XOSD Guide and Reference

by Michael JasonSmith

XOSD Guide and Reference

by Michael JasonSmith

Copyright © 2003 Michael JasonSmith

Table of Contents

List of Figures

List of Tables

List of Examples

Part I. XOSD Guide

Table of Contents

Chapter 1. Introduction

A Potted History of XOSD

XOSD is a system that displays text on top of the other windows, much like the on-screen display (OSD) used by most modern televisions and video-players. In typical open-source style, it was written to scratch an itch.

André Renaud had just installed an infa-red sensor in his computer and was using a stereo remote-control to change what track XMMS was playing, but he could not see the track title from across the room. So André sat down and wrote XOSD to display track-names in XMMS. Over time features were added and XOSD became a stand-alone library, rather than *just* a plug-in for XMMS; eventually, XOSD 1.0 was released.

André's friend, Tim Wright, took over maintenance of XOSD when André's son (Maximilian) was born. (Being a Ph.D student meant that Tim had more time to spend on XOSD than André.) Tim created a simpler API, culminating in the 2.0 release. Patches are sent in (usually by the master bug-squasher, Philipp Matthias Hahn) and Tim keeps XOSD updated; currently XOSD is up to version 2.2.0.

While originally written to support an XMMS plug-in, other programs have adopted the XOSD library, including Hotkeys, and OSD Clock. There are also *bindings* for the Python programing language.

This document will act as a guide to the XOSD C-API. It will cover

- How to install XOSD,
- How to use the XOSD C-API, and
- How to hack the XOSD library.

Finally there is a function reference, which is a reproduction of the XOSD man pages.

1.1. Acknowledgements

Firstly, thanks to Tim Wright and André Renaud for writing the bulk of the XOSD library. They have both been patient in explaining how XOSD works to a simple documenter. Thanks must also go the the GNOME Documentation Project. While I mainly lurk on the `<gnome-doc-list@gnome.org>` mailing list, they always have helpful advice, even if I don't agree with their policy of keeping developer documents separate from user documents. *The GNOME Handbook of Writing Software Documentation* [http://developer.gnome.org/projects/gdp/handbook.html] and *The GNOME Documentation Style Guide* [http://developer.gnome.org/projects/gdp/styleguide.html] were invaluable resources in writing this guide. The sharp-eyed will note that the legal notice at the start of this document was based on the legal notice found in GNOME documents.

Chapter 2. Installing XOSD

There are two ways of installing XOSD: using pre-built packages (if you use the Debian™ or Mandrake®) or from source.

2.1. Installing from Packages

Debian™ (as of the `Sid` distribution, thanks to the work of Martijn van de Streek) and Mandrake® (thanks to Goetz Waschk) have pre-compiled packages available. Installing XOSD is fairly strait-forward if you are lucky enough to use one of these fine Linux distributions.

2.1.1. Installing Using Debian

Debian™ splits XOSD into four packages:

- `libxosd` contains the base library,
- `libxosd-dev` contains the header-file and man pages;
- `xosd-bin` contains the osd_cat binary; and
- `xmms-osd-plugin` contains the original XMMS plugin.

Use apt-get to install these packages.

2.1.2. Installing Using Mandrake

XOSD has been available in the `cotrib` section of Mandrake® since version 8.0. XOSD is split into three packages:

- `libxosd2` contains the base library;
- `libxosd2-devel` contains the header file and man pages; and
- `xosd` contains the XMMS plugin.

Use URPMI to install these packages.

2.2. Installing from Source

To install XOSD on an operating system that does not have pre-built packages, you will have to install from source. First you will have to install the packages that XOSD uses. It is a fairly small list:

- The X11 Windowing System (normally called X) and its associated libraries,
- Posix treads (`libpthread`),
- The standard C library (`libc`), and
- A compiler than understands C-99, such as GCC. [1]

[1] The only part of C-99 that is widely used throughout the code is the single-line comments, so most C compilers should be able to compile XOSD, even if they do not support all of C-99.

Almost all Unix variants [2] will have these packages installed already.

The XMMS plugin requires the development versions of the XMMS (version 1.2.7), `GTK+` (version 1.2), and `gdk-pixbuf` (version 0.21) libraries. The plugin will not be built if you do not have these libraries, but you can still use the XOSD library.

Start by downloading the tar archive of XOSD and unpack it. (You have probably done this already.) Next you have to build and install XOSD; the industrious efforts of Chris Weyl mean that this is fairly simple because Autoconf is used to control the building process. Just follow the commands shown in Example 2.1.

Example 2.1. Building XOSD from Source Using Autoconf

```
bash$ ./configure
bash$ make
bash$ make install
```

Note

You will have to be `root` to run **make install** unless you set your home directory as the base-directory of the install:

```
$ ./configure --prefix=$HOME \
   --with-plugindir=$HOME/.xmms/Plugins/General/
```

[2] Except for MacOS X®, which uses Aqua as its windowing system, rather than X.

Chapter 3. Using `libxosd`

The XOSD library is modelled on the `GTK+` libraries, in that it is an object-oriented library written in C. There are *constructor* functions, *getter* and *setter* functions, and *destructor* functions, just as you would find in a Java™ or Python module. In this chapter we will introduce XOSD by creating a small application that uses XOSD to write a message on the display.

3.1. Our First XOSD Program

Like all C programs, ours will begin with the basic *boiler-plate* code.

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

int main (int argc, char *argv[])
{
  xosd *osd;
```

In the above code there are two XOSD specific lines: the inclusion of the `xosd.h` header, and the declaration of the `osd` variable. The header file is needed so we can access all the functions and type definitions associated with XOSD, and `osd` will be used to store all information relating to the XOSD window.

Now lets create a XOSD window (or *instance*). The function `xosd_create` will create a new window, but we must know how big the window should be. Our new window will have two lines in its display, so we pass in `2` as the argument to `xosd_create`. After creating the window we check to see if it was created correctly.

```
  osd = xosd_create (2);
  if (osd == NULL)
    {
      perror ("Could not create \"osd\"");
      exit (1);
    }
```

Now we have to display something. The `xosd_display` function is the most complex of all the functions in the XOSD library. To display some data we have to say:

- The line to modify,
- The type of data to display, and
- The data.

We will display the string `You have been R00ted` on the first line of the display, so our code will look like the following.

```
  xosd_display (osd, 0, XOSD_string, "You have been R00ted");
```

Notice that `osd` was passed in as the first argument. All functions in the XOSD library (except for `xosd_create`) take a XOSD window as the first argument.

After displaying the string, all that is left to do is to wait for a few seconds before destroying the window (by calling the `xosd_destroy` function) and exiting the program.

```
  sleep (8);
  xosd_destroy (osd);
  exit (0);
}
```

3.1.1. Compiling

Compiling the newly created program is a strait-forward process thanks to the xosd-config command that generates the correct arguments to GCC. Using xosd-config is similar to using pkg-config, which is used to compile programs that use the GNOME libaries: the command is placed on the GCC command-line, as shown below.

```
bash$ gcc ‘xosd-config --cflags --libs‘ xosd_eg1.c -o xosd_eg1
```

In the above code the C-file `xosd_eg1.c` is compiled into the executable `xosd_eg1` that can be run from the command line.

```
bash$ ./xosd_eg1
```

Did you see the on-screen display? Try looking at the top-left of the display, where the words `You have been R00ted` will be written in small, green letters for eight seconds before the program finishes. We should make the text larger, which brings us to our next section!

3.2. Changing Attributes

In this section we will look at how to change the appearance and location of an XOSD window. We will start by changing the font so the text is more readable, before adding a shadow and changing the colour. Finally we will look at how to move the window to a new position.

3.2.1. Changing the Font

The default values for an XOSD object are close to unreadable. The reason for this is that the *only* font that can be guaranteed to be present in X-Windows is `fixed`, which is close to unreadable. Chances are that you have a myriad of fonts installed on your system. To change the font the `xosd_set_font` function is used. We could explicitly change the font to `fixed` in our example program by adding the following line *before* the call to `xosd_display`.

```
  xosd_set_font (osd, "fixed");
```

To change the font to something other than `fixed` we need to know the XLFD (X Logical Font Descriptor) of the font we wish to use. A program such as xfontsel(1) becomes very useful at this point. Start xfontsel and select a font. Click the select button and paste the text into the call to the `xosd_set_font` function call you added before. Your call to `xosd_set_font` should now look something like the following.

```
  xosd_set_font (osd, "-adobe-helvetica-bold-r-normal-*-*-320-*-*-p-*-iso8859-1");
```

The above code makes XOSD use Helvetica Bold at 32 points, which you probably have on your system. Chose a bold sans-serif font (such as Futura, Helvetica, Arial, or Avant Garde) whenever possible as the sans-serif fonts are more legible than their serifed counterparts (such as Times, Palatino, or Lucida Bright).[3]

After making the modifications to change the font, compile and run the example program again. The text will be much larger and easier to read.

3.2.2. Adding a Shadow

To increase readability (and because is looks good) a shadow can be added to the text. Figure 3.1 shows the example text without a shadow, while Figure 3.2 shows the same text with a shadow.

Figure 3.1. Example Text without a Shadow

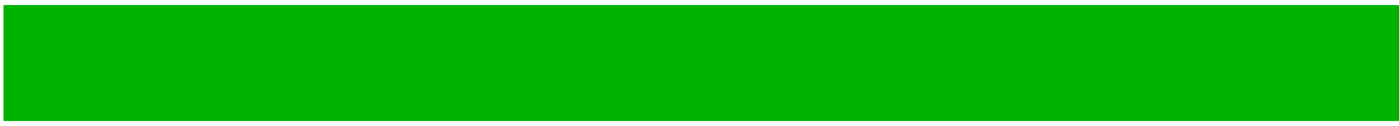
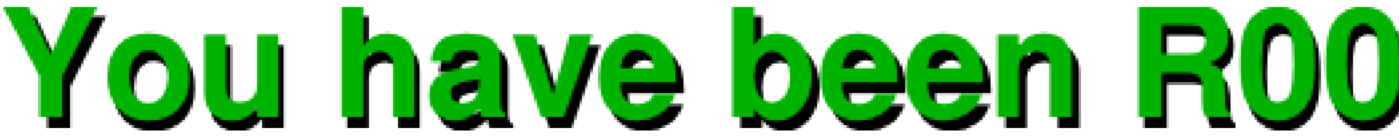


Figure 3.2. Example Text with a Shadow



The `xosd_set_shadow_offset` function is used to change the number of pixels the shadow is offset to the bottom-right of the main display. (Initially the shadow-offset is zero.) A value between one and four pixels looks good. The following code adds a four-pixel shadow to our example program; place the code just after the call to `xosd_set_font`.

```
  xosd_set_shadow_offset (osd, 4);
```

Once again, compile and run the example program. Experiment with the size of the shadow.

3.2.3. Changing the Colour

Spelling

As XOSD is mainly developed in New Zealand, where English is spoken and written, the API uses English, rather than American. Therefore *colour* is written correctly, with a *u*.

The next the visual attributes that we can change is the colour of the display, which is `green` by default. To do this we use the `xosd_set_colour` function. Colours can be specified by name, or hexadecimal colour specifier.

The valid X11® colour-names are stored in the file `rgb.txt`; from the command line type the following to find where `rgb.txt` is stored.

```
bash$ locate rgb.txt
```

[3]Serifed fonts are more *readable* than sans-serif fonts, so serifed fonts should be used for a long body of text (such as this paragraph) while sans-serif should be used for short bodies of text (such as a heading) because it is easier to dsitinguish each letter.

The colour name is passed as a string to the `xosd_set_colour` For example, the following code makes the text displayed in the XOSD window `lawngreen`, which is close to the standard colour for television on-screen displays.

```
xosd_set_colour (osd, "lawngreen");
```

Add the above code to your program, putting it before the call to `xosd_display` and see the difference.

Hexadecimal colour specifiers are similar to those used in HTML. They are made up of three pairs of characters (a *hex triplet*) with a `#` at the start. Rather than explain how hexadecimal colours work, it is easier to use a GUI tool to create the triplet for you.

Start the GIMP drawing program, and double-click on the colour-swab at the bottom of the Toolbox window. The colour selector window will appear. Change the colour, using whatever method you like, select the Hex Triplet, and paste it into you code, replacing the colour name. Put a `#` at the start of the number and you are done! Your code will look something like the following, which makes the XOSD window use the light-purple which is often used in GNOME icons.

```
xosd_set_colour (osd, "#494066");
```

3.2.4. Changing the Outline Colour

The final visual aspect of the window that we can change is the outline colour of the display, so the border of the text or graphics is different to the rest of the display. This final tweak makes the display easier to read, if you get the colours right! Generally, the outline colour is black, while the fill-colour is light. Place the following example code to your program just after the call to `xosd_set_colour`. The call to `xosd_set_outline_offset` sets the outline to be two pixels wide, while the `xosd_set_outline_colour` call sets the outline colour to black.

```
xosd_set_outline_offset(osd, 2);
xosd_set_outline_colour(osd, "black");
```

3.2.5. Changing the Position of the XOSD Window

The XOSD window can be placed in three and vertical three horizontal positions, with minor adjustments to fine-tune the placement.

3.2.5.1. Changing the Vertical Position

There are three possible vertical positions for an XOSD window (Table 3.1). By default the window is placed at the top of the display, but `xosd_set_pos` function can be called to change this.

Table 3.1. Vertical Positions Declared in the xosd_pos Enumeration

`XOSD_top`	The top of the display.
`XOSD_middle`	The middle of the display.
`XOSD_bottom`	The bottom of the display

The `xosd_set_pos` function takes two arguments: the window to act on, and a position. The position is specified using one of the values from the xosd_pos enumeration, which are listed in Table 3.1. For example, the following code places the XOSD window at the bottom of the display.

```
xosd_set_pos (osd, XOSD_bottom);
```

Most desktop systems have a panel at the bottom of the display. When the XOSD display covers the panel the text in the XOSD window can be hard to read. To overcome this we can move the window up slightly by calling the `xosd_set_vertical_offset` function that moves the window by a small number of pixels. The direction that the `xosd_set_vertical_offset` moves the XOSD window depends on its position: if the window is positioned at the top of the display (`XOSD_top`) then the window is moved down, while the window is moved up if it positioned at the bottom of the display (`XOSD_bottom`). In our example we will move the XOSD window *up* by 48 pixels, which should be enough to avoid most panels.

```
  xosd_set_vertical_offset (osd, 48);
```

3.2.5.2. Changing the Horizontal Alignment

Often it looks better if text is aligned to the right of the display, rather than being shown on the left. To do this we use the `xosd_set_align` function to align the XOSD window to the left, centre, or right of the display. The enumeration xosd_align declares the three constants used to specify the horizontal alignment of the window (Table 3.2).

Table 3.2. Horizontal Positions Declared in the xosd_align Enumeration

`XOSD_left`	The left of the display.
`XOSD_center`	The center of the display.
`XOSD_right`	The right of the display.

Fine control of horizontal placement is performed using the `xosd_set_horizontal_offset` function, which alters the number of pixels the display is offset from the left or right of the display (depending which is closer).

Add the following code to our small program, just after the call to the `xosd_set_vertical_offset` function, and before the call to the `xosd_display` function. It will move the XOSD window to the right, with a gap of 48 pixels.

```
xosd_set_align (osd, XOSD_right);
  xosd_set_horizontal_offset (osd, 48);
```

3.2.6. Conclusion

That is about all the functions you need to alter the appearance and location of an XOSD window. Your example C file will look similar to Example 3.1.

Example 3.1. A Simple XOSD Program

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

int main (int argc, char *argv[])
{
  xosd *osd;

  osd = xosd_create (2);
  if (osd == NULL)
    {
      perror ("Could not create \"osd\"");
      exit (1);
    }

  xosd_set_font (osd, "-adobe-helvetica-bold-r-normal-*-*-320-*-*-p-*-iso8859-1");
  xosd_set_shadow_offset (osd, 2);
  xosd_set_colour (osd, "#494066");
  xosd_set_outline_offset(osd, 2);
  xosd_set_outline_colour(osd, "black");

  xosd_set_pos (osd, XOSD_bottom);
  xosd_set_vertical_offset (osd, 48);

  xosd_set_align (osd, XOSD_right);
  xosd_set_horizontal_offset (osd, 48);

  xosd_display (osd, 0, XOSD_string, "You have been R00ted");

  sleep (8);
  xosd_destroy (osd);
  exit (0);
}
```

3.3. Timeouts

In our previous example we displayed the text and then removed it from the display when the program finished. In this section we will cover how to remove an XOSD window from the display without destroying it or stopping the program. We begin by explicitly hiding the display (Section 3.3.1, “Explicitly Hiding a Display”) before covering the XOSD timeout mechanism (Section 3.3.3, “Using Timeouts”).

3.3.1. Explicitly Hiding a Display

The `xosd_hide` is used to explicitly make an XOSD window invisible (*unmapped* in X11 terminology). It is a simple function to call, as it only takes the XOSD window to act on as an argument. For example we could replace the last two function calls in our previous example with the following code that hides the window, prints out some text, and waits for three seconds before exiting.

```
  xosd_hide (osd);
  printf ("Up to here!\n");
  sleep (3);
  xosd_destroy (osd);
  exit (0);
```

3.3.2. Redisplaying a Window

To redisplay a window that has been hidden we use the `xosd_show` function. To illustrate how `xosd_show` and `xosd_hide` can work together we will create a function that makes an XOSD window blink (Example 3.2). The function takes three arguments: the XOSD window, the number of times to blink, and the delay between blinks. The body of the function goes through a for-loop showing and hiding the display before exiting.

Example 3.2. A Function to Make an XOSD Window Blink

```
void xosd_blink (xosd *osd, int blink_num, int blink_delay)
{
  int i = 0;

  for (i = blink_num; i >= 0; i--)
    {
      xosd_show (osd);
      sleep (blink_delay);
      xosd_hide (osd);
      sleep (blink_delay);
    }

  return;
}
```

For the function listed in Example 3.2 to work correctly `xosd_display` must be called first. It is left as an exercise to the reader to rewrite the function so it takes the string to display as an argument and calls `xosd_display` on its own accord.

3.3.3. Using Timeouts

XOSD windows can automatically hide themselves, rather than the controlling program explicitly calling the `xosd_hide` function. To do this, set the windows timeout value by calling the `xosd_set_timeout` function. For example, the following causes the display to be hidden two seconds before the `sleep` function returns.

```
  xosd_set_timeout(osd, 6);
  xosd_display (osd, 0, XOSD_string, "You have been R00ted");
  sleep (8);
```

You may want to display some information, perform a few functions, then ensure the XOSD window is hidden before carrying on. The `xosd_wait_until_no_display` function can be used for just this purpose. It causes execution of the calling program to be suspended until the window has been removed from the display. For example, the following code displays some text, waits for two seconds, prints some data out, and then waits for the display to become hidden.

```
  xosd_set_timeout(osd, 6);
  xosd_display (osd, 0, XOSD_string, "R00ting Machine");
  sleep(2);
  fprintf(stderr, "Hacking away...\n");
  fprintf(stderr, "Just assume this can take a\n");
  fprintf(stderr, "variable amount of time.\n");
  xosd_wait_until_no_display (osd);
```

3.4. Error Handling

Errors in XOSD are handled in a very similar way to errors in the base C libraries. All XOSD functions, except for `xosd_create`, return an integer value; if the value is `-1` an error occurred. For example, the following code checks to see if the call to the `xosd_show` function was successful, and calls `exit` if it was not.

```
if (xosd_show(osd) == -1)
  {
    /* An error occurred */
    exit(1);
  }
```

To know *why* the called failed you need to print out the value of the `xosd_error` variable, as is done in the following example.

```
if (xosd_show(osd) == -1)
  {
    /* An error occurred */
    fprintf(stderr, "Could not show XOSD window: %s\n", xosd_error);
    exit(1);
  }
```

The `xosd_create` function returns a pointer rather than an integer. To indicate an error the pointer returned by `xosd_create` is set to `NULL`. The first example in Section 3.1, “Our First XOSD Program” shows how to check if a call to `xosd_create` was successful.

3.5. Displaying Interger Values

Previously we only looked at how to display simple strings. In this section we will look at the two methods for displaying integer values: percentage bars (Section 3.5.1, “Percentage Bars”) and sliders (Section 3.5.2, “Sliders”). Both techniques can only display values between `0` and `100`. The XOSD window will only display `100` if a value greater than `100` is passed; similarly a value less than `0` will cause `0` to be displayed.

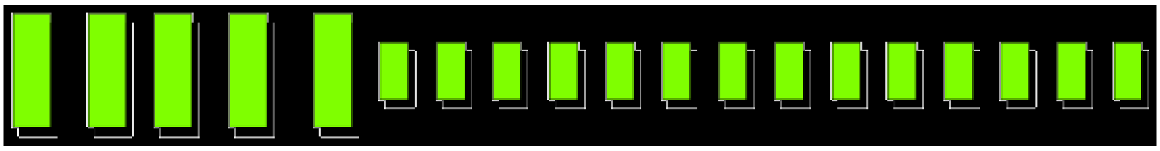
3.5.1. Percentage Bars

A percentage bar is similar to the progress bar seen in many applications. It is made up of two parts (Figure 3.3):

1. A sequence of dashes that indicate the total length of the percentage bar, and
2. A sequence of pipes that show the current value.

Less pipes indicate a smaller value. You cannot set the total number of pipes and dashes that are displayed as it is dependent on the size the display.

Figure 3.3. Percentage Bar Example



The sequence in pipes (on the left) show the current value of the percentage bar, while the sequence of dashes shows the total length.

The `xosd_display` is called to display a percentage bar. However, the arguments passed to `xosd_display` are different to what we have seen earlier (Section 3.1, “Our First XOSD Program”). For example, the following displays the value `77` using a percentage bar.

```
xosd_display (osd, 0, XOSD_percentage, 77);
```

The first argument in the above example is the XOSD window to use, as normal. However, instead of passing `XOSD_string` as the value for *`command`* the literal `XOSD_percentage` is passed, followed by the value to display.

3.5.2. Sliders

Sliders were developed to display the *balance* for the XMMS plugin. A slider represents an interger value, like the percentage bar, but the value is shown as a *point* rather than a bar. Figure 3.4 is an example of a slider showing a small value.

To display a slider, call `xosd_display` with `XOSD_slider` as the second argument, such as the following example.

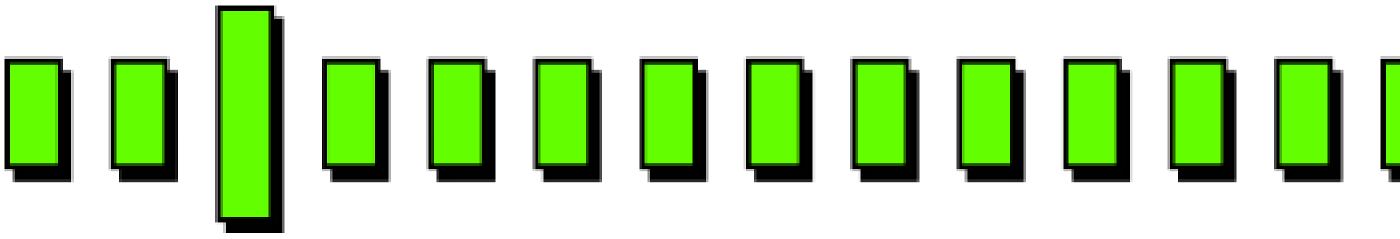
```
xosd_display (osd, 0, XOSD_percentage, 77);
```

The third argument is the distance along the bar the slider should appear, expressed as a percentage. So in the above example the slider appears 77% along the bar, which is near the right of the bar for left-aligned XOSD windows, or near the left for right-aligned windows.

3.5.3. Changing the Size of the Bar

Normally you will not have to change the length of a percentage bar or slider, as the default size that XOSD uses is sensible. However, there are two functions to call if you do want to change the bar dimensions.

Figure 3.4. Slider Example



The current value of the slider is shown by the pipe near the left, while the blocks show other possible values for the slider.

The first is `xosd_set_bar_length`, which changes the horizontal length of the bar. It takes one augment, besides the XOSD window: the width of the display that the bar should take up. For example, if we wanted our slider from earlier (Section 3.5.2, “Sliders”) to only take up half the screen we would call `xosd_set_bar_length` like the following.

```
xosd_set_bar_length(osd, 50);
```

The vertical size of the bar is dependent on the size of the font used to display the text; see Section 3.2.1, “Changing the Font” for an example on how to make the font larger.

3.5.4. Bar Example

The following example shows how a progress bar can be implemented using a percentage bar. Most of the code is the same as the previous example as it creates the window and set the colour, size and position. The core part of the code runs through a loop which displays an ever-increasing value in the XOSD window.

Example 3.3. A Progress Bar in XOSD

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

int main (int argc, char *argv[])
{
  xosd *osd = NULL;
  int i = 0;
  char *font = "-adobe-helvetica-bold-r-normal-*-*-320-*-*-p-*-iso8859-1"
  osd = xosd_create (2);
  if (osd == NULL)
    {
      perror ("Could not create \"osd\"");
      exit (1);
    }

  xosd_set_font (osd, font);
  xosd_set_shadow_offset (osd, 2);
  xosd_set_colour (osd, "lawngreen");

  xosd_set_pos (osd, XOSD_bottom);
  xosd_set_vertical_offset (osd, 48);

  xosd_set_align (osd, XOSD_left);
  xosd_set_horizontal_offset (osd, 48);
  xosd_set_bar_length(osd, 50);
  for (i = 0; i <= 100; i += 5)
    {
      xosd_display (osd, 0, XOSD_percentage, i);
      sleep(1);
    }
  xosd_destroy (osd);
  exit (0);
```

3.6. Multi-Line Output

In all our previous examples, we only used a single line to display data, even though the XOSD window was big enough to display two lines. In this section we will display text and percentage bars on multiple lines, and scroll the window.

3.6.1. Displaying Two Lines of Data

The number of lines that can be displayed in an XOSD window is set when the window is created by `xosd_create` and the the line to use is specified by `xosd_display`. The lines are numbered from the topmost line (line 0), so to display two lines of text in a two-line window we would use the following code.

```
xosd_display(osd, 0, XOSD_string, "You have been R00t3d");
xosd_display(osd, 1, XOSD_string, "By an l33t h4x0r");
```

We could also display a percentage bar on the bottom line (line 1) as the XMMS plugin does when the volume changes. The following code *fragment* combines the progress bar example with a single line of text, which is then replaced when bar reaches 100%.

Example 3.4. A Two Line Window in XOSD

```
xosd_display(osd, 0, XOSD_string, "Being R00t3d");
for (i = 0; i <= 100; i += 5)
  {
     xosd_display (osd, 1, XOSD_percentage, i);
     sleep(1);
  }
xosd_display(osd, 0, XOSD_string, "You have been R00t3d");
xosd_display(osd, 1, XOSD_string, "By an l33t h4x0r");
sleep(2);
```

3.6.2. Scrolling

Rather than simply replacing what is displayed on each line, as we did in the previous example, we can scroll the window, using `xosd_scroll`. This is useful when you have a large amount of data to display but do not want to create a large window or use a small font. The following example extends the previous example by scrolling the window to blank the display.

Example 3.5. Scrolling in XOSD

```
xosd_display(osd, 0, XOSD_string, "Being R00t3d");
for (i = 0; i <= 100; i += 5)
   {
      xosd_display (osd, 1, XOSD_percentage, i);
      sleep(1);
   }

xosd_scroll(osd, 1);
```

```
sleep(1);
xosd_scroll(osd, 1);
sleep(1);

xosd_display(osd, 1, XOSD_string, "You have been R00t3d");
sleep(1);
xosd_scroll(osd, 1);
sleep(1);
xosd_display(osd, 1, XOSD_string, "By an l33t h4x0r");
sleep(1);
xosd_scroll(osd, 1);
sleep(1);
xosd_scroll(osd, 1);
sleep(1);
xosd_display(osd, 1, XOSD_string, "Long live script k1dd3z");
sleep(1);
xosd_scroll(osd, 1);
sleep(1);
xosd_scroll(osd, 1);
sleep(1);
```

At this point the display contains a text-line at the top, and a percentage bar at the bottom.
The display is blank at this point.
The display contains two text lines at this point, which are then scrolled up and replaced by a single line.

Chapter 4. Hacking `libxosd`

This guide will give you a brief tour of the internals of the XOSD library. It will look at how XOSD is designed and the coding standards that the main authors use. To understand how XOSD works internally you will need a good knowledge of POSIX threading (`pthread`), and a general knowlege of how X11 works. Almost all the other libaries used in XOSD are part of ANSI-C.

4.1. Architecture of XOSD

How XOSD Works

All of the XOSD library is implemented in a single C-file: `xosd.c` — roughly 40 functions, two type definitions, and a struct. You can think of the struct, xosd, as an object that has all its atributes as private, and all the functions in `xosd.c` as methods; The functions that begin with `xosd_` are public methods, while the other functions are *private*.

If you look at the xosd structure (Figure 4.1) you will see that the first five fields are concerned with threading. Each XOSD window consists of two threads. The first, `event_thread`, runs the function `event_loop` that recieves X11 events, and the second, `timeout_thread`, runs the function `timeout_loop` that hides the window when timeouts expire.[4] A *mutex* is used so only one thread ever modifies the fields in the xosd structure. To use the mutex, each function in `xosd.c` has a call to `pthread_mutex_lock` near the top of the function and call `pthread_mutex_unlock` near the end. Any code you add to `xosd.c` will also have to use the mutex, otherwise *race conditions* can occour.

The next ten fields in xosd store information about how the XOSD window is displayed. XOSD does not use any GUI toolkits, such as GTK+ or Qt, instead it calls the basic X11 functions for drawing text and graphics. The following eleven fields (from `width` to `bar_length`) store information about the window that are set by the various `xosd_set_` functions.

The two boolean fields, `mapped` and `done`, have quite different functions. The `mapped` field is set to true if the display is shown. On the other hand, the `done` field is only set to true when `xosd_destroy` is called as it causes the event and timeout threads to exit.

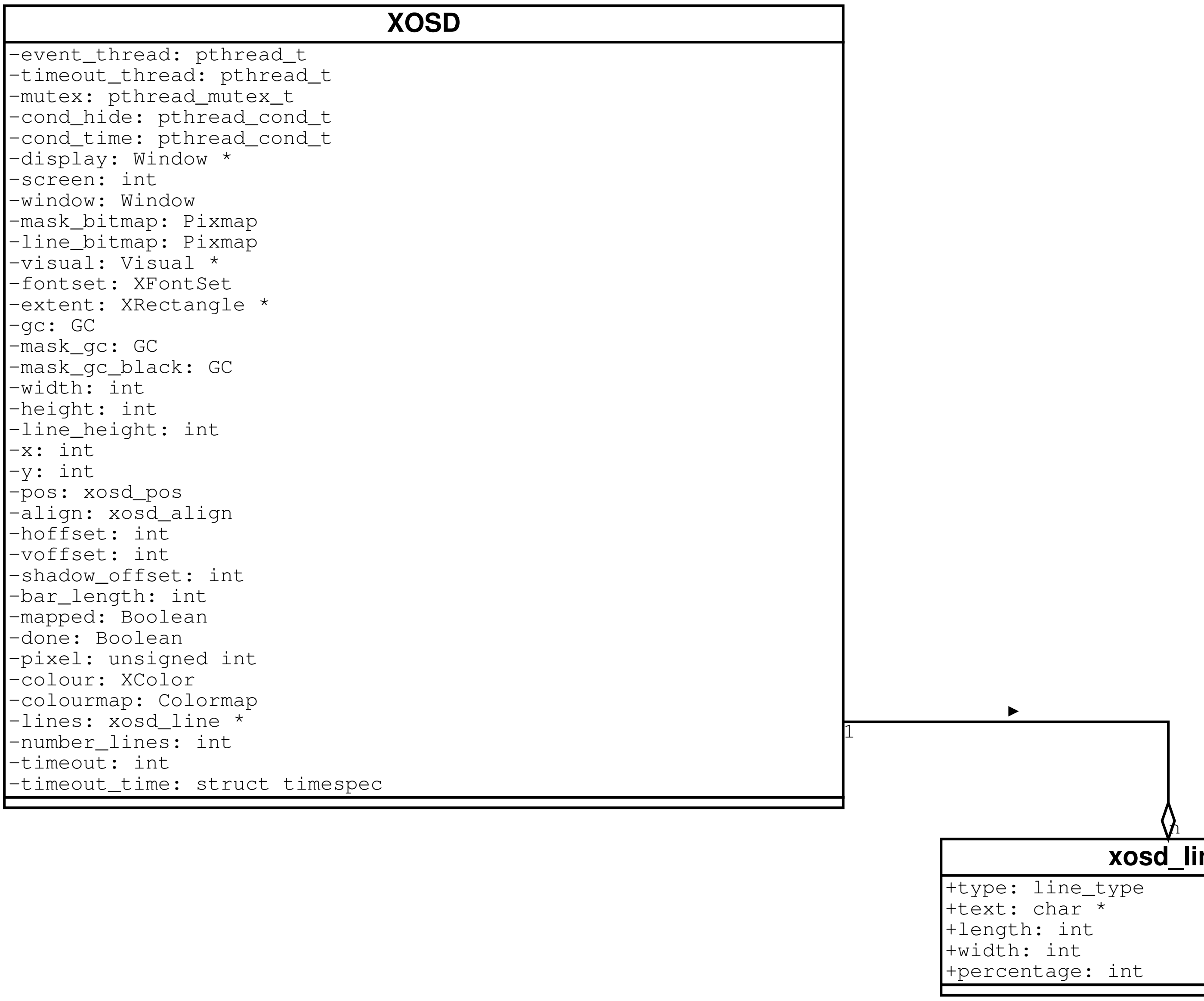
Colours are controlled by the three fields: `pixel`, `colour` and `Colourmap`. The astute reader will now realise why `xosd_set_colour` changes the colour of all the text and graphics displayed in the window, and why lines in the same window cannot have different colours.

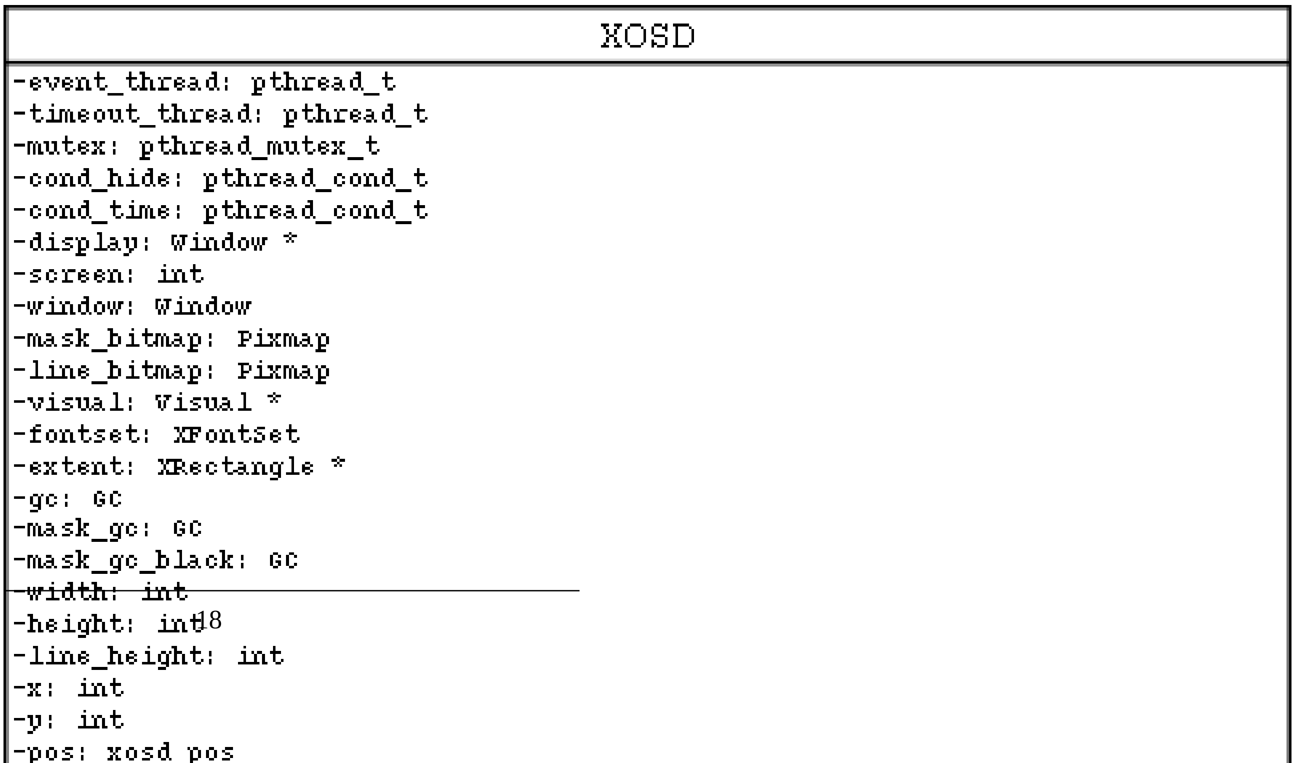
The content of the display is contained in the `lines` field, which is allocated when the window is created. There are five fields associated with each line:

- The type of the line,
- The text displayed by the line (if any),
- The length of the text,
- The width of the bar, and
- The percentage displayed by the line (if any).

[4] There are at least three threads in any program that uses `libxosd` as the main-program also runs in a seperate thread.

Figure 4.1. A UML Representation of the XOSD Structure





Not all of these fields are used at any one time, as a line can *only* display text, a percentage bar, or a slider.

The final two filds are used to control the timeout values. The number of seconds untill the window is unmapped from the display is controlled by the `timeout` field, while `timeout_time` is to set the date when the window will be unmapped (as required by `pthread_cond_timedwait` in the `timeout_loop` thread).

4.2. Coding Standards

There are a few standards that the XOSD authors (Tim and André) follow; for the sake of readability it is best if any new code follows these standards.

- Use the C-99 standard when writing code, except in the `xosd.h` header-file, which should use Ansi-C (C-89). While GCC does not implement all of C-99, it implements the useful features, such as single line comments (`//`).
- Use single-line comments (`//`) in preference to multi-line comments (`/* */`), *except* in the `xosd.h` header-file, which should use multi-line comments only.
- Put spaces between function names and the parameters that follow.
- Public functions must begin with `xosd_` and have a forward-declaration in the `xosd.h` header-file; all functions that are private must not begin with `xosd_`.
- Public constants must begin with `XOSD_` and are declared in the `xosd.h` header-file. Numeric constants are declared as part of an enumerated type-definition (`typedef enum`) the names of which are formatted like functions.
- The xosd structure shall remain hidden, and all modifications to fields shall be done through setter functions.
- Document functions added to the `xosd.h` header-file should have a brief man-page like comment written just before the function declaration. The comment should contain the following.
  - The function name, along with the purpose of the function;
  - A list of the function arguments, titled `ARGUMENTS`, with an explanation of each argument; and
  - The return value for the function, titled `RETURNS`, with an explanation on how the return value should be interpreted.
- When testing code using GCC, turn all warnings on (`-Wall`).

Part II. Function Reference

The XOSD library provides an object-oriented C API. The style of the functions is similar to those found in the `GTK+` libraries, with constructor and destructor functions, functions to get and set attributes, and function to controll the display of data.

Table of Contents

Creating and Destroying

New XOSD windows are created by the `xosd_create` function, which can be thought of as the *constructor* method for an XOSD window. The equivalent destructor method is `xosd_destroy` function, which frees all memory used by the XOSD window.

Name

`xosd_create` -- Create a new XOSD window

`xosd_create`

Synopsis

#include <xosd.h>

```
xosd *xosd_create(int number_lines);
```

Description

`xosd_init` creates a new xosd window that can be used to display textual or numerical data on a X11 display in a unmanaged, shaped window that appears to be transparent. It provides a similar effect to the on-screen display of many televisions and video recorders.

By default the XOSD window is placed at the top-left of the display, with the horizontal and vertical offsets set to zero. Text is displayed in `green`, using the `fixed` font by default. These values can be changed by using one of the setter methods listed in the See Also section below.

Arguments

`number_lines`	The maximum number of lines of text that the window can display. This value cannot be changed.

Return Value

On success a pointer to a new `xosd` window is returned, otherwise `NULL` is returned.

Environment

`DISPLAY`	The environment variable that determines which X11 display the XOSD window appears.
`char *xosd_error`	A string describing the error, if one occurred.

History

The `xosd_create` function first appeared in version 2.0 of the XOSD library, replacing the now depricated `xosd_init` function.

Authors

The XOSD library was originally written by André Renaud, and is currenly maintained by Tim Wright. This document was written by Michael JasonSmith.

Bugs

There are no known bugs with `xosd_create`. Bug reports can be sent to `<xosd@ignavus.net>`.

See Also

`xosd_destroy`(3xosd), `xosd_display`(3xosd), `xosd_show`(3xosd), `xosd_hide`(3xosd), `xosd_set_pos`(3xosd), `xosd_set_align`(3xosd), `xosd_set_shadow_offset`(3xosd), `xosd_set_vertical_offset`(3xosd), xosd_set_horizontal_offset(3xosd), `xosd_set_timeout`(3xosd), `xosd_set_colour`(3xosd), `xosd_get_colour`(3xosd), `xosd_set_font`(3xosd), `xosd_scroll`(3xosd),

Name

xosd_destroy -- Destroy an XOSD window

xosd_destroy

Synopsis

#include <xosd.h>

```
int xosd_destroy(xosd *osd);
```

Description

xosd_destroy destroys an XOSD object freeing up any memory used. The *osd* object must have been created by xosd_create(3xosd).

Arguments

osd	The object to destroy.

Return Value

On success 0 is returned, otherwise -1 is returned and xosd_error is set to indicate the reason for the error.

Environment

char *xosd_error	A string to a text string describing the error, if one occurred.

History

The xosd_destroy function first appeared in version 2.0 of the XOSD library, replacing the now depricated xosd_uninit function.

Authors

The XOSD library was originally written by André Renaud, and is currenly maintained by Tim Wright. This document was written by Michael JasonSmith.

Bugs

There are no known bugs with xosd_destroy. Bug reports can be sent to <xosd@ignavus.net>.

See Also

xosd_create(3xosd)

Getting and Setting Attributes

There are eleven attributes of an XOSD window that can altered:

- The vertical position,
- The vertical offset,
- The horizontal alignment,
- The horizontal indent,
- The size of the shadow,
- The size of the outline,
- The colour of the outline,
- The colour used to display data,
- The font used to display text,
- The length of percentage bars, and
- How long the XOSD window remains visible.

Name

`xosd_set_pos` -- Change the vertical position of the XOSD window

`xosd_set_pos`

Synopsis

#include <xosd.h>

```
int xosd_set_pos(xosd *osd, xosd_pos pos);
```

Description

`xosd_set_pos` changes the vertical position of the XOSD window. There are three possible vertical positions for the XOSD window, corresponding to the top, middle or bottom of the display. By default the XOSD window is positioned at the top of the display.

Arguments

`osd`	The XOSD window to change.
`pos`	The position of the display. The value of *`pos`* can be one of the following three values.

- `XOSD_top` for the top (the default),
- `XOSD_middle` for the middle, or
- `XOSD_bottom` for the bottom of the display.

Return Value

On success, a zero is returned. On error, -1 is returned and `xosd_error` is set to indicate the reason for the error.

Environment

`char *xosd_error`	A string describing the error, if one occurred.
enum xosd_pos	The possible vertical positions of the display, defined as an enumerated type. There are three values defined:

- `XOSD_top`,
- `XOSD_bottom`, and
- `XOSD_middle`.

Authors

The XOSD library was originally written by André Renaud, and is currently maintained by Tim Wright. This document was written by Michael JasonSmith.

Bugs

The xosd_pos enumeration defines the three constants in an illogical order, with `XOSD_middle` at the end. Bug reports can be sent to `<xosd@ignavus.net>`.

See Also

`xosd_create`(3xosd), `xosd_set_align`(3xosd)

Name

`xosd_set_align` -- Change the horizontal position of the XOSD window

`xosd_set_align`

Synopsis

#include <xosd.h>

```
int xosd_set_align(xosd *osd, xosd_align align);
```

Description

`xosd_set_align` changes the horizontal alignment of the XOSD window. There are three possible horizontal positions for the XOSD window, corresponding to the top, middle, or bottom of the display. By default the XOSD window is aligned at the left of the display.

Arguments

`osd`	The XOSD window to change.
`align`	The alignment of the display. There are three possible values for *`align`*

- `XOSD_left` for the left (the default),
- `XOSD_center` for the center, and
- `XOSD_right` for the right of the display.

Return Value

On success, a zero is returned. On error, -1 is returned and `xosd_error` is set to indicate the reason for the error.

Environment

`char *xosd_error`	A string describing the error, if one occurred.
enum xosd_align	The possible horizontal alignment of the display, defined as an enumerated type. There are three values defined:

- `XOSD_left`,
- `XOSD_center`, and
- `XOSD_right`.

Authors

The XOSD library was originally written by André Renaud, and is currently maintained by Tim Wright. This document was written by Michael JasonSmith.

Bugs

There are no known bugs with `xosd_set_align`. Bug reports can be sent to `<xosd@ignavus.net>`.

See Also

`xosd_create`(3xosd), `xosd_set_pos`(3xosd)

Name

`xosd_set_vertical_offset` -- Change the vertical offset of the XOSD window

`xosd_set_vertical_offset`

Synopsis

#include <xosd.h>

```
int xosd_set_vertical_offset(xosd *osd, int offset);
```

Description

`xosd_set_vertical_offset` changes the number of pixels the XOSD window is offset from the top or bottom of the display. This is done to avoid desktop-panels, such as those provided by GNOME or KDE. By default the vertical offset is zero; a value of 48 will generally avoid most panels.

The direction the XOSD window is offset depends on the vertical positioning of the window (set by calling `xosd_set_pos`(3xosd))

- If the XOSD window is at the top of the display (`XOSD_top`) then `xosd_set_vertical_offset` moves the window *down* by *`offset`* pixels.
- If the XOSD window is at the bottom of the display (`XOSD_bottom`) then `xosd_set_vertical_offset` moves the window *up* by *`offset`* pixels.

Arguments

`osd`	The XOSD window to change.
`offset`	The number of pixels the XOSD window is offset from the top of bottom of the display.

Return Value

On success, a zero is returned. On error, -1 is returned and `xosd_error` is set to indicate the reason for the error.

Environment

`char *xosd_error`	A string describing the error, if one occurred.
`xosd_set_pos`(3xosd)	The `xosd_set_pos`(3xosd) function is used to change the vertical position of the XOSD window. This, in turn, affects the how `xosd_set_vertical_offset` works.

Authors

The XOSD library was originally written by André Renaud, and is currently maintained by Tim Wright. This document was written by Michael JasonSmith.

Bugs

If the vertical position of the XOSD window is set to the middle of the display (`XOSD_middle`) then `xosd_set_vertical_offset` behaves strangely. Bug reports can be sent to `<xosd@ignavus.net>`.

See Also

`xosd_create`(3xosd), `xosd_set_pos`(3xosd), `xosd_set_horizontal_offset`(3xosd)

Name

xosd_set_horizontal_offset -- Change the horizontal offset of the XOSD window

xosd_set_horizontal_offset

Synopsis

#include <xosd.h>

```
int xosd_set_horizontal_offset(xosd *osd, int indent);
```

Description

xosd_set_horizontal_offset changes the number of pixels the XOSD window is indented to the right. This is done to avoid desktop-panels, such as those provided by GNOME or KDE. By default the horizontal indent is zero; a value of 48 will generally avoid most panels.

Arguments

osd	The XOSD window to change.
indent	The number of pixels the XOSD window is indented to the right.

Return Value

On success, a zero is returned. On error, -1 is returned and xosd_error is set to indicate the reason for the error.

Environment

char *xosd_error	A string describing the error, if one occurred.

History

The xosd_set_horizontal_offset function first appeared in version 2.0 of the XOSD library.

Authors

xosd_set_horizontal_offset was written by Etan Reisner. The XOSD library was originally written by André Renaud, and is currently maintained by Tim Wright. This document was written by Michael JasonSmith.

Bugs

Currently xosd_set_horizontal_offset does not move the XOSD window *left* when the window is aligned to the right (XOSD_right) of the display. Bug reports can be sent to <xosd@ignavus.net>.

See Also

xosd_create(3xosd), xosd_set_align(3xosd), xosd_set_vertical_offset(3xosd)

Name

`xosd_set_colour` -- Change the colour of the data displayed the XOSD window

`xosd_set_colour`

Synopsis

#include <xosd.h>

```
int xosd_set_colour(xosd *osd, const char *colour);
```

Description

`xosd_set_colour` changes the colour of all the text and graphics displayed in the XOSD window. By default the colour is `"green"`. `xosd_set_colour` will change the colour of the text and graphics to white if it could not change to the specified colour.

Arguments

`osd`	The XOSD window to query.
`colour`	The colour of the text and graphics displayed in the XOSD window. The colour can be specified by name, such as `"green"`, or by a hexadecimal colour specifier, such as `"#5eff72"`. (For a complete list of valid colour names see the file `rgb.txt` in you X11 distribution.)

Return Value

On success a zero is returned. On error -1 is returned and `xosd_error` is set to indicate the reason for the error.

Environment

`char *xosd_error`	A string describing the error, if one occurred.
`rgb.txt`	A list of all valid colour names. It is normally distributed as part of X11.

Authors

The XOSD library was originally written by André Renaud, and is currently maintained by Tim Wright. This document was written by Michael JasonSmith.

Bugs

There are no known bugs with `xosd_set_colour`. Bug reports can be sent to `<xosd@ignavus.net>`.

See Also

`xosd_create`(3xosd), `xosd_get_colour`(3xosd), `xosd_set_outline_colour`(3xosd).

Name

`xosd_get_colour` -- Get the colour of the data displayed the XOSD window

`xosd_get_colour`

Synopsis

#include <xosd.h>

```
int xosd_get_colour(xosd *osd, int *red, int *green, int *blue);
```

Description

`xosd_get_colour` returns the current colour being used to display text and graphics in an XOSD window. The value is passed back as three interger values, one for the red-component of the colour, one for the green-component, and one blue-component.

Arguments

`osd`	The XOSD window to query.
`red`	A pointer to an interger which will be set to the red-component of the colour. If *`red`* is `NULL` it will not be set.
`green`	A pointer to an interger which will be set to the green-component of the colour. If *`green`* is `NULL` it will not be set.
`blue`	A pointer to an interger which will be set to the blue-component of the colour. If *`blue`* is `NULL` it will not be set.

Return Value

On success, a zero is returned. On error -1 is returned and `xosd_error` is set to indicate the reason for the error.

Environment

`char *xosd_error`	A string describing the error, if one occurred.

Authors

The XOSD library was originally written by André Renaud, and is currently maintained by Tim Wright. This document was written by Michael JasonSmith.

Bugs

There are no known bugs with `xosd_get_colour`. Bug reports can be sent to `<xosd@ignavus.net>`.

See Also

xosd_create(3xosd), `xosd_set_colour`(3xosd)

Name

`xosd_set_shadow_offset` -- Change the shadow-offset of the XOSD window

`xosd_set_shadow_offset`

Synopsis

#include <xosd.h>

```
int xosd_set_shadow_offset(xosd *osd, int shadow_offset);
```

Description

To increase readability (and ’cos it looks cool) a black shadow can be drawn beneath the text and graphics displayed by XOSD. `xosd_set_shadow_offset` changes the size of this shadow by altering how many pixels the shadow is offset (to the bottom-right) from the main display. One to four pixels usually is sufficient to get a good effect.

Arguments

`osd`	The XOSD window to change.
`shadow_offset`	The number of pixels the shadow is offset from the main display. Initially the shadow-offset is zero.

Return Value

On success, a zero is returned. On error, -1 is returned and `xosd_error` is set to indicate the reason for the error.

Environment

`char *xosd_error`	A string describing the error, if one occurred.

Authors

The XOSD library was originally written by André Renaud, and is currently maintained by Tim Wright. This document was written by Michael JasonSmith.

See Also

`xosd_create`(3xosd), `xosd_set_shadow_colour`(3xosd), `xosd_outline_offset`(3xosd).

Name

`xosd_set_outline_offset` -- Change the size of the outline of the data displayed in the XOSD window

`xosd_set_outline_offset`

Synopsis

#include <xosd.h>

```
int xosd_set_outline_offset(xosd *osd, int offset);
```

Description

`xosd_set_outline_offset` changes the width (in pixels) of the outline which can be drawn on all text and graphics displayed in the XOSD window. This helps increase legibility, much like `xosd_set_shadow_offset`(3xosd). By default there is no outline (it is zero pixels wide); anything larger than two pixels looks odd.

Arguments

`osd`	The XOSD window to alter.
`offset`	The width of the outline in pixels

Return Value

On success a zero is returned. On error -1 is returned and `xosd_error` is set to indicate the reason for the error.

Environment

`char *xosd_error`	A string describing the error, if one occurred.

History

The `xosd_set_outline_offset` function first appeared in version 2.2 of the XOSD library.

Authors

The XOSD library was originally written by André Renaud, and is currently maintained by Tim Wright. Outline support was created by Andy Heroff, and `xosd_set_outline_offset` was written by Tim Wright. This document was written by Michael JasonSmith.

Bugs

There are no known bugs with `xosd_set_outline_offset`. Bug reports can be sent to `<xosd@ignavus.net>`.

See Also

`xosd_create`(3xosd), `xosd_set_outline_colour`(3xosd), `xosd_set_shadow_offset`(3xosd).

Name

xosd_set_outline_colour -- Change the outline colour of the data displayed the XOSD window

xosd_set_outline_colour

Synopsis

#include <xosd.h>

```
int xosd_set_outline_colour(xosd *osd, const char *colour);
```

Description

`xosd_set_outline_colour` changes the colour of the border that surrounds all text and graphics displayed in the XOSD window. By default the outline is black, but it is invisible until `xosd_set_outline_offset`(3xosd) is called.

Arguments

`osd`	The XOSD window to alter.
`colour`	The colour of the outline. The colour can be specified by name, such as `"green"`, or by a hexadecimal colour specifier, such as `"#5eff72"`. (For a complete list of valid colour names see the file `rgb.txt` in you X11 distribution.)

Return Value

On success a zero is returned. On error -1 is returned and `xosd_error` is set to indicate the reason for the error.

Environment

`char *xosd_error`	A string describing the error, if one occurred.
`rgb.txt`	A list of all valid colour names. It is normally distributed as part of X11.

History

The `xosd_set_outline_colour` function first appeared in version 2.2 of the XOSD library.

Authors

The XOSD library was originally written by André Renaud, and is currently maintained by Tim Wright. Outline support was created by Andy Heroff, and `xosd_set_outline_colour` was written by Tim Wright. This document was written by Michael JasonSmith.

Bugs

There are no known bugs with `xosd_set_outline_colour`. Bug reports can be sent to `<xosd@ignavus.net>`.

See Also

`xosd_create`(3xosd), `xosd_set_outline_offset`(3xosd), `xosd_set_colour`(3xosd).

Name

xosd_set_font -- Change the font used by the XOSD window

xosd_set_font

Synopsis

#include <xosd.h>

```
int xosd_set_font(xosd *osd, const char *font);
```

Description

`xosd_set_font` changes the font used to display the text shown in the XOSD window. The font is specified using an X Logical Font Descriptor (XLFD); the easiest way to create an XLFD is to use a program such as xfontsel(1). By default the `"fixed"` font is used, which is normally too small to see clearly.

Arguments

`osd`	The XOSD window to change.
`font`	The font used by the text.

Return Value

On success, a zero is returned. On error, -1 is returned and `xosd_error` is set to indicate the reason for the error.

Environment

`char *xosd_error`	A string describing the error, if one occurred.

Authors

The XOSD library was originally written by André Renaud, and is currently maintained by Tim Wright. This document was written by Michael JasonSmith.

Bugs

There are no known bugs with `xosd_set_font`. Bug reports can be sent to `<xosd@ignavus.net>`.

See Also

xosd_create(3xosd), xfontsel(1)

Name

`xosd_set_timeout` -- Change the time before the XOSD window is hidden

`xosd_set_timeout`

Synopsis

#include <xosd.h>

```
int xosd_set_timeout(xosd *osd, int timeout);
```

Description

By default, an XOSD window is never removed from the display unless `xosd_hide`(3xosd) is called. Calling `xosd_set_timeout` changes this behaviour so the window is automatically hidden *`timeout`* seconds after `xosd_display`(3xosd) is called.

Arguments

`osd`	The XOSD window to change.
`timeout`	The number of seconds that should pass before the XOSD window is removed from the display. If *`timeout`* is set to `-1` then the XOSD window is never removed from the display unless `xosd_hide`(3xosd) is called.

Return Value

On success, a zero is returned. On error, -1 is returned and `xosd_error` is set to indicate the reason for the error.

Environment

`char *xosd_error`	A string describing the error, if one occurred.

Authors

The XOSD library was originally written by André Renaud, and is currently maintained by Tim Wright. This document was written by Michael JasonSmith.

See Also

`xosd_create`(3xosd), `xosd_hide`(3xosd), `xosd_display`(3xosd)

Name

xosd_get_number_lines -- Get the number of lines able to be shown by the XOSD window

xosd_get_number_lines

Synopsis

#include <xosd.h>

```
int xosd_get_colour(xosd *osd);
```

Description

`xosd_get_number_lines` returns the maximum number of lines allowed in an XOSD window. The *`lines`* argument to `xosd_display`(3xosd) cannot be greater than the return value of `xosd_get_number_lines`.

Arguments

`osd`	The XOSD window to query.

Return Value

On success the maximum number of lines allowed in the XOSD window is returned. On error `-1` is returned and `xosd_error` is set to indicate the reason for the error.

Environment

`char *xosd_error`	A string describing the error, if one occurred.

Authors

The XOSD library was originally written by André Renaud, and is currently maintained by Tim Wright. This document was written by Michael JasonSmith.

Bugs

There are no known bugs with `xosd_get_number_lines`. Bug reports can be sent to `<xosd@ignavus.net>`.

See Also

xosd_create(3xosd), `xosd_display`(3xosd)

Name

`xosd_set_bar_length` -- Change the length of the percentage bar or slider

`xosd_set_bar_length`

Synopsis

#include <xosd.h>

```
int xosd_set_bar_length(xosd *osd, int displayPercentage);
```

Description

`xosd_set_bar_length` changes the percentage of the display used by a slider or percentage bar. Normally the XOSD choses a sensible length for the bar, but you may wish to change the default behavior if there are only a small number of possible values to be displayed.

Arguments

`osd`	The XOSD window to alter.
`displayPercentage`	The percentage of the display to be used up by the slider or percentage bar, as an interger between `0` and `100`. Setting *`displayPercentage`* to `-1` reverts to the default behaviour.

Return Value

On success, a zero is returned. On error, -1 is returned and `xosd_error` is set to indicate the reason for the error.

Environment

`char *xosd_error`	A string describing the error, if one occurred.

Bugs

There are no known bugs with `xosd_set_bar_length`. Bug reports can be sent to `<xosd@ignavus.net>`.

Authors

The XOSD library was originally written by André Renaud, and is currently maintained by Tim Wright. This document was written by Michael JasonSmith.

See Also

`xosd_create`(3xosd), `xosd_display`(3xosd).

Displaying Data

For most applications, only the `xosd_display` function will ever have to be called. However, other functions are available to provide fine-grained control of when XOSD windows are removded from the display.

Name

xosd_display -- Display data to an XOSD window

xosd_display

Synopsis

#include <xosd.h>

```
int xosd_display(xosd *osd, int line, xosd_command command, ...);
```

Description

`xosd_display` displays either a string, a percentage bar (like a progress bar) or a slider on an X display. The data is displayed in a borderless shaped-window, so it appears to float on top of the other windows, like the on-screen displays found in many televisions. Users cannot interact with the display in any way.

The data is displayed until the timeout limit, set by calling `xosd_set_timeout`(3xosd), is reached, but `xosd_display` returns immediately. If blocking is required `xosd_wait_until_no_display`(3xosd) should be called after `xosd_display`. A window that is displaying data can be hidden by calling `xosd_hide`(3xosd).

The type of data displayed is determined by the *`command`* argument. There are two types of data that can be displayed: text or integers.

Displaying Textual Data

Text is normally displayed by passing `XOSD_string` as the argument to *`command`*, followed by a string in UTF-8 format. If formatted text is desired, pass `XOSD_printf` as the argument to *`command`*, followed by string that has the same format as `printf`(3), and as many additional arguments as is required by the format string.

Displaying Integer Values

Integer values (which must be within the range 0 to 100) can be displayed in two ways: a percentage-bar or a slider. A percentage bar looks like a volume display on a TV, and is created by passing `XOSD_percentage` as the argument to *`command`*. A slider (see the XOSD plug-in for xmms(1) for an example) is created by passing `XOSD_slider` as the argument to *`command`*. An int between 0 and 100 is expected as the final argument when either `XOSD_percentage` or `XOSD_slider` is passed as the argument to *`command`*.

Arguments

`osd`	The XOSD window to use as the display.
`line`	The line of the display to use. Lines are numbered from the topmost line (line 0). The value of *`line`* must be less than *`number_lines`*, set in the call to `xosd_create`(3xosd).
`command`	One of `XOSD_percentage`, `XOSD_slider` or `XOSD_string`. If the value of *`command`* is `XOSD_string`, then the next argument should be a string in UTF-8 format. If `XOSD_percentage` or `XOSD_slider` is given then an int between 1 and 100 is expected as the next argument.

Return Value

If the *`command`* is either `XOSD_percentage` or `XOSD_slider` then the integer value of the bar or slider is returned (between 1 and 100). For `XOSD_string` and `XOSD_printf` the number of characters written to the display is returned.

On error -1 is returned and `xosd_error` is set to indicate the reason for the error.

Environment

`char *xosd_error`	A string describing the error, if one occurred.
enum xosd_command	The type of information that can be displayed, defined as an enumerated type. There are four values defined: • `XOSD_percentage`, • `XOSD_string`, • `XOSD_printf`, and • `XOSD_slider`.

Authors

The XOSD library was originally written by André Renaud, and is currently maintained by Tim Wright. This document was written by Michael JasonSmith.

Bugs

There are no known bugs with `xosd_display`. Bug reports can be sent to `<xosd@ignavus.net>`.

See Also

`xosd_create`(3xosd), `xosd_set_timeout`(3xosd), `xosd_wait_until_no_display`(3xosd), `xosd_hide`(3xosd), `printf`(3)

Name

xosd_show -- Make the XOSD window visible

xosd_show

Synopsis

#include <xosd.h>

```
int xosd_show(xosd *osd);
```

Description

`xosd_show` is called to redisplay data that has been previously displayed using `xosd_display`(3xosd). The display will remain displayed until the timeout limit, set by calling `xosd_set_timeout`(3xosd), is reached. The display can be explicitly hidden by calling `xosd_hide`(3xosd).

Arguments

`osd`	The XOSD window to show.

Return Value

On success, a zero is returned. On error, -1 is returned and `xosd_error` is set to indicate the reason for the error.

Environment

`char *xosd_error`	A string describing the error, if one occurred.

Bugs

There are no known bugs with `xosd_show`. Bug reports can be sent to `<xosd@ignavus.net>`.

Authors

The XOSD library was originally written by André Renaud, and is currently maintained by Tim Wright. This document was written by Michael JasonSmith.

See Also

`xosd_create`(3xosd), `xosd_display`(3xosd), `xosd_set_timeout`(3xosd), `xosd_hide`(3xosd).

Name

`xosd_hide` -- Make the XOSD window invisible

`xosd_hide`

Synopsis

#include <xosd.h>

```
int xosd_hide(xosd *osd);
```

Description

`xosd_hide` is called to unmap (hide) an XOSD window that is currently displaying data. Normally the XOSD window will be visible until the timeout limit, set by calling `xosd_set_timeout`(3xosd), is reached; `xosd_hide` is used to prematurely remove the XOSD window from the display. To redisplay the XOSD window call `xosd_show`(3xosd).

Arguments

`osd`	The XOSD window to hide.

Return Value

On success, a zero is returned. On error, -1 is returned and `xosd_error` is set to indicate the reason for the error.

Environment

`char *xosd_error`	A string describing the error, if one occurred.

Bugs

There are no known bugs with `xosd_hide`. Bug reports can be sent to `<xosd@ignavus.net>`.

Authors

The XOSD library was originally written by André Renaud, and is currently maintained by Tim Wright. This document was written by Michael JasonSmith.

See Also

`xosd_create`(3xosd), `xosd_display`(3xosd), `xosd_set_timeout`(3xosd), `xosd_show`(3xosd).

Name

xosd_scroll -- Scroll the XOSD window

xosd_scroll

Synopsis

#include <xosd.h>

```
int xosd_scroll(xosd *osd, int lines);
```

Description

xosd_scroll is called to scroll the XOSD window up so the bottom lines are blank. The number of lines that can be scrolled cannot be greater than the total number of lines in a XOSD window, which is set when the window is created by xosd_create(3xosd), and cannot be changed.

Arguments

osd	The XOSD window to scroll.
lines	The number of lines to scroll.

Return Value

On success, a zero is returned. On error, -1 is returned and xosd_error is set to indicate the reason for the error.

Environment

char *xosd_error	A string describing the error, if one occurred.

Bugs

There are no known bugs with xosd_scroll. Bug reports can be sent to <xosd@ignavus.net>.

Authors

The XOSD library was originally written by André Renaud, and is currently maintained by Tim Wright. This document was written by Michael JasonSmith.

See Also

xosd_create(3xosd), xosd_display(3xosd).

Name

`xosd_is_onscreen` -- Returns wether the XOSD window is shown

`xosd_is_onscreen`

Synopsis

#include <xosd.h>

```
xosd *xosd_is_onscreen(xosd *osd);
```

Description

`osd_is_onscreen` determines weather a XOSD window, is currently being shown (is mapped to the X display). Because XOSD displays data asynchronously (see `xosd_display`(3xosd) for details) it can be difficult to know if data is being displayed, `xosd_is_onscreen` solves this problem.

Call `xosd_show`(3xosd) or `xosd_hide`(3xosd) to alter the visibility of the XOSD window.

Arguments

`osd`	The XOSD window to query.

Return Value

A `1` is returned if the window is onscreen (mapped), or `0` if it is hidden (unmapped). On error, `-1` is returned and `xosd_error` is set to indicate the reason for the error.

Environment

`char *xosd_error`	A string to a text string describing the error, if one occurred.

History

The `xosd_is_onscreen` function first appeared in version 2.1 of the XOSD library.

Authors

The XOSD library was originally written by André Renaud and is currenly maintained by Tim Wright, who also wrote the `xosd_is_onscreen` function. Michael JasonSmith thinks he wrote this document, but is not sure; drop Micahel an email (`<mike@ldots.org>`) if you think he didn't write this document.

Bugs

There are no known bugs with `xosd_is_onscreen`. Bug reports can be sent to `<xosd@ignavus.net>`.

See Also

`xosd_display`(3xosd), `xosd_show`(3xosd), `xosd_hide`(3xosd)

Name

`xosd_wait_until_no_display` -- Suspend execution until the XOSD window is hidden

`xosd_wait_until_no_display`

Synopsis

#include <xosd.h>

```
int xosd_wait_until_no_display(xosd *osd);
```

Description

`xosd_wait_until_no_display` suspends execution of the calling program until the XOSD window is no-longer displayed because a timeout, set by `xosd_set_timeout`(3xosd), has expired. This behavior can be emulated by removing the timeout for the window, calling `xosd_show`(3xosd), sleeping, and calling `xosd_hide`(3xosd).

Arguments

`osd`	The XOSD window that is being waited on.

Return Value

On success, a zero is returned. On error, -1 is returned and `xosd_error` is set to indicate the reason for the error.

Environment

`char *xosd_error`	A string describing the error, if one occurred.

Bugs

There are no known bugs with `xosd_wait_until_no_display`. However, the calling program can dead-lock by removing the timeout and calling `xosd_wait_until_no_display`. Bug reports can be sent to `<xosd@ignavus.net>`.

Authors

The XOSD library was originally written by André Renaud, and is currently maintained by Tim Wright. This document was written by Michael JasonSmith.

See Also

`xosd_create`(3xosd), `xosd_display`(3xosd), `xosd_set_timeout`(3xosd), `xosd_show`(3xosd), `xosd_hide`(3xosd).

Command-Line Utilities

The two command-line utilities that ship with XOSD are used for quite different functions. **osd_cat** provides a command-line interface to the various XOSD library calls, while **xosd-config** is used to aid in compiling programs that use `libxosd`.

Name

osd_cat -- X on-screen file displayer

osd_cat

Synopsis

```
osd_cat  [Option...] [File...]
```

Description

osd_cat displays the contents of `File`, or standard input, in a unmanaged shaped window. The effect is similar to the on-screen display of many televisions and video recorders.

Options

`-p, --pos=`*`POS`*	The vertical position of the text. *`POS`* can be `top`, `middle`, or `bottom`. The default is `top`.
`-o, --offset=`*`OFFSET`*	The number of pixels the text is offset from the top or bottom of the display. The default is `0`.
`-A, --align=`*`ALIGN`*	The horizontal alignment of the text. *`ALIGN`* can be `left`, `centre` or `right`. The default is `centre`.
`-i, --indent=`*`INDENT`*	The number of pixels the text is indented from the left or right of the display. The default is `0`.
`-f, --font=`*`FONT`*	The font used to display the text. The default is `fixed`, which may be too small to see clearly.
`-c, --colour=`*`COLOUR`*	The text colour. The default is `red`.
`-d, --delay=`*`TIME`*	The number of seconds the text is displayed before being removed from the display. The default is `5`.
`-l, --lines=`*`LINES`*	The maximum number of lines that can be displayed. The default is `5`.
`-s, --shadow=`*`SHADOW`*	The number of pixels the shadow is offset behind the text. The default is `0`, so no shadow is displayed.
`-S, --shadowcolour=`*`COLOUR`*	The colour of the shadow. The default is `black`.
`-O, --outline=`*`OUTLINE`*	The width of the outline, in pixels. The default is `0`, so no outline is displayed.
`-u, --outlinecolour=`*`COLOUR`*	The colour of the outline. The default is `black`.
`-a, --age=`*`SCROLL_AGE`*	This option affects screen redrawing. If *`SCROLL_AGE`* seconds pass before a new line is ready (for example, you're reading from a pipe), the display is cleared instead of being scrolled. The default is `0`, which means all lines are added to the scroll.

`-w, --wait=`*`SCROLL_AGE`* This option also affects screen redrawing. When there is data ready to be put on screen, this option will cause osd_cat to wait until the display is clear. An alternative to scrolling.

Environment

`DISPLAY` The X11 display to use.

Example

Display new entries in `/var/log/messages` in an OSD window.

```
bash$ FONT="-adobe-helvetica-bold-*-*-*-34-*-*-*-*-*-*-*"
bash$ tail -f /var/log/messages | osd_cat --font=$FONT --shadow=2
```

History

The osd_cat application first appeared in version 0.3 of the XOSD.

Authors

Martijn van de Streek `<martijn@foodfight.org>`, with some patching by Malcolm Valentine `<farkit@iprimus.com.au>` and Tim Wright `<tim@ignavus.net>`.

The XOSD library was originally written by André Renaud, and is currently maintained by Tim Wright.

Bugs

There are no known bugs with osd_cat. Bug reports can be sent to `<xosd@ignavus.net>`.

See Also

More information on the X OSD Library and its author can be found at http://www.ignavus.net/software.html.

Copyright

osd_cat is distributed under the GNU General Public License.

Name

xosd-config -- Get information about the installed version of libxosd

xosd-config

Synopsis

```
xosd-config  [--prefix [=DIR]] [--exec-prefix [=DIR]] [--version] [--libs] [--cflags]
```

Description

xosd-config is a tool that is used to determine the compiler and linker flags that should be used to compile and link programs that use `libxosd`. It is also used internally by the .m4 macros for GNU autoconf that are included with `libxosd`.

Options

`--version`	Print the currently installed version of `libxosd` on standard output.
`--libs`	Print the linker flags that are necessary to link a program that uses `libxosd`.
`--cflags`	Print the compiler flags that are necessary to compile a `libxosd` program.
`--prefix=`*`DIR`*	If specified, use *`DIR`* instead of the installation prefix that `libxosd` was built with when computing the output for the `--cflags` and `--libs` options. This option is also used for the exec prefix if `--exec-prefix` was not specified. This option must be specified before any `--libs` or `--cflags` options.
`--exec-prefix=`*`DIR`*	If specified, use *`DIR`* instead of the installation exec prefix that `libxosd` was built with when computing the output for the `--cflags` and `--libs` options. This option must be specified before any `--libs` or `--cflags` options.

Bugs

There are no known bugs with xosd-config. Bug reports can be sent to `<xosd@ignavus.net>`.

Copyright

This manual page was adapted by Philipp Hahn `<pmhahn@titan.lahn.de>`, for the Debian GNU/Linux system (but may be used by others).

Colophon

This document was written in XEmacs 21.4 [http://www.xemacs.org/] using the Docbook XML 4.1.2 [http://www.docbook.org/] markup scheme. The `psgml` mode in XEmacs was invaluable in preventing me from having to remember all the tags!.

The fonts used to present this document will depend on how the Docbook source was processed. The author uses Passive TeX to produce PDF files, so the Times, Courier and Helvetica are the mainly used. The authors Web Browser (Galeon [http://galeon.sourceforge.net/]) is set up to use Caxton as the Serif font, Futura as the Sans serif and Fixed as the Monospace font.