NAME

XAllocColor, XAllocNamedColor, XAllocColorCells, XAllocColorPlanes, XFreeColors - allocate and free colors

SYNTAX

Status XAllocColor(Display *_display_, Colormap _colormap_, XColor *_screen_in_out_);

Status XAllocNamedColor(Display *_display_, Colormap _colormap_, _Xconst char *_color_name_, XColor *_screen_def_return_, XColor *_exact_def_return_);

Status XAllocColorCells(Display *_display_, Colormap _colormap_, Bool _contig_, unsigned long _plane_masks_return_[], unsigned int _nplanes_, unsigned long _pixels_return_[], unsigned int _npixels_);

Status XAllocColorPlanes(Display *_display_, Colormap _colormap_, 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_);

int XFreeColors(Display *_display_, Colormap _colormap_, unsigned long _pixels_[], int _npixels_, unsigned long _planes_);

color_name Specifies the color name string (for example, red) whose color definition structure you want returned.

colormap Specifies the colormap.

contig Specifies a Boolean value that indicates whether the planes must be contiguous.

display Specifies the connection to the X server.

exact_def_return
Returns the exact RGB values.

ncolors Specifies the number of pixel values that are to be returned in the pixels_return array.

npixels Specifies the number of pixels.

nplanes Specifies the number of plane masks that are to be returned in the plane masks array.

nreds

ngreens

nblues
Specify the number of red, green, and blue planes. The value you pass must be nonnegative.

pixels Specifies an array of pixel values.

pixels_return Returns an array of pixel values.

plane_mask_return
Returns an array of plane masks.

planes Specifies the planes you want to free.

rmask_return

gmask_return

bmask_return Return bit masks for the red, green, and blue planes.

screen_def_return
Returns the closest RGB values provided by the hardware.

screen_in_out Specifies and returns the values actually used in the colormap.

DESCRIPTION

The **XAllocColor** function allocates a read-only colormap entry corresponding to the closest RGB value supported by the hardware. **XAllocColor** returns the pixel value of the color closest to the specified RGB elements supported by the hardware and returns the RGB value actually used. The corresponding colormap cell is read-only. In addition, **XAllocColor** returns nonzero if it succeeded or zero if it failed. Multiple clients that request the same effective RGB value can be assigned the same read-only entry, thus allowing entries to be shared. When the last client deallocates a shared cell, it is deallocated. **XAllocColor** does not use or affect the flags in the **XColor** structure.

XAllocColor can generate a **BadColor** error.

The **XAllocNamedColor** function looks up the named color with respect to the screen that is associated with the specified colormap. It returns both the exact database definition and the closest color supported by the screen. The allocated color cell is read-only. The pixel value is returned in

screen_def_return. If the color name is not in the Host Portable Character Encoding, the result is implementation-dependent. Use of uppercase or lowercase does not matter. If screen_def_return and exact_def_return point to the same structure, the pixel field will be set correctly, but the color values are undefined. **XAllocNamedColor** returns nonzero if a cell is allocated; otherwise, it returns zero.

XAllocNamedColor can generate a **BadColor** error.

The **XAllocColorCells** function allocates read/write color cells. The number of colors must be positive and the number of planes nonnegative, or a **BadValue** error results. If ncolors and nplanes are requested, then ncolors pixels and nplane plane masks are returned. No mask will have any bits set to 1 in common with any other mask or with any of the pixels. By ORing together each pixel with zero or more masks, ncolors * 2^*nplanes* distinct pixels can be produced. All of these are allocated writable by the request. For **GrayScale** or **PseudoColor**, each mask has exactly one bit set to 1. For **DirectColor**, each has exactly three bits set to 1. If contig is **True** and if all masks are ORed together, a single contiguous set of bits set to 1 will be formed for **GrayScale** or **PseudoColor** and three contiguous sets of bits set to 1 (one within each pixel subfield) for **DirectColor**. The RGB values of the allocated entries are undefined. **XAllocColorCells** returns nonzero if it succeeded or zero if it failed.

XAllocColorCells can generate **BadColor** and **BadValue** errors.

The specified ncolors must be positive; and nreds, ngreens, and nblues must be nonnegative, or a **BadValue** error results. If ncolors colors, nreds reds, ngreens greens, and nblues blues are requested, ncolors pixels are returned; and the masks have nreds, ngreens, and nblues bits set to 1, respectively. If contig is **True**, each mask will have a contiguous set of bits set to 1. No mask will have any bits set to 1 in common with any other mask or with any of the pixels. For **DirectColor**, each mask will lie within the corresponding pixel subfield. By ORing together subsets of masks with each pixel value, ncolors * 2^((*nreds* + *ngreens* + *nblues*)) distinct pixel values can be produced. All of these are allocated by the request. However, in the colormap, there are only ncolors * 2^*nreds* independent red entries, ncolors * 2^*ngreens* independent green entries, and ncolors * 2^*nblues* independent blue entries. This is true even for **PseudoColor**. When the colormap entry of a pixel value is changed (using **XStoreColors**, **XStoreColor**, or **XStoreNamedColor**), the pixel is decomposed according to the masks, and the corresponding independent entries are updated. **XAllocColorPlanes** returns nonzero if it succeeded or zero if it failed.

XAllocColorPlanes can generate **BadColor** and **BadValue** errors.

The **XFreeColors** function frees the cells represented by pixels whose values are in the pixels array. The planes argument should not have any bits set to 1 in common with any of the pixels. The set of all pixels is produced by ORing together subsets of the planes argument with the pixels. The request frees all of these pixels that were allocated by the client (using **XAllocColor**, **XAllocNamedColor**,

XAllocColorCells, and **XAllocColorPlanes**). Note that freeing an individual pixel obtained from **XAllocColorPlanes** may not actually allow it to be reused until all of its related pixels are also freed. Similarly, a read-only entry is not actually freed until it has been freed by all clients, and if a client allocates the same read-only entry multiple times, it must free the entry that many times before the entry is actually freed.

All specified pixels that are allocated by the client in the colormap are freed, even if one or more pixels produce an error. If a specified pixel is not a valid index into the colormap, a **BadValue** error results. If a specified pixel is not allocated by the client (that is, is unallocated or is only allocated by another client) or if the colormap was created with all entries writable (by passing **AllocAll** to **XCreateColormap**), a **BadAccess** error results. If more than one pixel is in error, the one that gets reported is arbitrary.

XFreeColors can generate **BadAccess**, **BadColor**, and **BadValue** errors.

DIAGNOSTICS

BadAccess A client attempted to free a color map entry that it did not already allocate.

BadAccess A client attempted to store into a read-only color map entry.

BadColor A value for a Colormap argument does not name a defined Colormap.

BadValue Some numeric value falls outside the range of values accepted by the request. Unless a specific range is specified for an argument, the full range defined by the argument's type is accepted. Any argument defined as a set of alternatives can generate this error.

SEE ALSO

XCreateColormap(3), XQueryColor(3), XStoreColors(3)
Xlib - C Language X Interface