int
*XParseGeometry(const_char * parsestring, int * x_return, int * y_return,*
*unsigned int * width_return, unsigned int * height_return)*

XPeekEvent
Looks at the event at the top of the input queue and takes a copy of that event while leaving the queue unchanged; if the queue is empty, it waits until an event occurs.

void
*XPeekEvent(Display * display, XEvent * event_return)*

XPeekIfEvent
As **XPeekEvent**, only does not wait if the queue is empty.

void
*XPeekIfEvent(Display * display, XEvent * event,*
Bool (predicate)(Display *, XEvent *, char *), char * arg)*

XPending
Returns the number of events pending on the input queue.

int
*XPending(Display * display)*

Xpermalloc
Allocates *size* bytes of 'permanent' memory.

```
char
XPermalloc( unsigned int size )
```

XPointInRegion
Checks to see if the specified point is within the given region.

```
int
XPointInRegion( Region r, int x, int y )
```

XPolygonRegion
Connects the given points to make a polygon and enables that region to be referred to again later.

```
Region
XPolygonRegion( XPoint * points, int n, int fill_rule )
```

XPutBackEvent
Puts the given event back onto the input queue.

```
void
XPutBackEvent( Display * display, XEvent * event )
```

XPutImage
Draws a rectangular image, or a section of a rectangular image, in a window or pixmap using the graphics context components.

```
void
XPutImage( Display * display, Drawable drawable, GC gc,
    XImage * image, int src_x, int src_y, int dst_x, int dst_y,
    unsigned int width, unsigned int height )
```

XPutPixel
Changes the pixel value at the specified point in the X image for the given pixel value.

```
int
XPutPixel( XImage * ximage, int x, int y, unsigned long pixel )
```

XQueryBestCursor
Gets the 'best' cursor whose size most closely corresponds to the given parameters.

Status
*XQueryBestCursor(Display * display, Drawable drawable,*
unsigned int width, unsigned int height, unsigned int width_return,
unsigned int height_return)

XQueryBestSize

Gets the 'best' supported size that most closely agrees with the given parameters.

Status
*XQueryBestSize(Display * display, int class, Drawable drawable,*
unsigned int width, unsigned int height, unsigned int rwidth,
unsigned int rheight)

XQueryBestStipple

Gets the 'best' supported stipple shape that most closely agrees with the given parameters.

Status
*XQueryBestStipple(Display * display, Drawable drawable,*
unsigned int width, unsigned int height, unsigned int width_return,
unsigned int height_return)

XQueryBestTile

Gets the 'best' supported tile that most closely agrees with the given parameters.

Status
*XQueryBestTile(Display * display, Drawable drawable, unsigned int width,*
unsigned int height, unsigned int width_return,
unsigned int height_return)

XQueryColor

Finds and returns the RGB values and flags for a given pixel value.

void
*XQueryColor(Display * display, Colormap cmap, XColor * colorcell_def)*

XQueryColors

Finds and returns an array of RGB values.

void
*XQueryColors(Display * display, Colormap cmap, XColor * colorcell_def,*
int ncolors)

XQueryExtension

Finds out if the named extension is available.

```
Bool
XQueryExtension( Display * display, const char * name,
    int * major_opcode_return, int * first_event_return,
    int * first_error_return )
```

XQueryFont

Finds and returns information on the specified font.

```
XFontStruct *
XQueryFont( Display * display, XID font_ID )
```

XQueryKeymap

Returns a bit vector that reflects the current logical state of the keyboard.

```
void
XQueryKeymap( Display * display, char keys[ 32 ] )
```

XQueryPointer

Finds out if the pointer is in the same screen as the specified window.

```
Bool
XQueryPointer( Display * display, Window w, Window * root_return,
    Window * child_return, int * root_x_return, int * root_y_return,
    int * win_x_return, int * win_y_return, unsigned int * mask_return )
```

XQueryTextExtents

Finds and returns measurements for the specified string in the given font.

```
int
XQueryTextExtents( Display * display, XID font_ID, const char * string,
    int nchars, int * direction_return, int * ascent_return,
    int * descent_return, XCharStruct * overall_return )
```

XQueryTextExtents16

Finds and returns measurements for the specified 16-bit character string in the given font.

```
int
XQueryTextExtents16( Display * display, XID font_ID,
    const XChar2b * string, int nchars, int * direction_return,
    int * ascent_return, int * descent_return, XCharStruct * overall_return )
```

XQueryTree

Finds out about the given window's hierarchy, returning its root ID, parent ID, number of children and a pointer to a list of its children.

```
Status
XQueryTree( Display * display, Window w, Window * root_return,
    Window * parent_return, Window ** children_return,
    unsigned int * nchildren_return )
```

XRaiseWindow

Lifts the given window up to the top of the stacking order, while keeping it in the same x/y position in the display.

```
void
XRaiseWindow( Display * display, Window w )
```

XReadBitmapFile

Reads the named file, which contains bitmap information, into a new pixmap of the specified size.

```
int
XReadBitmapFile( Display * display, Drawable drawable,
    const char * filename, unsigned int * width_return,
    unsigned int * height_return, Pixmap * bitmap_return,
    int * x_hot_return, int * y_hot_return )
```

XRebindKeysym

Rebinds the meaning of a keysym for a client; when the corresponding key is pressed at the same time as the relevant modifier, *keysym* and *string* are returned.

```
void
XRebindKeysym( Display * display, KeySym keysym, KeySym * mod_list,
    int mod_count, const unsigned char * string, int num_bytes )
```

XRecolorCursor

Changes the colour of the given cursor to the specified background and foreground colours.

```
void
XRecolorCursor( Display * display, Cursor cursor,
    XColor * foreground_color, XColor * background_color )
```

XReconfigureWMWindow

Sends a request that a top-level window be reconfigured.

Status
*XRecoconfigureWindow(Display * display, Window w, int screen_number,*
*unsigned int value_mask, XWindowChanges * values)*

XRectInRegion
Finds out if a given rectangle is within a particular region.

int
XRectInRegion(Region r, int x, int y, unsigned int width,
unsigned int height)

XRefreshKeyboardMapping
Updates the calling application's stored keyboard information to reflect the current mapping of keycodes to keysyms.

void
*XRefreshKeyboardMapping(XMappingEvent * event)*

XRemoveFromSaveSet
Removes the subwindows of the given window from the calling client's save set.

void
*XRemoveFromSaveSet(Display * display, Window w)*

XRemoveHost
Removes the specified host from the access control list.

void
*XRemoveHost(Display * display, XHostAddress * host)*

XRemoveHosts
Removes many hosts from the access control list.

void
*XRemoveHosts(Display * display, XHostAddress * hosts, int num_hosts)*

XReparentWindow
Reparents a window by inserting another window in the hierarchy, just below its current parent.

void
*XReparentWindow(Display * display, Window win, Window parent, int x,*
int y)

XResetScreenSaver
Resets the screen saver: redisplays the screen if the screen saver is running.

void
*XResetScreenSaver(Display * display)*

XResizeWindow
Alters the size of the given window according to the specified width and height.

void
*XResizeWindow(Display * display, Window w, unsigned int width,*
unsigned int height)

XRestackWindow
Alters the stacking order of sibling windows.

void
*XRestackWindow(Display * display, Window * windows, int nwindows)*

XResourceManagerString
Returns the default string used by the resource manager on the given display.

*char **
*XResourceManagerString(Display * display)*

XrmGetFileDatabase
Creates a resource database with information from the named file.

XrmDatabase
*XrmGetFileDatabase(const char * filename)*

XrmDestroyDatabase
Destroys a resource database and frees any allocated memory used by that database.

void
XrmDestroyDatabase(XrmDatabase database)

XrmGetResource
Looks for a resource of the given string name and class within the specified resource database.

Bool
*XrmGetResource(XrmDatabase database, const char * str_name,*
*const char * str_class, char ** str_type_return,*
*XrmValue * value_return)*

XrmGetStringDatabase
Creates a resource database with information from the given character string.

```
XrmDatabase
XrmGetStringDatabase( const char * data )
```

XrmInitialize
Initialises the resource manager.

```
void
XrmInitialize( void )
```

XrmMergeDatabases
Merges the contents of two existing databases to create one combined resource database; the specified target database is modified with the contents of the original source database, whereupon the source is then destroyed.

```
void XrmMergeDatabases( XrmDatabase source_db,
    XrmDatabase * target_db )
```

XrmParseCommand
Parses the command line arguments and loads options into the specified resource database.

```
void
XrmParseCommand( XrmDatabase * db, XrmOptionList table,
    int table_count, const char * name, int * argc, char ** argv )
```

XrmPutFileDatabase
Places a copy of the given database in the specified file.

```
void
XrmPutFileDatabase( XrmDatabase database, const char * stored_db )
```

XrmPutLineResource
Inserts a resource entry in the specified database, where the resource name and value are given as a single character string.

```
void
XrmPutLineResource( XrmDatabase * database, const char * line )
```

XrmPutResource
Inserts information in the specified resource database.

void
*XrmPutResource(XrmDatabase * database, const char * specifier,*
*const char * type, XrmValue * value)*

XrmPutStringResource

Inserts a resource entry in the specified database, where the resource name and value are given in separate character strings.

void
*XrmPutStringResource(XrmDatabase * database, const char * resource,*
*const char * value)*

XrmQGetResource

Finds and returns a resource in the specified database that matches the fully qualified name, class and type arguments; it acts like **XrmGetResource**, except the arguments here refer to quarks rather than to strings.

Bool
XrmQGetResource(XrmDatabase database, XrmNameList quark_name,
*XrmClassList quark_class, XrmRepresentation * quark_type,*
*XrmValue * value)*

XrmQGetSearchList

Searches a list of resource names and classes and returns a list of database levels ordered by the likelihood of a match.

Bool
XrmQGetSearchList(XrmDatabase database, XrmNameList names,
XrmClassList classes, XrmSearchList search_list_return,
int list_length_return)

XrmQGetSearchResource

Searches a list of ordered database levels for a specified resource.

Bool
XrmQGetSearchResource(XrmSearchList search_list, XrmName name,
*XrmClass class, XrmRepresentation * type_return,*
*XrmValue * value_return)*

XrmQPutResource

Places a resource entry in a database using the quarks argument.

void
*XrmQPutResource(XrmDatabase * database, XrmBindingList bindings,*
*XrmQuarkList quarks, XrmRepresentation type, XrmValue * value)*

XrmQPutStringResource
Places a string resource value in a database using the quarks argument.

void
*XrmQPutStringResource(XrmDatabase * database,*
*XrmBindingList bindings, XrmQuarkList quarks, const char * value)*

XrmQuarkToString
Converts the given quark to its equivalent character string.

XrmString
XrmQuarkToString(XrmQuark quark)

XrmStringToBindingQuarkList
Converts the given string to a binding list and a quark list.

void
*XrmStringToBindingQuarkList(const char * string,*
XrmBindingList bindings, XrmQuarkList quarks)

XrmStringToQuark
Converts the given string to a quark, binding the quark to that string.

XrmQuark
*XrmStringToQuark(const char * string)*

XrmStringToQuarkList
Converts the given string to a quark list.

void
*XrmStringToQuarkList(const char * string, XrmQuarkList quarks)*

XrmUniqueQuark
Allocates a new quark, one that has no string equivalent.

XrmQuark
XrmUniqueQuark(void)

XRotateBuffers
Cycles the cut buffers by a given number of positions.

void
*XRotateBuffers(Display * display, int rotate)*

XRotateWindowProperties

Cycles the window properties in the property array of the specified window according to the given amount and direction.

void
*XRotateWindowProperties(Display * display, Window w,*
*Atom * properties, int num_prop, int npositions)*

XSaveContext

Saves a data value, which corresponds to the given window's data and data context type, in the context manager database.

int
*XSaveContext(Display * display, Window w, XContext context,*
*const void * data)*

XScreenNumberOfScreen

Returns the screen index of the specified screen

int
*XScreenNumberOfScreen(Screen * screen)*

XSelectInput

Selects an input event that matches the given mask and then sends it to the specified window.

void
*XSelectInput(Display * display, Window w, unsigned long event_mask)*

XSendEvent

Sends an event to the specified window.

Status
*XSendEvent(Display * display, Window w, Bool propagate,*
*unsigned long event_mask, XEvent * event)*

XSetAccessControl

Turns on or off access control checking.

void
*XSetAccessControl(Display * display, int mode)*

XSetAfterFunction

Sets a user-defined function to be called after all Xlib functions.

```
int( *
    XSetAfterFunction( Display * display, int( * proc )( Display * ) )() )
```

XSetArcMode

Sets the arc mode in the given graphics context to specify chords or pie-slices.

```
void XSetArcMode( Display * display, GC gc, int arc_mode )
```

XSetBackground

Sets or resets the background pixel value in the given graphics context.

```
void
XSetBackground( Display * display, GC gc, unsigned long background )
```

XSetClassHint

Sets or resets the class hint property of the specified window.

```
void
XSetClassHint( Display * display, Window w, XClassHint * class_hints )
```

XSetClipMask

Sets the clip mask component of the given graphics context to the specified pixmap.

```
void
XSetClipMask( Display * display, GC gc, Pixmap clip_mask )
```

XSetClipOrigin

Sets or resets the clip origin in the given graphics context.

```
void
XSetClipOrigin( Display * display, GC gc, int clip_x_orgin,
    int clip_y_origin )
```

XSetClipRectangles

Sets the clip mask component of the given graphics context to a list of rectangles whose position is relative to the specified origin.

```
void
XSetClipRectangles( Display * display, GC gc, int clip_x_orgin,
    int clip_y_origin, XRectangle * rectangles, int nrects, int ordering )
```

XSetCloseDownMode

Sets the close down mode for a client; that is, prescribes what will happen to the dependent resources when a client is closed down.

void
*XSetCloseDownMode(Display * display, int close_mode)*

XSetCommand
Sets or resets the specified window's command property atom according to the values of the shell command line arguments.

void
*XSetCommand(Display * display, Window w, char ** argv, int argc)*

XSetDashes
Sets or resets the dash style in the given graphics context.

void
*XSetDashes(Display * display, GC gc, int dash_offset,*
*const char * dash_list, int n)*

XSetErrorHandler
Sets up a handler to handle non-fatal error events.

*int (**
*XSetErrorHandler(int(* handler)(Display *, XErrorEvent *))()*

Sets up the fill rule in the given graphics context to specify how a polygon is to be filled when there are overlapping areas.

*void XSetFillRule(Display * display, GC gc, int fill_rule)*

XSetFillStyle
Sets the fill style in the given graphics context to be solid, tiled, or stippled.

void
*XSetFillStyle(Display * display, GC gc, int fill_style)*

XSetFont
Sets the font in the given graphics context.

void
*XSetFont(Display * display, GC gc, Font font)*

XSetFontPath
Specifies the search path for font lookup. Directories are searched in the order in which they are listed. There is only one search path for each X Server.

void
*XSetFontPath(Display * display, char ** directories, int ndirs)*

XSetForeground
Sets or resets the foreground pixel value in the given graphics context.

void
*XSetForeground(Display * display, GC gc, unsigned long foreground)*

XSetFunction
Sets the bitwise logical function in the given graphics context.

void
*XSetFunction(Display * display, GC gc, int function)*

XSetGraphicsExposures
Sets the graphics exposures in the given graphics context.

void
*XSetGraphicsExposures(Display * display, GC gc,*
Bool graphics_exposures)

XSetIconName
Sets the name to be displayed in the given window's icon to the specified string.

void
*XSetIconName(Display * display, Window w, const char * icon_name)*

XSetIconSizes
Sets or resets the size preferences for icons associated with given window.

void
*XSetIconSize(Display * display, Window w, XIconSize * size_list, int count)*

XSetInputFocus
Sets or resets the keyboard focus window.

void
*XSetInputFocus(Display * display, Window focus, int revert_to, Time time)*

XSetIOErrorHandler
Sets up a handler to handle fatal I/O error events.

*int (**
*XSetIOErrorHandler(int(* handler)(Display *))()*

Sets up the line drawing attributes for the given graphics context to prescribe the line width and style to be used.

void
*XSetLineAttributes(Display * display, GC gc, unsigned int line_width,*
int line_style, int cap_style, int join_style)

XSetModifierMapping
Sets up which keycodes are to be used as modifiers (ALT, SHIFT, and so on).

void
*XSetModifierMapping(Display * display, XModifierKeymap * mod_map)*

XSetNormalHints
Sets or resets the size hints property of the given window in normal state (that is, not zoomed or iconified).

void
*XSetNormalHints(Display * display, Window w, XSizeHints * hints)*

XSetPlaneMask
Sets the plane mask in the given graphics context.

void
*XSetPlaneMask(Display * display, GC gc, unsigned long plane_mask)*

XSetPointerMapping
Sets or resets the pointer button mapping.

int
*XSetPointerMapping(Display * display, const unsigned char * map,*
int nmap)

XSetRGBColormaps
Sets or resets a standard colourmap structure for a given window.

void
*XSetRGBColormaps(Display * display, Window w,*
*XStandardColormap * std_colourmap, int count, Atom property)*

XSetRegion
Sets the clip mask in the given graphics context to the specified region.

void
*XSetRegion(Display * display, GC gc, Region r)*

XSetScreenSaver

Sets up parameters that specify the action of the screen saver.

```
void
XSetScreenSaver( Display * display, int timeout, int interval,
    int prefer_blanking, int allow_exposures )
```

XSetSelectionOwner

Specifies the owner of the selection property atom and the Time when a grab should happen.

```
void
XSetSelectionOwner( Display * display, Atom selection, Window owner,
    Time time )
```

XSetSizeHints

Sets or resets the size hints property for the given window.

```
void
XSetSizeHints( Display * display, Window w, XSizeHints * hints,
    Atom property )
```

XSetStandardColormap

Sets or resets the standard colormap property associated with the given window.

```
void
XSetStandardColormap( Display * display, Window w,
    XStandardColormap * cmap, Atom property )
```

XSetStandardProperties

Sets or resets the essential properties for the given window (that is, the minimum set of properties required).

```
void
XSetStandardProperties( Display * display, Window w,
    const char * window_name, const char * icon_name,
    Pixmap * icon_pixmap, char ** argv, int argc, XSizeHints * hints )
```

XSetState

Sets or resets the foreground and background pixel values, the logical function and the plane mask in the given graphics context.

void
*XSetState(Display * display, GC gc, unsigned long foreground,*
unsigned long background, int function, unsigned long plane_mask)

XSetStipple
Specifies the stipple pixmap for the given graphics context.

void
*XSetStipple(Display * display, GC gc, Pixmap * stipple)*

XSetSubwindowMode
Sets or resets the subwindow mode for the given graphics context.

void
*XSetSubwindowMode(Display * display, GC gc, int subwindow_mode)*

XSetTextProperty
Sets a specified property of type TEXT on the specified window

void
*XSetTextProperty(Display * display, Window w, XTextProperty * text_prop,*
Atom property)

XSetTile
Specifies the tile pixmap for the given graphics context.

void
*XSetTile(Display * display, GC gc, Pixmap * tile)*

XSetTransientForHint
Sets or resets the XA_WM_TRANSIENT_FOR property for the given window.

void
*XSetTransientForHint(Display * display, GC gc, Window prop_window)*

XSetTSOrigin
Sets or resets the origin of the tile or stipple fill pattern in the given graphics context.

void
*XSetTSOrigin(Display * display, GC gc, int ts_x_orgin, int ts_y_origin)*

XSetWindowBackground
Sets or resets the background pixel value that is to be used for background filling in the given window.

void
*XSetWindowBackground(Display * display, Window w,*
unsigned long background_pixel)

XSetWindowBackgroundPixmap
Sets or resets the background tile pixmap for the given window.

void
*XSetWindowBackgroundPixmap(Display * display, Window w,*
Pixmap background_tile)

XSetWindowBorder
Sets or resets the given window's border pixel value and redraws the border accordingly.

void
*XSetWindowBorder(Display * display, Window w,*
unsigned long border_pixel)

XSetWindowBorderPixmap
Sets or resets the given window's border tile pixmap and redraws the border accordingly.

void
*XSetWindowBorderPixmap(Display * display, Window w,*
Pixmap border_tile)

XSetWindowBorderWidth
Sets or resets the border width of the given window.

void
*XSetWindowBorderWidth(Display * display, Window w,*
unsigned int width)

XSetWindowColormap
Sets or resets the colormap for the specified window.

void
*XSetWindowColormap(Display * display, Window w, Colormap cmap)*

XsetWMClientMachine
Sets or resets the WM_CLIENT_COMMAND property for the given window.

void
*XSetWMClientMachine(Display * display, Window w,*
*XTextProperty * text_prop)*

XSetWMColormapWindows
Sets or resets the WM_COLORMAP_WINDOWS property for the specified window.

```
void
XSetWMColormapWindows( Display * display, Window w,
    Window * colourmap_windows, int count )
```

XSetWMHints
Sets or resets the window manager hints property for the given window.

```
void
XSetWMHints( Display * display, Window w, XWMHints * wmhints )
```

XSetWMIconName
Sets the WM_ICON_NAME property for the given window

```
void
XSetWMIconName( Display * display, Window w,
    XTextProperty * text_prop )
```

XSetWMName
Sets the WM_NAME property for the given window

```
void
XSetWMName( Display * display, Window w, XTextProperty * text_prop )
```

XSetWMNormalHints
Sets or resets the WM_NORMAL_HINTS property of the given window.

```
void
XSetWMNormalHints( Display * display, Window w, XSizeHints * hints )
```

XSetWMProperties
Sets or resets the standard properties required by window managers for the given window

```
void
XSetWMProperties( Display * display, Window w,
    XTextProperty * window_name, XTextProperty * icon_name,
    char ** argv, int argc, XSizeHints * normal_hints,
    XWMHints * wm_hints, XClassHint * class_hint )
```

XSetWMProtocols
Sets the WM_PROTOCOLS property on the given window

void
*XSetWMProtocols(Display * display, Window w, Atom * property,*
int count)

XSetWMSizeHints
Sets or resets the WM_SIZE_HINTS property for the given property of the given window.

void
*XSetWMSizeHints(Display * display, Window w, XWMSizeHints * hints,*
Atom property)

XSetZoomHints
Sets or resets the size hints property for the given window in zoomed state.

void
*XSetZoomHints(Display * display, Window w, XSizeHints * zhints)*

XShrinkRegion
Alters the dimensions of the given region by a specified amount, using positive values to expand the region or negative values to contract it.

void
XShrinkRegion(Region r, int dx, int dy)

XStringListToTextProperty
Converts a list of strings into a TextProperty structure.

Status
*XStringListToTextProperty(char ** list, int count,*
*XTextProperty * text_prop_return)*

XStoreBuffer
Stores information in one of the cut buffers.

void
*XStoreBuffer(Display * display, const char * bytes, int nbytes, int buffer)*

XStoreBytes
Stores the given information in cut buffer 0.

void
*XStoreBytes(Display * display, const char * bytes, int nbytes)*

XStoreColor

Sets or resets the value of the specified colorcell in the given colormap to the nearest equivalent hardware RGB values.

void
*XStoreColor(Display * display, Colormap cmap, XColor * colorcell_def)*

XStoreColors

Sets or resets the value of each of the colorcells in the given colormap to the nearest equivalent hardware RGB values.

void
*XStoreColors(Display * display, Colormap cmap, XColor * colorcell_def, int ncolors)*

XStoreName

Sets the specified string as the name of the given window.

void
*XStoreName(Display * display, Window w, char * window_name)*

XStoreNamedColor

Looks for the colour name string and stores its value in the specified colormap cell.

void
*XStoreNamedColor(Display * display, Colormap cmap, const char * colorname, unsigned long pixel, int flags)*

XStringToKeysym

Converts the given string to its corresponding keysym.

KeySym
*XStringToKeysym(const char * string)*

XSubImage

Creates a subimage within the given X image according to the specified dimensions.

XImage
*XSubImage(XImage * ximage, int x, int y, unsigned int subimage_width, unsigned int subimage_height)*

XSubtractRegion

Subtracts one region from another and returns the difference.

void
XSubtractRegion(Region sra, Region srb, Region dr_return)

XSync

Flushes the output buffer and waits for all outstanding events and errors to be processed by the server.

void
*XSync(Display * display, int discard)*

XSynchronize

Turns synchronisation on or off for the purposes of debugging and returns the previous After function.

*int(**
*XSynchronize(Display * display, Bool onoff))*

Returns the measurements for the given string and font.

XTextExtents

Returns the measurements for the given 8-bit character string in the named font.

void
*XTextExtents(XFontStruct * font_struct, const_char * string, int nchars,*
*int * direction_return, int * ascent_return, int * descent_return,*
*XCharStruct * overall_return)*

XTextExtents16

Returns the measurements for the given 16-bit character string in the named font.

void
*XTextExtents16(XFontStruct * font_struct, const XChar2b * string,*
*int nchars, int * direction_return, int * ascent_return,*
*int * descent_return, XCharStruct * overall_return)*

XTextPropertyToStringList

Converts an XTextProperty structure into an array of strings

Status
*XTextProprtyToStringList(XTextProperty * text_prop, char *** list_return,*
*int * count_return)*

XTextWidth

Returns the width, in pixels, of an 8-bit character string of a given length and font.

```
int
XTextWidth( XFontStruct * font_struct, const char * string, int count )
```

XTextWidth16

Returns the width, in pixels, of a 16-bit character string of a given length and font.

```
int
XTextWidth16( XFontStruct * font_struct, const XChar2b * string,
    int count )
```

XTranslateCoordinates

Translates the coordinates in the source window to those in the destination window, if possible.

```
Bool
XTranslateCoordinates( Display * display, Window src_w, Window dest_w,
    int src_x, int src_y, int dest_x_return, int dest_y_return,
    Window * child_return )
```

XUndefineCursor

Removes an association, defined previously with **XDefineCursor**, between a cursor and the given window.

```
void
XUndefinedCursor( Display * display, Window w )
```

XUngrabButton

Releases an existing passive grab made by the calling client of a mouse button or key combination on the given window.

```
void
XUngrabButton( Display * display, unsigned int button,
    unsigned int modifiers, Window w )
```

XUngrabKey

Releases a passive grab made by the calling client of the specified key or key combination on the given window.

void
*XUngrabKey(Display * display, unsigned int keycode,*
unsigned int modifiers, Window w)

XUngrabKeyboard
Releases an active grab of the keyboard made by the calling client.

void
*XUngrabKeyboard(Display * display, Time time)*

XUngrabPointer
Releases an active grab of the pointer made by the calling client.

void
*XUngrabPointer(Display * display, Time time)*

XUngrabServer
Releases the server from a grab.

void
*XUngrabServer(Display * display)*

XUninstallColormap
Uninstalls the specified colormap and, if necessary, installs the default map instead.

void
*XUninstallColormap(Display * display, Colormap cmap)*

XUnionRectWithRegion
Unions the given rectangle and source region to make the resulting destination region.

void
*XUnionRectWithRegion(XRectangle * rectangle, Region src_region,*
Region dest_region_return)

XUnionRegion
Unions the two specified regions and returns the resulting region.

void
XUnionRegion(Region sra, Region srb, Region dr_return)

XUniqueContext
Creates a new context resource ID.

XContext
XUniqueContext(void)

XUnloadFont
Unloads the given font.

void
*XUnloadFont(Display * display, Font font)*

XUnmapSubwindows
Unmaps all the mapped children of the given window in stacking order, ending with the top sibling.

void
*XUnmapSubwindows(Display * display, Window w)*

XUnmapWindow
Unmaps the given mapped, removing it and its children from the screen while it remains unmapped.

void
*XUnmapWindow(Display * display, Window w)*

XVisualIDFromVisual
Returns the VisualID of the given visual.

void
*XVisualIDFromVisual(Visual * visual)*

XWarpPointer
Moves the pointer 'instantaneously' from one defined point on the screen to another one.

void
*XWarpPointer(Display * display, Window src_w, Window dest_w, int src_x,*
int src_y, unsigned int src_width, unsinged int src_height, int dest_x,
int dest_y)

XWithdrawWindow
Unmaps a top-level window.

Status
*XWithdrawWindow(Display * display, Window w, int screen_number)*

XWindowEvent

Looks in the input event queue of the given window for the next event that matches the event mask, and then removes that event.

```
void
XWindowEvent( Display * display, Window w, long event_mask,
    XEvent * event_return )
```

XWMGeometry

Obtains the geomerty information for a given window.

```
int
XWMGeometry( Display * display, int screen_number,
    const char * user_geom, const char * default_geom,
    unsigned int_border_width, XSizeHints * size_hints, int * x_return,
    int * y_return, int * width_return, int * height_window,
    int * gravity_return )
```

XWriteBitmapFile

Writes the given bitmap to the named file.

```
void
XWriteBitmapFile( Display * display, const char * filename, Pixmap bitmap,
    unsigned int width, unsigned int height, int x_hot, int y_hot )
```

XXorRegion

Does an exclusive OR operation on the two source regions (by adding the total area of the two regions together and then removing the area where they overlap) to produce the resulting destination region.

```
void
XXorRegion( Region sra, Region srb, Region dr_return )
```

Appendix A

Bibliography

A full machine-readable version of the X reference manual can be obtained with the X System V11.4 release tape from The X Consortium at MIT, Cambridge, Mass., USA. Although this is 'free', it will take your printer, if you have one, many hours to print out. This is the definitive version of the documentation and forms the basis of all other X manuals.

A set of X manuals can be obtained from IXI Limited of Cambridge, England, who specialise in X for Unix-based workstations. You can contact IXI on (0223) 462131 (or the international dialing code + 44 223 462131 if you are telephoning from outside the UK).

For a good coverage of all the features of X, look out for the series known as *The Definitive Guides to the X Window System* by O'Reilly & Associates, Inc. This is an American publication and so it may not be readily available in all European bookshops.

Volume 0, X Protocol Reference Manual *for X Version 11*, Edited by Adrian Nye, ISBN 0-937175-40-4

Volume 1, Xlib Programming Manual, Adrian Nye, ISBN 0-937175-26-9

Volume 2, Xlib Reference Manual, Edited by Adrian Nye, ISBN 0-937175-27-7

Volume 3, X Window System User's Guide, Tim O'Reilly et. al., ISBN 0-937175-29-3

Volume 4, X Toolkit Intrinsics Programming Manual, Adrian Nye and Tim O'Reilly, ISBN 0-937175-56-0

Volume 5, X Toolkit Intrinsics Reference Manual, Edired by Tim O'Reilly, ISBN 0-937175-33-1

For a good tutorial introduction to X, look out for the *Introduction to the X Window System* by Oliver Jones, published by Prentice Hall (ISBN 0-13-499997-5).

Reference material about the C languge interface library (Xlib) and the X protocol specification is also provided by *The X Window System, C Library and Protocol Reference*, by R.W Sheifler et. al., published by Digital Press (ISBN 1-55558-012-2).

A good tutorial guide for X programmers is given by *X-Window Applications Programming*, by E.F Johnson and Kevin Johnson, published by MIS Press (ISBN 1-55828-016-2).

A tutorial and reference manual on the X toolkit is given by *The X Window system Programming and Applications with Xt*, by Douglas Young, published by Prentice Hall. (ISBN 0-13-497074-8).

The following is a more complete list, broken down by subject.

A.1 Motif

Berlage, Thomas, *OSF/Motif: Concepts and Programming*, Addison-Wesley, UK, 1991. ISBN 0-201-55792-4.

Johnson, Eric F. and Kevin Reichard, *Power Programming Motif*, MIS: Press, Portland, OR, 1991. ISBN 1-55828-059-6. Book with disk, ISBN 1-55828-061-8.

Nye, Adrian and Tim O'Reilly, *X Toolkit Intrinsics Programming Manual*, Motif Edition, O'Reilly and Assoc., Sebastopol, CA, 1991. ISBN 0-937175-62-5.

Open Software Foundation, *Application Environment Specification (AES): User Environment Volume, Rev. B*, Prentice Hall, Englewood Cliffs, NJ, 1991. ISBN 0-13-043530-9.

Open Software Foundation, *OSF/Motif Programmer's Reference, Revision 1.1*, Prentice Hall, Englewood Cliffs, NJ, 1991. ISBN 0-13-640681-5.

Open Software Foundation, *OSF/Motif Style Guide, Revision 1.1*, Prentice Hall, Englewood Cliffs, NJ, 1991. ISBN 0-13-640616-5.

Open Software Foundation, *OSF/Motif Programmer's Guide, Revision 1.1*, Prentice Hall, Englewood Cliffs, NJ, 1991. ISBN 0-13-640673-4.

Young, Douglas A., *The X Window System: Programming and Applications with Xt, OSF/Motif Edition*, Prentice Hall, Englewood Cliffs, NJ, 1990. ISBN 0-13-497074-8.

A.2 X Toolkit

Asente, Paul J. and Ralph R. Swick, *X Window System Toolkit*, Digital Press, Bedford, MA, 1990 (distributed by Prentice Hall). ISBN (Digital Press) 1-55558-051-3, (Prentice Hall) 0-13-972191-6.

Keller, Brian J., *A Practical Guide to X Window Programming*, CRC Press, 1990. ISBN 0-8493-7406-5.

McCormack, Joel, Paul Asente and Ralph R. Swick, *X Toolkit Intrinsics: C Language Interface, X11 Release 4 version*, 1989, MIT X Consortium. This document comes with the X Window System Release 4, from MIT.

Nye, Adrian and Tim O'Reilly, *X Toolkit Intrinsics Programming Manual*, O'Reilly and Assoc., Sebastopol, CA, 1990. ISBN 0-937175-33-1.

O'Reilly, Tim (editor), *X Toolkit Intrinsics Reference Manual*, O'Reilly and Assoc., Sebastopol, CA, 1990. ISBN 0-937175-35-8.

Smith, Jerry D., *Object-Oriented Programming with the X Window System Toolkits*, John Wiley, New York, NY, 1991. ISBN 0-471-53259-2.

A.3 X Library

Barkakati, Nabajyoti, *X Window System Programming*, SAMS, 1991. ISBN 0-672-22750-9.

Johnson, Eric F. and Kevin Reichard, *X Window Applications Programming*, MIS: Press, Portland, OR, 1989. ISBN 1-55828-016-2. Book with disk ISBN 1-55828-035-9.

Johnson, Eric F. and Kevin Reichard, *Advanced X Window Applications Programming*, MIS: Press, Portland, OR, 1990. ISBN 1-55828-029-4. Book with disk ISBN 1-55828-054-5.

Jones, Oliver, *Introduction to the X Window System, Prentice Hall*, Englewood Cliffs, NJ, 1989. ISBN 0-13-499997-5.

Nye, Adrian, *Xlib Programming Manual, vol. 1, 2nd ed.*, O'Reilly and Assoc., Sebastopol, CA, 1990. ISBN 0-937175-11-0.

Nye, Adrian (editor), *Xlib Reference Manual, vol. 2, 2nd ed.*, O'Reilly and Assoc., Sebastopol, CA, 1990. ISBN 0-937175-12-9.

Scheifler, Robert W. and James Gettys, with Jim Flowers, Ron Newman and David Rosenthal, 2nd ed., *X Window System: The Complete Reference to Xlib, X Protocol*, ICCCM, XLFD, Digital Press, Bedford, MA, 1990. ISBN (Digital Press) 1-5558-050-5, (Prentice Hall) 0-13-972050-2.

A.4 Open Look

AT&T, *UNIX System V Release 4 Programmer's Guide: OPEN LOOK Graphical User Interface*, Prentice Hall, Englewood Cliffs, NJ, 1989. ISBN 0-13-931908-5.

Heller, Dan, *XView Programming Manual*, O'Reilly and Assoc., Sebastopol, CA, 1990. ISBN 0-937175-52-8.

Miller, John David, *An OPEN LOOK at UNIX: A Developer's Guide to X*, M&T Books, 1990. ISBN 1-55-851057-5.

Sun Microsystems, *OPEN LOOK: Graphical User Interface Fucntional Specification*, Addison-Wesley, Reading, MA, 1990. ISBN 0-201-52365-5.

Sun Microsystems, *OPEN LOOK: Graphical User Interface Application Style Guidelines*, Addison-Wesley, Reading, MA, 1990. ISBN 0-201-52364-7.

A.5 Quick Reference Guides

Mikes, Steven, *X Window System Technical Reference*, Addison-Wesley, Reading, MA, 1990. ISBN 0-201-52370-1.

O'Reilly and Assoc., *The X Window System in a Nutshell*, O'Reilly and Associates, 1990. ISBN 0-937175-24-2.

Rost, Randi J., *X and Motif Quick Reference Guide*, Digital Press, Bedford, MA, 1990 (distributed by Prentice Hall). ISBN (Digital Press) 1-55558-052-1, (Prentice Hall) 0-13-972209-2.

Young, Douglas A., *OSF/Motif Reference Guide, Prentice Hall*, Englewood Cliffs, NJ, 1990. ISBN 0-13-642786-3.

A.6 X User Guides

Mansfield, Niall, *The X Window System: A User's Guide*, Addison-Wesley, Amsterdam, 1989. ISBN 0-201-51341-2.

Quercia, Valerie and Tim O'Reilly, *X Window System User's Guide*, O'Reilly and Assoc., 1990. ISBN 0-937175-14-5.

Index

READERS' COMMENTS

Distributed Software Limited welcomes your comments about all of its documentation. We would be particularly interested to hear your comments about this publication, and we hope that you will take the time to complete this section. Please write your comments in the space provided on the reverse of this sheet, and return this form (or a copy) to:

Distributed Software Limited,
The Maltings,
Charlton Road,
Shepton Mallet,
Somerset BA4 5QE.
United Kingdom.
Fax: 0749 344 977 (U.K.)
Fax: +44 749 344 977 (Worldwide)

Name: .....

Company: .....

Address: .....

.....

.....

Country: .....

Telephone: .....

Facsimile: .....

The X Window System Manual

YOUR COMMENTS:

FOR OFFICE USE ONLY

The X Window System Manual

Part number: DM5049